Τ.Ε.Ι – Πάτρας
Τμήμα :Ηλεκτρολογίας

Πτυχιακή Εργασία
Αριθμός 537

Αλγόριθμοι Συμπίεσης Δεδομένων

Εισηγητής:

Δ.Καρέλης

Σπουδάστες:

Ε.Αβούρης
Κ.Γκόγκας

Πάτρα – Σ.2001

Εισαγωγή

Η εποχή που ζούμε έχει, πολύ εύστοχα χαρακτηριστεί σαν 'εποχή της πληροφορικής'. Αυτό που ίσως έχει αναδείξει την αξία και την χρησιμότητα των ηλεκτρονικών υπολογιστών περισσότερο από οτιδήποτε άλλο είναι η δυνατότητα που δίνουν στον χρήστη να αξιοποιεί με τον πιο εύκολο και αποδοτικό τρόπο την πληροφορία. Ο όγκος αυτής της πληροφορίας είναι τέτοιος που χωρίς τους υπολογιστές δεν θα ήταν δυνατόν να την επεξεργαστούμε. Από την άλλη γνωρίζούμε ότι η πληροφορία είναι γνώση οπότε γίνεται φανερός ο ρόλος της πληροφορικιής στην ανθρωπότητα.

Στην προσπάθεια για ακόμα πιο εύκολη και πρακτική επεξεργασία της πληροφορίας καθώς και για την επίλυση κάποιον προβλημάτων που προέκυπταν έχουν αναπτυχθεί κάποιοι αλγόριθμοι συμπίεσης δεδομένων. Πρέπει να πούμε εδώ ότι καθημερινά κάθε ένας από εμάς έρχεται μπροστά σε εφαρμογές συμπίεσης δεδομένων. Όποιαδηποτε μεταφορά δεδομένων μέσω υπολογιστικών ή ακόμα και δορυφορικών συστημάτων, οποιαδήποτε επεξεργασία εικόνας, ήχου, ή βίντεο είναι αποτέλεσμα συμπίεσης. Αυτό σημαίνει ότι βάσεις δεδομένων, fax, modems, ιατρικές, multimedia εφαρμογές και άλλες είναι όλα πεδία στα οποία χρησιμοποιούμε τους αλγόριθμους αυτούς.

Τρείς τέτοιους αλγόριθμους θα εξετάσουμε στο δεύτερο κεφάλαιο αυτής της πτυχιακής. Στο πρώτο μέρος θα δούμε κάποια πράγματα εισαγωγικά για τη συμπίεση και τις εφαρμογές της. Τέλος στο τρίτο και τελευταίο μέρος θα εξετάσουμε τι συμβαίνει με τις εικόνες.

Θα θέλαμε να ευχαριστήσουμε τον καθηγητή μας και εισηγητή αυτής της εργασίας επίκουρο καθηγητή Τ.Ε.Ι. κύριο Δ. Καρέλη για την βοήθεια του και για την βιβλιογραφία που μας παραχώρησε.

ΠΕΡΙΕΧΟΜΕΝΑ

1. Λογισμικό και χρησιμοποίησή του

Στη χρονολογία της ανάπτυξης των υπολογιστών, έχει βεβαιωθεί μια τεράστια ανάπτυξη στη μεταφορά πληροφοριών μεγάλης κλίμακας και στην ανάπτυξη συστημάτων μαζικής αποθήκευσης και ανάκτησης πληροφοριών. Παράλληλα με την ανάπτυξη αυτή, έχουν δημιουργηθεί διάφορες προβληματικές περιοχές που μπορούν να προκαλέσουν μεγάλες, αλλά μη αναγκαίες, οικονομικές δαπάνες.

'Ενα πρόβλημα είναι η αποκαλούμενη "απομακρυσμένη (run-away) βάση δεδομένων". Εδώ το μέγεθος της βάσης δεδομένων που χρησιμοποιείται από έναν οργανισμό για τα προγράμματά του αποθήκευσης και ανάκτησης πληροφοριών γίνεται όλο και μεγαλύτερο, απαιτώντας συμπληρωματικές συσκευές χειρισμού δίσκων (disk-drives) για συστήματα εν γραμμή (online).

Ένα άλλο πρόβλημα για τις προοπτικές αποθήκευσης δεδομένων είναι η αυξανόμενη χρήση εφαρμογών πολυμέσων σε υπολογιστές. Μια οθόνη κειμένου που αποτελείται από 80 στήλες επί 25 σειρές περιέχει ένα μέγιστο 2000 χαρακτήρων. Συγκριτικά, σκεφθείτε μια VGA 640 από μια 16χρωμη εικόνα των 480 pixel που εμφανίζεται σε μια οθόνη προσωπικού υπολογιστή. Η εικόνα θα απαιτούσε 640 χ 480 bits αποθήκευσης χωρίς να ληφθεί υπόψη το γεγονός ότι τέσσερα bits απαιτούνται για την αναπαράσταση του χρώματος κάθε pixel. 'Ετσι, η εικόνα θα απαιτούσε 153.600 bytes αποθήκευσης, ή περίπου 77 φορές

την αποθήκευση που απαιτείται για την αναπαράσταση μιας πλήρους οθόνης κειμένου.

Σε συνοδεία της ανάπτυξης του μεγέθους των βάσεων δεδομένων και της χρήσης των εικόνων έχει έρθει μια μεγάλη αύξηση στον αριθμό χρηστών και στη διάρκεια χρήσης από το προσωπικό από απομακρυσμένες τοποθεσίες. Οι παράγοντες αυτοί έχουν ως αποτέλεσμα τεράστια ποσά δεδομένων να μεταφέρονται μεταξύ των υπολογιστών και απομακρυσμένων τερματικών. Για την παροχή διευκολύνσεων μετάδοσης για τις απαιτούμενες μεταφορές δεδομένων, γραμμές επικοινωνίας και οι βοηθητικές συσκευές, όπως τα μόντεμ και οι πολλαπλοί μεταβιβαστές (multiplexers), συνεχώς αναβαθμίζονται από πολλούς οργανισμούς για να επιτρέψουν την ικανότητα υψηλότερης μετάδοσης δεδομένων. Παρόλο που οι φανερές λύσεις στα προβλήματα αυτά της αποθήκευσης δεδομένων και της μετάδοσης πληροφοριών είναι να εγκαθίστανται συσκευές συμπληρωματικής αποθήκευσης και να επεκτείνονται οι υπάρχουσες εγκαταστάσεις επικοινωνιών, για να γίνει κάτι τέτοιο απαιτείται μια επιπρόσθετη αύξηση στον εξοπλισμό του οργανισμού και στα λειτουργικά κόστη. Μια μέθοδος που μπορεί να εφαρμοσθεί για την ελάφρυνση ενός μέρους των προβλημάτων αποθήκευσης δεδομένων και μεταφοράς πληροφοριών είναι μέσω της αναπαράστασης των δεδομένων από περισσότερο αποτελεσματικούς κώδικες. Αν εξετάζεις μια βάση δεδομένων ενός οργανισμού ή παρακολουθείς μια γραμμή μετάδοσης, υπάρχει μια εξαιρετική ευκαιρία οι ανεξάρτητοι χαρακτήρες που φτιάχνουν και τη βάση δεδομένων και την αλληλουχία μετάδοσης να μπορούν να κωδικοποιηθούν περισσότερο αποτελεσματικά. Δυο τεχνικές που μπορούν να έχουν ως αποτέλεσμα μια περισσότερο

αποτελεσματική αναπαράσταση κωδικοποιημένων δεδομένων είναι η λογική και η φυσική συμπίεση δεδομένων.

1.1. Λογική συμπίεση

Όταν σχεδιάζεται μια βάση δεδομένων, ένα από τα πρώτα βήματα του αναλυτή είναι να επιτύχει την περισσότερο δυνατή ελαχιστοποίηση δεδομένων. Αυτή η ελαχιστοποίηση δεδομένων έχει ως αποτέλεσμα την εξάλειψη των υπεράριθμων πεδίων πληροφόρησης, ενώ αναπαριστάνει τα στοιχεία δεδομένων στα εναπομείναντα πεδία με όσο το δυνατό λιγότερους λογικούς ενδείκτες.

Παρόλο που η λογική συμπίεση εξαρτάται από τα δεδομένα και η μέθοδος που θα εφαρμοστεί μπορεί να ποικίλει με βάση την πρόβλεψη του αναλυτή, τα ακόλουθα δυο παραδείγματα θα δείξουν την ευκολία υλοποίησης και τα πλεονεκτήματα από αυτή την τεχνική συμπίεσης. Ένα απλό παράδειγμα λογικής συμπίεσης δεδομένων είναι το επαγγελματικό πεδίο σε μια βάση δεδομένων προσωπικού. Υποθέστε ότι 30 αλφαριθμητικές θέσεις κατανέμονται στο πεδίο αυτό. Εάν το πεδίο είναι κανονισμένο, επαγγέλματα όπως η 10γραμματη επαγγελματική περιγραφή "DISHWASHER" (λαντζέρης) έχουν 20 κενά τοποθετημένα στο κατάλοιπο (υπόλοιπο) του πεδίου. Κατόπιν, 30 εκατομμύρια χαρακτήρες αποθήκευσης θα απαιτούνταν για το επαγγελματικό πεδίο του 1 εκατομμυρίου εργατών. Υποθέστε το πολύ ότι υπήρχαν 32.768 ευδιάκριτα επαγγέλματα. Αντί να δηλώνεται ο επαγγελματικός τίτλος, θα μπορούσατε να κωδικοποιείτε τον ισοδύναμο

5-ψήφιο κώδικα δεδομένων, αφαιρώντας 25 θέσεις χαρακτήρων ανά πεδίο. Το μέγεθος του πεδίου θα μειωνόταν περαιτέρω με την κατανομή της δυαδικής τιμής 1 ή περισσότερων χαρακτήρων στον επαγγελματικό κώδικα. Ως παράδειγμα, ένας χαρακτήρας των 8 bit θα μπορούσε να αναπαραστήσει 2^8-1 ή 255 ξεχωριστές τιμές ή επαγγελματικούς κώδικες. Η σύνδεση δυο χαρακτήρων των 8 bit μέσω κατάλληλου λογισμικού θα παρείσχε 2^{16}-1 ή 65.535 ευκρινείς κώδικες. Αυτό θα μείωνε το μέγεθος του πεδίου από 30 σε 2 χαρακτήρες, σώζοντας 28 εκατομμύρια χαρακτήρες αποθήκευσης. Αν το μέτρημά μας ξεκινάει από το μηδέν αντί από ένα συμβατικό σημείο έναρξης του 1, ένας χαρακτήρας των 8 bit θα αναπαρίστανε 2567 κώδικες, ενώ ένας χαρακτήρας των 16 bit θα μπορούσε να εφαρμοστεί για να αναπαραστήσει 65.536 ευκρινείς κώδικες.

Ένα δεύτερο παράδειγμα λογικής συμπίεσης είναι ένα πεδίο δεδομένων. Αυτός ο τύπος πεδίου εμφανίζεται συχνά στις βάσεις δεδομένων. Κανονικά, τα αριθμητικά ισοδύναμα των υποπεδίων που αναπαριστάνουν τήν ημέρα, το μήνα και το χρόνο χρησιμοποιούνται στη θέση της χειρόγραφης αρίθμησης. Έτσι, η 01.04.81 θα αναπαρίστανε την 1 η Απριλίου 1981. Ενώ αυτή η λογική συμπίεση έχει ως αποτέλεσμα μια αποθήκευση έξι αριθμητικών χαρακτήρων, επιπρόσθετη μείωση δεδομένων μπορεί να εξαχθεί από την αποθήκευση της ημερομηνίας ως δυαδικής τιμής. Αφού η ημέρα δεν θα μπορούσε ποτέ να υπερβαίνει την 31, πέντε bits θα επαρκούσαν. για την αναπαράσταση του πεδίου δεδομένων. Παρόμοια, τέσσερα bits θα μπορούσαν να χρησιμοποιηθούν για την αναπαράσταση της τιμής του μήνα, ενώ επτά bits θα μπορούσαν να αναπαραστήσουν 127 χρόνια,

επιτρέποντας ένα σχετικό έτος που κυμαίνεται από το 1900 έως το 2027.

Η λογική συμπίεση με τη χρήση αριθμητικής και δυαδικής αναπαράστασης φαίνεται στο Σχήμα 1.1 για το προηγούμενο παράδειγμα πεδίου ημερομηνίας. Είναι ενδιαφέρον να σημειωθεί ότι η εφαρμογή της δυαδικής αναπαράστασης μειώνει το πεδίο ημερομηνίας σε 16 δυαδικά ψηφία ή δύο αλυσιδωτά συνδεμένους χαρακτήρες αποθήκευσης των 8 bit.

Χαρακτήρες χειρόγραφου	ΜΕΡΑ	ΜΗΝΑΣ	ΧΡΟΝΟΣ
Παράδειγμα	1	Απρίλης	1981
Λογική συμπίεση χρησιμοποιώντας αριθμιτική αναπαράσταση			
Παράδειγμα	01	04	81
Λογική συμπίεση χρησιμοποιώντας δυαδική αναπαράσταση			
Παράδειγμά	00001	0100	1010001

Σχήμα 1.1. Μέθοδοι λογικής συμπίεσης. Η λογική συμπίεση μπορεί να είναι αποτέλεσμα αλφαριθμητικής, αριθμητικής ή δυαδικής αναπαράστασης δεδομένων σε μια στενογραφική σημείωση.

Όπως αναφέρθηκε, πολλές μέθοδοι λογικής συμπίεσης μπορούν να ληφθούν υπόψη από έναν αναλυτή κατά τη διάρκεια της διαδικασίας σχεδιασμού βάσης δεδομένων. Κάθε μέθοδος μπορεί να έχει ως αποτέλεσμα έναν ευκρινή βαθμό μείωσης αποθήκευσης δεδομένων. Αντίστοιχα, όταν οι λογικά συμπιεσμένες βάσεις δεδομένων ή τμήματα τέτοιων βάσεων μεταδίδονται μεταξύ τοποθεσιών, ο χρόνος μετάδοσης μειώνεται, αφού μεταδίδονται λιγότεροι χαρακτήρες.

Ενώ η λογική συμπίεση μπορεί να είναι ένα αποτελεσματικό εργαλείο στην ελαχιστοποίηση του μεγέθους μιας βάσης δεδομένων, μειώνει το χρόνο μετάδοσης μόνο όταν μεταδίδονται λογικά

συμπιεσμένα δεδομένα. Έτσι, η μετάδοση ερωτήσεων και αποκρίσεων, που τυπικά κωδικοποιούνται ως ξεχωριστές και ευκρινείς ολότητες στην κατάλληλη αναπαράσταση bit του κώδικα για κάθε χαρακτήρα, δεν επηρεάζεται κανονικά. Παρόμοια, δεν θα επηρεαστεί η ύπαρξη επαναλαμβανόμενων μοντέλων και ομάδων χαρακτήρων, που κανονικά περιέχονται σε αναφορές που μεταδίδονται μεταξύ υπολογιστών ή από υπολογιστικά συστήματα σε τερματικές συσκευές. Για τέτοιες καταστάσεις, μια μείωση στο χρόνο μετάδοσης δεδομένων εξαρτάται από τη φυσική συμπίεση των δεδομένων κατά την ώρα που θα συναντηθεί (θα αντιμετωπιστεί).

1.2. Φυσική συμπίεση

Η φυσική συμπίεση μπορεί να θεωρηθεί ως η διαδικασία της μείωσης της ποσότητας των δεδομένων πριν την είσοδο τους σε ένα μέσο μετάδοσης και την επέκταση των δεδομένων αυτών στο αρχικό σχήμα τους (format) στη λήψη μιας απομακρυσμένης τοποθεσίας. Παρόλο που και η φυσική και η λογική συμπίεση μπορούν να έχουν ως αποτέλεσμα έναν μειωμένο χρόνο μετάδοσης, μεταξύ των δυο τεχνικών υπάρχουν ευκρινείς διαφορές εφαρμογής. Η λογική συμπίεση κανονικά χρησιμοποιείται για την πιο αποτελεσματική αναπαράσταση βάσεων δεδομένων και δεν λαμβάνει υπόψη τη συχνότητα εμφάνισης χαρακτήρων ή ομάδων χαρακτήρων.

Η φυσική συμπίεση επωφελείται από το γεγονός ότι όταν τα δεδομένα κωδικοποιούνται ως ξεχωριστές και ευκρινείς οντότητες, οι

πιθανότητες εμφάνισης των χαρακτήρων και των ομάδων χαρακτήρων διαφέρουν. Αφού οι συχνά εμφανιζόμενοι χαρακτήρες κωδικοποιούνται σε τόσα bits όσα εκείνοι οι χαρακτήρες που εμφανίζονται σπάνια, η μείωση δεδομένων γίνεται δυνατή από την κωδικοποίηση των συχνά εμφανιζομενων χαρακτήρων σε σύντομους κώδικες bit, ενώ η αναπαράσταση μη συχνά εμφανιζόμενων χαρακτήρων σε πιο μακρούς κώδικες bit. Όπως στη λογική συμπίεση, υπάρχουν πολλές τεχνικές φυσικής συμπίεσης. Μερικές τεχνικές αντικαθιστούν επαναλαμβανόμενες αλληλουχίες χαρακτήρων από έναν ειδικό χαρακτήρα ένδειξης συμπίεσης και έναν χαρακτήρα μέτρησης ποσότητας. Άλλες τεχνικές αντικαθιστούν τους συχνά εμφανιζόμενους χαρακτήρες ή αλληλουχίες χαρακτήρων με έναν σύντομο δυαδικό κώδικα, ενώ οι μη συχνά συναντώμενοι χαρακτήρες και αλληλουχίες αντικαθίστανται από μακρύτερους δυαδικούς κώδικες. Στο Κεφάλαιο 2, ένας αριθμός ευκρινών μεθόδων φυσικής συμπίεσης καλύπτονται με λεπτομέρεια.

1.3. Πλεονεκτήματα της συμπίεσης

Όταν η συμπίεση δεδομένων χρησιμοποιείται για τη μείωση των απαιτήσεων αποθήκευσης, ο συνολικός χρόνος εκτέλεσης του προγράμματος μπορεί να μειωθεί. Αυτό συμβαίνει επειδή η μείωση στην αποθήκευση θα έχει ως αποτέλεσμα μια μείωση των προσπαθειών πρόσβασης του δίσκου, ενώ η κωδικοποίηση και αποκωδικοποίηση που απαιτούνται από την τεχνική συμπίεσης που

εφαρμόζεται θα έχουν ως αποτέλεσμα να εκτελούνται επιπρόσθετες προγραμματικές λειτουργίες. Αφού ο χρόνος εκτέλεσης μιας ομάδας προγραμματικών οδηγιών είναι συνήθως σημαντικά λιγότερος από το χρόνο που απαιτείται για την πρόσβαση και μεταφορά δεδομένων σε μια περιφερειακή διάταξη, ο συνολικός χρόνος εκτέλεσης του προγράμματος μπορεί να μειωθεί.

Σε αντιστοιχία με τη μετάδοση των δεδομένων, η συμπίεση παρέχει στο σχεδιαστή δικτύου διάφορα πλεονεκτήματα επιπρόσθετα στη δυνητική εξοικονόμηση κόστους, που σχετίζονται με την αποστολή λιγότερων δεδομένων από το τηλεφωνικό δίκτυο όπου το κόστος κλήσης συνήθως βασίζεται από τη διάρκειει της. Πρώτα, η συμπίεση μπορεί να μειώσει την πιθανότητα εμφανιζόμενων λαθών μετάδοσης, αφού μεταδίδονται λιγότεροι χαρακτήρες όταν τα δεδομένα είναι συμπιεσμένα, ενώ η πιθανότητα εμφανιζόμενου λάθους παραμένει σταθερή. Δεύτερο, αφού η συμπίεση αυξάνει την αποτελεσματικότητα, μπορεί να μειώσει ή τις υπερωρίες. Τέλος, με την μετατροπή κειμένου, που αναπαριστάνεται με έναν συμβατικό (τυπικό) κώδικα όπως ο τυποποιημένος ASCIi, σε ένα διαφορετικό κώδικα, οι αλγόριθμοι συμπίεσης μπορούν να παρέχουν ένα επίπεδο ασφάλειας ενάντια στην παράνομη παρακολούθηση.

Για επικοινωνίες δεδομένων, η μεταφορά συμπιεσμένων δεδομένων μέσο ενός διαμέσου έχει ως αποτέλεσμα μια αύξηση του αποτελεσματικού λόγου μεταφοράς πληροφοριών, ακόμα και αν ο πραγματικός λόγος μεταφοράς δεδομένων που εκφράζεται σε bits ανά δευτερόλεπτο παραμένει ο ίδιος. Η συμπίεση δεδομένων μπορεί να υλοποιηθεί στους περισσότερους υπάρχοντες μηχανικούς εξοπλισμούς υπολογιστή (hardware) από λογισμικό (software) ή μέσό της χρήσης

ειδικών συσκευών hardware που ενσωματώνουν μια ή περισσοτερες τεχνικές συμπίεσης.

Η συμπίεση και αποσυμπίεση μπορεί να εμφανίζονται μέσα σε έναν επεξεργαστή δομημένο σ' έναν προσωπικό υπολογιστή, ένα λογικό τερματικό ή μια διάταξη (συσκευή) ξένη προς τον επεξεργαστή, τέτοια όπως ένα εξειδικευμένο συστατικό επικοινωνιών. Πρώτοι μεταξύ των συστατικών αυτών εμφανίστηκαν οι συγκεντρωτές δεδομένων και οι στατιστικοί πολυπλέκτες, στη διάρκεια του `70 και των αρχών του `80. Περίπου από τα μέσα του '80, μια επανάσταση στη χρησιμοποίηση των προϊόντων απόδοσης συμπίεσης δεδομένων εμφανίστηκε στις περιοχές των μόντεμ διακοπτόμενων δικτύων, απομακρυσμένων γεφυρών, δρομολογητών, προσωπικών υπολογιστών και κύριων πλαισίων αποθήκευσης δίσκου. Κυριοελκτικά εκατοντάδες κατασκευαστές μόντεμ διακοπτόμενων δικτύων, γεφυρών και δρομολογητών ενσωμάτωσαν στα προϊόντα τους μια ποικιλία αλγορίθμων συμπίεσης δεδομένων, ενώ αρκετοί κατασκευαστές λογισμικού πατεντάρισαν προγράμματα που αυξάνουν την ικανότητα αποθήκευσης των δίσκων με τη συμπίεση δεδομένων πριν την αποθήκευση.

Για την εξέταση σε λεπτομέρεια ενός μέρους των πλεονεκτημάτων που μπορούν να προέλθουν από την εφαρμογή μιας ή περισσότερων τεχνικών συμπίεσης, απαιτείται η επισκόπηση της βασικής ορολογίας συμπίεσης.

1.4. Ορολογία

'Ενα μεγάλο λεξιλόγιο ορολογίας αναπτύχθηκε και συνεχίζει να αναπτύσσεται για την αναφορά στην τεχνολογία που σχετίζεται με τη συμπίεση. Σ' αυτό το εισαγωγικό κεφάλαιο, θα περιορίσουμε την εξέτασή μας της τεχνολογίας που σχετίζεται με τη συμπίεση σε δυο γενικές περιοχές ,την αποδοτικότητα συμπίεσης και τις μεθόδους συμπίεσης. Σύμφωνα μ' αυτή, θα περιγράψουμε και συζητήσουμε την ορολογία που σχετίζεται με τη συμπίεση βάσει της σχέσης της με την τεχνολογία που καλύπτεται σε κατάλληλα σημεία του βιβλίου αυτού.

1.4.1. Αποδοτικότητα συμπίεσης

Ένα αρχικό ρεύμα δεδομένων λειτουργεί βάσει ενός συγκεκριμένου αλγορίθμου για να παράγει ένα ρεύμα συμπιεσμένων δεδομένων. Η συμπίεση αυτή του αρχικού ρεύματος δεδομένων μερικές φορές αναφέρεται ως μια κωδικοποιητική διαδικασία με αποτέλεσμα το ρεύμα συμπιεσμένων δεδομένων να καλείται επίσης και ρεύμα κωδικοποιημένων δεδομένων. Αντιστρέφοντας τη διαδικασία, το ρεύμα συμπιεσμένων δεδομένων αποσυμπιέζεται ώστε να αναπαραχθεί το αρχικό ρεύμα δεδομένων. Αφού αυτή η διαδικασία αποσυμπίεσης έχει ως αποτέλεσμα την αποκωδικοποίηση του ρεύματος συμπιεσμένων δεδομένων, το αποτέλεσμα μερικές φορές αναφέρεται και ως ρεύμα αποκωδικοποιημένων δεδομένων. Θα

χρησιμοποιήσουμε τους όρους ρεύμα συμπιεσμένων δεδομένων και ρεύμα κωδικοποιημένων δεδομένων.

Ο βαθμός της μείωσης δεδομένων που επιτυγχάνεται ως αποτέλεσμα της διαδικασίας συμπίεσης είναι γνωστός ως λόγος συμπίεσης. Ο λόγος αυτός μετράει την ποσότητα των συμπιεσμένων δεδομένων σε σύγκριση με την ποσότητα των αρχικών δεδομένων

$$\text{Λόγος συμπίεσης} = \frac{\text{Μήκος αλληλουχίας αρχικών δεδομένων}}{\text{Μήκος αλληλουχίας συμπιεσμένων δεδομένων}}$$

Από την παραπάνω εξίσωση φαίνεται ότι όσο υψηλότερος είναι ο λόγος συμπίεσης, τόσο περισσότερο αποτελεσματική είναι η τεχνική συμπίεσης που εφαρμόζεται. Ένας άλλος όρος που χρησιμοποιείται όταν μιλάμε για συμπίεση είναι η αξία του πλεονεκτήματός της, όπου:

$$\text{Αξία πλεονεκτήματος} = \frac{\text{Μήκος αλληλουχίας συμπιεσμένων δεδομένων}}{\text{Μήκος αλληλουχίας αρχικών δεδομένων}}$$

Η αξία πλεονεκτήματος είναι αντίστροφη του λόγου συμπίεσης και πρέπει πάντα να είναι μικρότερη από τη μονάδα για να είναι αποτελεσματική η διαδικασία συμπίεσης. Το κλάσμα μείωσης των δεδομένων είναι ένα μείον την αξία πλεονεκτήματος. Έτσι, μια τεχνική συμπίεσης που έχει ως αποτέλεσμα ένα χαρακτήρα συμπιεσμένων δεδομένων για κάθε τρεις χαρακτήρες στο ρεύμα αρχικών δεδομένων θα έχει ένα λόγο συμπίεσης 3, μια αξία πλεονεκτήματος 0,33 και ένα κλάσμα μείωσης δεδομένων 0,66.

Ο λόγος συμπίεσης για τον τύπο αυτό δεδομένων πρέπει να είναι μονάδα,ενώ σε πολλές περιπτώσεις ο λανθασμένος σχεδιασμός ενός αλγόριθμου συμπίεσης ή η μη σωστή εφαρμογή του μπορεί να έχουν ως αποτέλεσμα ένα βαθμό επέκτασης δεδομένων, με αποτέλεσμα ένα λόγο συμπίεσης που πέφτει κάτω από τη μονάδα.

Όταν τα δεδομένα συμπιέζονται, ο λόγος συμπίεσης θα ποικίλει ανάλογα με την ευαισθησία των δεδομένων στον αλγόριθμο ή τους αλγόριθμους που χρησιμοποιούνται. 'Ετσι, θα πρέπει κανείς να επικεντρώσει την προσοχή του στο μέσο λόγο συμπίεσης και όχι σ' ένα λόγο που επιτυγχάνεται κάποια συγκεκριμένη στιγμή. Γενικά, οι καλοί αλγόριθμοι που λειτουργούν σε κείμενο μπορεί να αναμένεται ότι επιτυγχάνουν ένα μέσο λόγο συμπίεσης 2,0, ενώ οι εξαιρετικοί αλγόριθμοι που βασίζονται σε πολύπλοκες τεχνικές επεξεργασίας θα επιτύχουν ένα μέσο λόγο συμπίεσης που υπερβαίνει το 3,0.

1.4.2. Μέθοδοι συμπίεσης

Οι τεχνικές συμπίεσης μπορούν να ταξινομηθούν σε μια από δυο γενικές κατηγορίες ή μεθόδους - τις "χωρίς απώλειες" ή "με απώλειες".

Οι τεχνικές συμπίεσης χωρίς απώλειες μπορούν να αναπαραχθούν πλήρως και είναι πρωταρχικά περιορισμένες σε λειτουργίες δεδομένων. Σε τελευταία ανάλυση, αφού τα δεδομένα μπορούν να απαρτίζονται από αποδεκτούς λογαριασμούς, μισθολόγια και απαιτήσεις υγειονομικής ασφάλισης, είναι εξαιρετικά

σημαντικό, από τη στιγμή που συμπιέζονται τέτοια δεδομένα, η αποσυμπίεσή τους να έχει ως αποτέλεσμα την ακριβή αναδόμηση των αρχικών δεδομένων. Άλλοι κοινοί όροι που χρησιμοποιούνται για την αναφορά στη συμπίεση χωρίς απώλειες περιλαμβάνουν την "ανακτήσιμη" και "μη-καταστρεπτική" συμπίεση.

Οι τεχνικές συμπίεσης με απώλειες μπορεί ή και δεν μπορεί να αναπαραχθούν πλήρως και είναι πρωταρχικά περιορισμένες σε λειτουργίες που αφορούν εικόνες, βίντεο και ήχο. Παρόλο που το αποτέλεσμα της αποσυμπίεσης ίσως να μην παρέχει ένα ακριβές αντίγραφο των αρχικών δεδομένων, οι διαφορές μεταξύ των αρχικών και των αναδομημένων δεδομένων ίσως να είναι τόσο ελάχιστες ώστε να είναι δύσκολο να ειδωθούν ή να ακουστούν. Από τη στιγμή που ο λόγος συμπίεσης που εξάγεται με τη χρήση της συμπίεσης με απώλειες μπορεί να υπερβεί κατά μεγάλο βαθμό το λόγο συμπίεσης που εξάγεται από τη συμπίεση χωρίς απώλειες, η πρωταρχική ανταλλαγή αφορά την ανάγκη κάποιου για δυνατότητα αναπαραγωγής ενάντια στις απαιτήσεις του για αποθήκευση και μετάδοση.

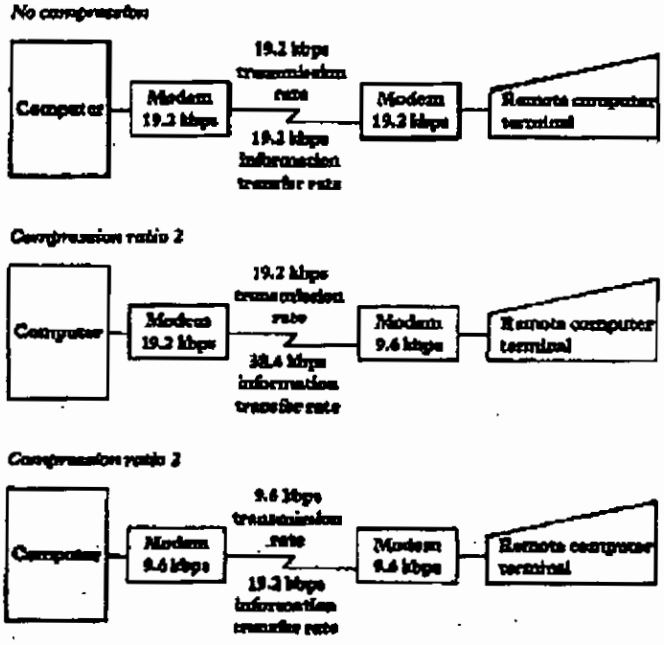
1.5. Επικοινωνιακές εφαρμογές

Για την εξαγωγή μιας σύνοψης κάποιων από τα επικοινωνία πλεονεκτήματα που διατίθενται μέσα από την ενσωμάτωση της συμπίεσης δεδομένων, μπορούμε να θεωρήσουμε μια τυπική εφαρμογή επικοινωνιών δεδομένων. Όπως φαίνεται στο επάνω τμήμα του Σχήματος 1.2, ένας απομακρυσμένος υπολογιστής ή τερματικό είναι

συνδεμένος με έναν κεντρικό υπολογιστή με μετάδοση που εμφανίζεται σε ένα λόγο δεδομένων των 19.2 kbps. Ας υποθέσουμε ότι τα προς μεταφορά δεδομένα δεν έχουν συμπιεστεί. Αν μέσω του προγραμματισμού ενός ή περισσότερων αλγορίθμων συμπίεσης ή την εγκατάσταση μιας διάταξης συμπίεσης hardware, εξάγεται ένας λόγος συμπίεσης 2, διάφορες εναλλακτικές λύσεις μπορούν να είναι διαθέσιμες, ανάλογα με τη μεθοδολογία επικοινωνιών δεδομένων. Πρώτα, ο χρόνος μετάδοσης δεδομένων μειώνεται, αφού ο αποτελεσματικός λόγος μεταφοράς πληροφόρησης έχει αυξηθεί σχεδόν κατά 38.4 kbps όπως φαίνεται στο μέσο τμήμα του Σχήματος 1.2. Αγνοώντας το λογισμικό επικοινωνιών, ο χρόνος μετάδοσης δεδομένων μειώνεται στο μισό. Έτσι, μπορείτε ίσως να κάνετε χρήση του απομακρυσμένου υπολογιστή ή τερματικού για άλλες απομακρυσμένες εφαρμογές επεξεργασίας ή ίσως μια ακριβή βάρδια υπερωριών ή ένα τμήμα μιας τέτοιας βάρδιας μπορούν να αποφευχθούν.

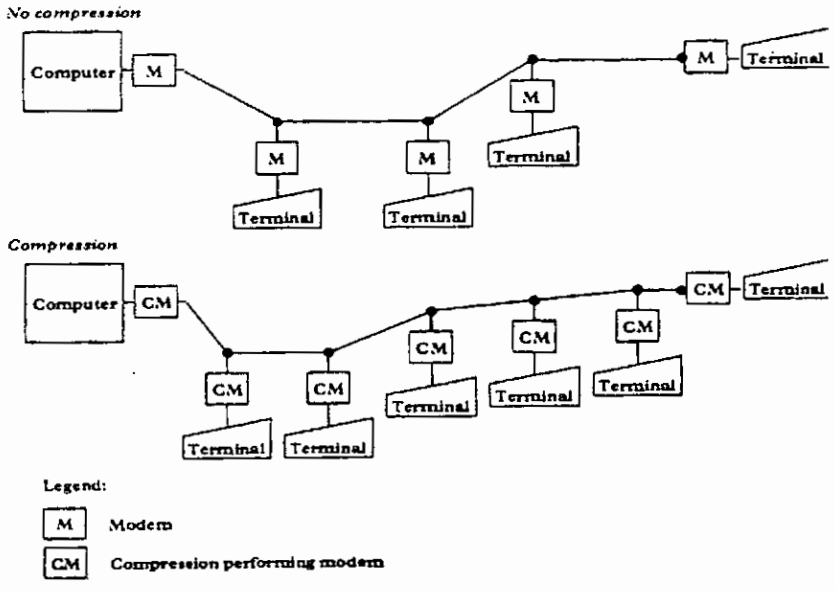
Στο χαμηλότερο τμήμα του Σχήματος 1.2 φαίνεται άλλη μια εναλλακτική λύση χρήστη. Εδώ ο λόγος μετάδοσης μπορεί να μειωθεί στα 9600 bps. Με ένα λόγο συμπίεσης 2, αυτός είναι ισοδύναμος σε ένα λόγο μεταφοράς πληροφοριών των 19.200 bps. Χαμηλώνοντας το λόγο μετάδοσης δεδομένων, τα ακριβά μόντεμ των 19.200 bps μπορούν να αντικατασταθούν με άλλα φθηνότερα των 9.600 bps και μπορεί να μην χρειάζεται η γραμμική κατάσταση που συνήθως απαιτείται όταν μεταφέρονται δεδομένα των 19.200 bps, με αποτέλεσμα μια συμπληρωματική μείωση κόστους.

Ένας δεύτερος τύπος εφαρμογής επικοινωνιών που μπορεί να ωφεληθεί από την εφαρμογή της συμπίεσης δεδομένων φαίνεται στο



Σχήμα 1.2. Η συμπίεση δεδομένων επηρεάζει το λόγο μεταφοράς πληροφοριών (information transfer ratio - ITR). Μέσω της χρήσης της συμπίεσης δεδομένων, η μεθοδολογία και δομή μιας εγκατάστασης επικοινωνίας μπορεί να αλλάξει.

Σχήμα 1.3. Ένα τυπικό πολύ-ελλειμματικό (multidrop) δίκτυο φαίνεται στο επάνω τμήμα του Σχήματος 1.3, που συνδέει τερματικά σε διάφορες γεωγραφικές τοποθεσίες μέσω μιας κοινής ενοικιασμένης γραμμής σε μια τοποθεσία υπολογιστή. Τυπικά, η δραστηριότητα μετάδοσης των τερματικών είναι ο βασικός παράγοντας που περιορίζει την πολύ-ελλειμματική γραμμή σε ένα μέγιστο αριθμό ελλείψεων (διακοπών). Στο κάτω τμήμα του Σχήματος 1.3, υποθέτεται ότι τα μόντεμ που εκτελούν τη συμπίεση αντικαταστάθηκαν για τα συμβατικά μόντεμ που χρησιμοποιούνται στον αρχικό πολύ - ελλειμματικό



Σχήμα 1.3. Συμπίεση δεδομένων σε μια πολύ-ελλειμματική (multidrop) γραμμή μειώνει τη ροή δεδομένων στη γραμμή, επιτρέποντας την εξυπηρέτηση συμπληρωματικών τερματικών.

σχηματισμό. Από τη στιγμή που η συμπίεση δεδομένων σε μια πολύ-ελλειμματική γραμμή μειώνει τη ροή δεδομένων της γραμμής, η χρησιμοποίησή του θα επιτρέπει συνήθως να προστίθενται συμπληρωματικές ελλείψεις στη γραμμή πριν την εμφάνιση των καθυστερήσεων που αποδίδει ο υπολογιστής που επηρεάζουν το χρόνο απόκρισης των τερματικών που είναι προσαρμοσμένα σε κάθε έλλειψη. Σ' αυτό το συγκεκριμένο παράδειγμα, υποτίθεται ότι η χρήση των μόντεμ που εκτελούν τη συμπίεση επιτρέπει μια αύξηση στον αριθμό των ελλείψεων γραμμών από το 4 στο 6.

Για μόντεμ διακοπτόμενου δικτύου, η συμπίεση δεδομένων παρέχει τη δυνατότητα να εξαχθεί μια ικανότητα μεταφοράς πληροφοριών σ' ένα κλάσμα του κόστους των μόντεμ υψηλότερης ταχύτητας. Αυτό οφείλεται στα πολύπλοκα σχήματα διαμόρφωσης που χρησιμοποιούνται από τα μόντεμ που λειτουργούν στα 19.200 bps σε σύγκριση με λιγότερο πολύπλοκα σχήματα που χρησιμοποιούνται από μόντεμ των 9.600 bps. Ενσωματώνοντας έναν αλγόριθμο συμπίεσης σε ένα μόντεμ χαμηλότερου λειτουργικού λόγου, είναι δυνατό να επιτευχθεί μια απόδοση δεδομένων μεταξύ των 9.600 και 19.200 bps για το κόστος ενός τσιπ μνήμης μόνο ανάγνωσης (read only memory - ROM) που μπορεί να κοστίζει στο χονδρεμπόριο πολύ φθηνά. Συγκριτικά, το αναλογικό σχέδιο κυκλώματος που απαιτείται για να λειτουργήσει ένα μόντεμ στα 19.200 bps μπορεί να έχει ένα κόστος μεγάλο στο χονδρεμπόριο ή και περισσότερο από το σχέδιο κυκλώματος που απαιτείται για ένα μόντεμ των 9.600 bps. Από τη στιγμή που οι περισσότεροι κατασκευαστές πρέπει να πωλούν τα προϊόντα τους σε διπλάσιες ή τριπλάσιες τιμές από το κόστος τους, η χρήση συμπίεσης σε μόντεμ χαμηλότερης ταχύτητας μπορεί να μεταφραστεί σε σημαντικές εξοικονομήσεις κόστους στη λιανική σε σύγκριση με το κόστος των μόντεμ που λειτουργούν σε υψηλότερους λόγους δεδομένων. Επιπρόσθετα, η ενσωμάτωση της συμπίεσης δεδομένων σε μόντεμ διακοπτόμενου δικτύου υψηλής ταχύτητας παρέχει στους χρήστες μια ικανότητα μετάδοσης πληροφοριών πέρα από την ικανότητα που παρέχει η παρούσα τεχνολογία των μόντεμ.

1.6. Άλλες εφαρμογές

Άλλες εφαρμογές για τη συμπίεση περιλαμβάνουν εκτεταμένες ηχητικές μαγνητοφωνήσεις αποθηκευμένες σε CD-ROM, μεταδόσεις fax, βίντεο και την εφαρμογή των πολυμέσων (multimedia) που χρησιμοποιούνται για την παρουσίαση συνενωμένων ,κειμένου, εικόνων, ήχου και βίντεο.

Παραδείγματα ηχογράφησης ήχου περιλαμβάνουν τους μεταφορείς επικοινωνιών που χρησιμοποιούν μια τεχνική που αναφέρεται ως Προσαρμόσιμη Διαμόρφωση Παλμικού Κώδικα (Adaptive Pulse Code Modulation - ADPCM) αντί για την PCM για την κωδικοποίηση ψηφιοποιημένων φωνητικών συζητήσεων σε διεθνή κυκλώματα. Με τον τρόπο αυτό μειώνεται ο λόγος δεδομένων του PCM που απαιτεί 64 kpbs στα 32 kbps κάτω από την ADPCM. Άλλες τεχνικές συμπίεσης ήχου μειώνουν το λόγο δεδομένων που απαιτείται για τη μετάδοση μιας ψηφιοποιημένης φωνητικής συζήτησης σχεδόν μέχρι τα 2.4 kbps ωστόσο, η αναπαράξιμη φωνητική ποιότητα ελαττώνεται καθώς μειώνεται ο λόγος δεδομένων.

Παρόλο που ο περισσότερο δημοφιλής τύπος αποθήκευσης ήχου είναι ο Συμπαγής Δίσκος (Compact Disc - CD), οι ηχογραφήσεις σε CD δεν είναι συμπιεσμένες και περιορίζονται περίπου στη μια ώρα. Ο λόγος γι' αυτό είναι το γεγονός ότι η ανάπτυξη των μηχανημάτων που παίζουν CD (CD player) χρονολογείται από το 1970, χρόνος κατά τον οποίο η συμπίεση απαιτούσε εκτεταμένα σχέδια κυκλωμάτων. Ως αποτέλεσμα του γεγονότος αυτού, το CD player σχεδιάστηκε χωρίς συμπίεση. Συγκριτικά, οι εφαρμογές πολυμέσων που αποθηκεύονται σε CD-ROM σχεδιάστηκαν να εργάζονται με υπολογιστές ή λογισμικό για

προσωπικούς υπολογιστές, που εκτελούν συμπίεση. Με τον τρόπο αυτό γίνεται δυνατή η αποθήκευση εικόνων, βίντεο και ήχου σε συμπιεσμένη μορφή.

Όσον αφορά το βίντεο, τις ταινίες καθώς επίσης και τις τηλε-συνεδριάσεις, μπορούν να θεωρηθούν ως αλληλουχία σταθερών εικόνων επαναλαμβανόμενων κατά πολλά πλαίσια ανά δευτερόλεπτο. Με την παρατήρηση επανάληψης μεταξύ των πλαισίων και την εξάλειψη της επανάληψης αυτής, μπορεί κάποιος να μειώσει την αποθήκευση και μετάδοση του βίντεο.

Σήμερα υπάρχουν στην κυριολεξία εκατοντάδες τεχνικών συμπίεσης. Μερικές τεχνικές, όπως η συμπίεση εκτέλεσης μήκους (run length), εφαρμόζονται σε ήχο, δεδομένα, εικόνες και βίντεο. 'Αλλες τεχνικές, όπως η JPEG, είναι εφαρμόσιμες μόνο σε εικόνες. Δυστυχώς, περιορισμοί στο μέγεθος της εργασίας,απαιτούν η πρωταρχική εστίασή μας να βρίσκεται στις τεχνικές και εργαλεία πάνω στη συμπίεση δεδομένων και εικόνας.

2. Τεχνικές συμπίεσης που βασίζονται στο χαρακτήρα

Η τεράστια ανάπτυξη των απομακρυσμένων υπολογισμών κατά τη διάρκεια της τελευταίας δεκαετίας έχει επικεντρώσει το ενδιαφέρον του προσωπικού επικοινωνιών πάνω στις τεχνικές συμπίεσης δεδομένων. Η συμπίεση, που αρχικά προξένησε την προσοχή των χρηστών επεξεργασίας δεδομένων στη δεκαετία του '60, ως μηχανισμός για την αύξηση της ικανότητας μαζικής αποθήκευσης των συσκευών, σήμερα εφαρμόζεται γενικά στο πεδίο των επικοινωνιών . Εδώ, η συμπίεση έχει ως αποτέλεσμα τη μεταφορά των δεδομένων σε πιο μικρές χρονικές περιόδους απ' ότι εάν τέτοια δεδομένα μεταφέρονταν χωρίς την εφαρμογή μιας τεχνικής συμπίεσης.

Στο κεφάλαιο αυτό, καλύπτονται κάποιες ευκρινείς μέθοδοι που μπορούν να εφαρμοστούν για την συμπίεση δεδομένων. Καθεμιά από τις μεθόδους που περιγράφονται στο κεφάλαιο αυτό εκτελείται βάση της χρήσης ενός ειδικού χαρακτήρα ενδείκτη συμπίεσης και μπορεί να ταξινομηθεί ως τεχνική συμπίεσης δεδομένων βάσει χαρακτήρα. Παρόλο που καθεμιά από τις τεχνικές μπορεί να εφαρμοστεί και σε ανεξάρτητη βάση, από πρακτικής άποψης είναι πολύ κοινό να εφαρμόζονται δυο ή περισσότερες τεχνικές συμπίεσης που βασίζονται σε χαρακτήρα, αφού τα προς συμπίεση δεδομένα συνήθως θα είναι περισσότερο επιδεκτικά σε μια μίξη των μεθόδων συμπίεσης.

Η βάσει χαρακτήρων συμπίεση, ενώ προσφέρει το δυνητικό για σημαντική μείωση αποθήκευσης δεδομένων ή απαιτήσεων μετάδοσης, δεν θα πρέπει να θεωρείται ως αυτοσκοπός. Σε πολλές περιπτώσεις, αυτή μπορεί να είναι το πρώτο επίπεδο ενός πολυεπίπεδου σχήματος συμπίεσης, με άλλα επίπεδα να εφαρμόζουν μια μέθοδο συμπίεσης στατιστικής βάσης η οποία κωδικοποιεί τα δεδομένα βάσει της συχνότητας εμφάνισης όλων των χαρακτήρων, περιλαμβανομένων των ειδικών χαρακτήρων ένδειξης συμπίεσης.

Επιπρόσθετα με την εξέταση των μεθόδων συμπίεσης που βασίζονται στους χαρακτήρες, διάφοροι συνδυασμοί τεχνικών συζητούνται, με έμφαση πάνω στη χρήση και αποτελεσματικότητά τους. Μερικές από τις τεχνικές που καλύπτονται στο κεφάλαιο αυτό, για να είναι αποτελεσματικές, απαιτούν μια προσεκτική ανάλυση της παρούσας ή μέλλουσας κίνησης δεδομένων. Καμία από τις τεχνικές που παρουσιάζονται δεν απαιτεί περισσότερο από ένα μέσο επίπεδο δυσκολίας στην ανάπτυξη λογισμικού για να εξαχθούν οι αλγόριθμοι κωδικοποίησης και αποκωδικοποίησης. Στην πραγματικότητα, οι περισσότερες από τις τεχνικές του θα μπορούσαν εύκολα να εφαρμοστούν από τους τελικούς χρήστες και η εφαρμογή τους ίσως έχει ως αποτέλεσμα έναν υψηλό βαθμό μείωσης δεδομένων για ένα ελάχιστο μέγεθος προσπάθειας.

Με την εφαρμογή μιας ή περισσοτέρων τεχνικών συμπίεσης βάσει χαρακτήρα, οι λειτουργικές ικανότητες μπορεί να αυξηθούν ή να μειωθούν τα κόστη μετάδοσης. Για την πρώτη περίπτωση, η συμπίεση δεδομένων θα επιτρέψει μια αύξηση στις πληροφορίες που μεταφέρονται μέσο ενός συνδέσμου δεδομένων ανά διάστημα μονάδας χρόνου. Όσον αφορά τη δεύτερη περίπτωση, η μείωση του

ποσού των δεδομένων που πρόκειται να μεταφερθούν φυσικά ίσως κάνει επιτρεπτή την εφαρμογή ενός συνδέσμου δεδομένων χαμηλότερης ταχύτητας, με αποτέλεσμα μια μείωση στο κόστος συγκριτικά με τα έξοδα ενός συνδέσμου δεδομένων που λειτουργεί σε έναν υψηλότερο λόγο δεδομένων.

2.1. Καταστολή κενών

Η καταστολή κενών ήταν μια από τις πρωιμότερες τεχνικές συμπίεσης δεδομένων που εφαρμόστηκαν. Σήμερα, αυτή η απλοϊκή τεχνική εφαρμόζεται στο πρωτόκολλο μετάδοσης IBM 3780 BISNYC. Επιπρόσθετα, η καταστολή κενών είναι ευρέως χρησιμοποιούμενη με μια μίξη άλλων τεχνικών συμπίεσης προσανατολισμού χαρακτήρα για τη μείωση της αποθήκευσης δεδομένων και της μεταφοράς τους.

2.1.1. Τεχνική ανασκόπηση

Όπως δηλώνει και το όνομα, η καταστολή κενών είναι μια τεχνική συμπίεσης δεδομένων που ανιχνεύει ,ένα ρεύμα δεδομένων για επαναλαμβανόμενα κενά. Με το που συναντά μια τέτοια. αλληλουχία οι χαρακτήρες κενού αντικαθίσταται από ένα ειδικά διαταγμένο ζεύγος χαρακτήρων, των οποίων η μορφοποίηση φαίνεται στο σχήμα

2.1. πρώτα, ένας χαρακτήρας ένδειξης συμπίεσης εφαρμόζεται για να δηλώσει ότι έχει εμφανιστεί η καταστολή κενών. Ο δεύτερος χαρακτήρας χρησιμοποιείται για να δείξει την ποσότητα των χαρακτήρων κενών που συναντούνται και αντικαθίστανται από την αλληλουχία των δυο χαρακτήρων (Aronson, 1977-Ruth and Kreutzer, 1972).

Όταν η αλληλουχία των δύο χαρακτήρων μεταδίδεται μέσα σ' ένα ρεύμα δεδομένων, η συσκευή υποδοχής αρχίζει να ερευνά για τον ειδικό χαρακτήρα που χρησιμοποιήθηκε για να δείξει την καταστολή κενού. Με την ανίχνευση του χαρακτήρα αυτού ο δείκτης γνωρίζει ότι ο επόμενος χαρακτήρας περιέχει τον αριθμό των κενών που συμπιέστηκαν. Από την πληροφορία αυτή μπορεί να αναδομηθεί το αρχικό ρεύμα δεδομένων.

Στο μεσαίο τμήμα του Σχήματος 2.1 είναι ένα παράδειγμα της εφαρμογής της καταστολής κενού σε ένα ρεύμα δεδομένων. Εδώ ο χαρακτήρας S_C δείχνει έναν ειδικό χαρακτήρα ένδειξης συμπίεσης που δηλώνει ότι έχει συμβεί καταστολή κενού.

Στο χαμηλότερο τμήμα του σχήματος 2.1, φαίνεται ένα διάγραμμα ροής της διαδικασίας ανίχνευσης καταστολής κενού. Αν υποθέσουμε μια μορφοποίηση των 8 bit για τους χαρακτήρες δεδομένων τότε ο μετρητής χαρακτήρων μπορεί να αποθηκεύσει τιμές μέχρι το 255 διαδοχικά συναντώμενων κενών πριν την υπερφόρτωση αν αρχίσουμε την αρίθμησή μας από το 1, ή 256 αν η αρίθμησή μας αρχίζει με την υπόθεση ότι ένας μετρητής μηδέν παριστάνει μια τιμή του 1.

διαφορετικών τύπων συμπίεσης και αποσυμπίεσης, αναπτύχθηκε μια σειρά κοινών (συνηθισμένων) ή σχετικά κοινών λειτουργιών. Αυτές οι κοινές ή σχετικά κοινές λειτουργίες περιλαμβάνουν εντολές χειρισμού, εντολές συμπίεσης και αποσυμπίεσης, εντολές για την απαρίθμηση της συμπίεσης ή του αριθμού των εισαγόμενων χαρακτήρων, και εντολές που δείχνουν τα περιεχόμενα του φακέλου πριν και μετά το πρόγραμμα. Οι εντολές στην τελευταία λειτουργία επίσης παρέχουν μια περίληψη των χαρακτήρων που αφαιρέθηκαν ή εισήχθησαν βάσει του τύπου της λειτουργίας που εκτελέστηκε.

Το Σχήμα 2.2 δίνει τον κώδικα της γλώσσας C για το πρόγραμμα NULLC.C . Παρόλο που θα εξετάσουμε ολόκληρο το πρόγραμμα που περιέχεται στο Σχήμα 2.2, για επακόλουθα προγράμματα θα επικεντρώσουμε την προσοχή μας πρωταρχικά στις κυριότερες λειτουργίες κάθε προγράμματος και απλά θα αναφέρουμε τη λειτουργία κάποιων χειρισμών που αναφέρθηκαν προηγουμένως. Στην εξέταση του κώδικα πριν την «κύρια» λειτουργία, σημειώστε ότι οι ερμηνευμένοι ορισμοί χρησιμοποιούνται για τη συσχέτιση των προτεινόμενων ονομάτων των φακέλων I/0 σε ορισμούς στις λειτουργίες 'filein', και 'summary' που παράγουν μηνύματα σχετικά με φακέλους. Για παράδειγμα, ο ορισμός 'printf' ("Enter ηοη-compressed ASCII Filename [εισάγετε μη συμπιεσμένο ASCII όνομα αρχείου] Eg, %s:".infile);" έχει ως αποτέλεσμα την επίδειξη του μηνύματος 'Enter ηοη-compressed ASCII Filename. Eg NULL.DAT' αφού ο ορισμός DEFINE [καθόρισε] σχετίζει την αλληλουχία 'NULL.DAT' με τη μεταβλητή 'infile'. Η χρήση των ορισμών DEFINE επιτρέπει στις λειτουργίες 'filein' και 'summary' να χρησιμοποιηθούν είτε ως 'as is' [όπως είναι], ή με ελαφρές μετατροπές μέσα στο κεφάλαιο αυτό.

Οι διατάξεις 0 και προσωρινού καταχωρητή (bufter) έχουν τεθεί και οι δυο στα 256, καθώς το μέγεθος διατάξεως αυτό μπορεί να αποθηκεύσει το μεγαλύτερο μήκος γραμμής που υποστηρίζεται από το DOS. Η διάταξη 0 θα χρησιμοποιηθεί ως ο καταχωρητής εξόδου, ενώ η διάταξη καταχωρητή θα χρησιμοποιηθεί ως καταχωρητής εισόδου. Μετά τη χρήση των ορισμών DEFINE και την κατανομή των διατάξεων, κάθε πρόγραμμα στο κεφάλαιο αυτό θα τοποθετήσει αρχικές τιμές σε κατάλληλους μετρητές. Κατόπιν, η λειτουργία 'main' θα καλέσει τη λειτουργία filein, που θα υποκινήσει το χρήστη να εισάγει τον κατάλληλο φάκελο ASCII. Ο βρόγχος 'while' κυκλώνεται γύρω από κάθε γραμμή στο φάκελο, θέτοντας το η στο μήκος της πρόσφατης (παρούσας) γραμμής ανάγνώσης. Αυτό ακολουθείται από την εκτέλεση των λειτουργιών 'nullit' η οποία εκτελεί την πραγματική συμπίεση και 'tally' η οποία κρατά ένα τρέχον σύνολο χαρακτήρων αφαιρουμένων εξαιτίας της συμπίεσης κενού. Από τη στιγμή που επεξεργάζεται ο φάκελος εισόδου, η λειτουργία 'summary' δείχνει τα περιεχόμενα 'before and after' [πριν και μετά] του φακέλου που μόλις λειτούργησε και μια περίληψη των χαρακτήρων που αφαιρέθηκαν. Στην ανάπτυξη κάθε προγράμματος συμπίεσης - αποσυμπίεσης, συγγραφέας περιέλαβε ένα πλήθος σχολίων για να τεκμηριώσει τις πράξεις κάθε προγράμματος. Οι περισσότερες εκδοχές αποσυμπίεσης του προγράμματος συμπίεσης θα περιέχουν λιγότερα σχόλια, που κυρίως χρησιμοποιούνται για την καταγραφή των πραγματικών ενεργειών αποσυμπίεσης. Παρόλο που το μεγαλύτερο τμήμα της κωδικοποίησης είναι αυτοεπεξηγηματικό για άτομα με μια προίστορία στον προγραμματισμό της C, μερικές εντολές απαιτούν κάποιο βαθμό επεξεργασίας. Πρώτα, η εντολή (IF στη λειτουργία NULLIT τεστάρει για

την παρούσα και επόμενη θέση στον προσωρινό καταχωρητή εισόδου, εξισώνοντας το χαρακτήρα διαστήματος ο οποίος έχει μια τιμή ASCII 32. Έτσι, το πρόγραμμα στην πραγματικότητα εξαφανίζει τους χαρακτήρες διαστημάτων αντί για τα κενά, τα οποία έχουν μια τιμή ASCII 127. Αυτό έγινε για σκοπούς επίδειξης, αφού είναι ευκολότερο να παραχθούν κενά από το πληκτρολόγιο. Δεύτερο, η εντολή o[j]=125 έχει ως αποτέλεσμα τη χρήση της δεξιας αγκύλης } ως το χαρακτήρα ένδειξης καταστολής κενού.

Εξετάζοντας την καταλογοποίηση του προγράμματος για τους αναγνώστες των NULLC.C και NULLD.C, θα πρέπει να σημειωθεί ότι τα προγράμματα δεν ελέγχουν για τη φυσική εμφάνιση μιας δεξιάς αγκύλης στο ρεύμα δεδομένων. Έτσι, οι αναγνώστες ίσως θελήσουν να σκεφτούν τη μετατροπή του κώδικα για να εισάγουν μια δεξιά αγκύλη αν ο χαρακτήρας αυτός συναντάται στο ρεύμα δεδομένων όταν εκτελείται η συμπίεση κενού. Όταν εκτελείται η αποσυμπίεση κενού, θα πρέπει να μετατρέψετε το πρόγραμμα NULLD.C για να ελέγξει για την ύπαρξη δυο δεξιών αγκυλών σε σειρά, πράγμα που θα δήλωνε ότι μια αγκύλη έκανε φυσική εμφάνιση στα δεδομένα. Παρόμοια, αφού η τιμή ASCII της δεξιάς αγκύλης είναι 125, δεν θα πρέπει να συμπιέσετε την εμφάνιση ακριβώς 125 κενών, καθώς η πράξη αυτή θα είχε ως αποτέλεσμα την αντικατάσταση της σειράς αυτής από δυο δεξιές αγκύλες. Αντίθετα, θα πρέπει να σπάσετε την εμφάνιση μιας δέσμης 125 κενών σε δυο δέσμες έτσι ώστε να μπορέσετε να συνεχίσετε για χρησιμοποιείτε την τεχνική insert/delete [είσοδος/σβήσιμο] για να εμποδίσετε το ενάντιο αποτέλεσμα ενός χαρακτήρα δείκτη συμπίεσης που εμφανίζεται φυσικά στο φάκελο δεδομένων.

2.1.3. Εκτέλεση του προγράμματος

Η εκτέλεση του προγράμματος NULLC.C φαίνεται στο σχήμα 2.3. Στο επάνω μέρος του σχήματος το πρόγραμμα δίνει λίστα των περιεχομένων του φακέλου δοκιμής NULL.DAT που αναπτύχθηκε για επεξηγηματικούς σκοπούς. Το χαμηλότερο τμήμα του Σχήματος 2.3.

δείχνει τον κατάλογο των περιεχομένων του συμπιεσμένου φακέλου. Αφού τιμές ASCII μικρότερες από 32 δεν τυπώνονται, δεν μπορείς να πληροφορηθείς το μετρημένο χαρακτήρα που φαίνεται μετά τη δεξιά αγκύλη στη γραμμή 4. Παρόλα αυτά, θα σημειώσεις ότι η γραμμή 3 έχει 49 διαστήματα. Αφού το ASCII 49 είναι το ψηφίο 1 , η γραμμή 3 στην εκτύπωση του συμπιεσμένου φακέλου επαληθεύει ότι 49 διαστήματα αντικαταστάθηκαν από τη δεξιά αγκύλη ακολουθούμενη από το αριθμητικό 1

2.1.4. Πρόγραμμα αποσυμπίεσης

Ακολούθως από τις ονομασίες των συνθηκών μας το πρόγραμμα NULLD.C παρουσιάζει το πρόγραμμα γλώσσας C που αναπτύχθηκε για την αποσυμπίεση των φακέλων που συμπιέστηκαν προηγουμένως με τη χρήση του προγράμματος NULLC.C. Στην

παράγραφο αυτή θα εστιάσουμε την προσοχή μας πάνω στη λειτουργία αποσυμπίεσης και θα σημειώσουμε κάποιες μικρές αλλαγές μεταξύ του προγράμματος αυτού και του προηγούμενα περιγραφόμενου προγράμματος NULLC.C στην επικεφαλίδα του προγράμματος και σε μερικές λειτουργίες.

Αφού ο φάκελος εισόδου για το NULLD.C είναι τώρα ο NULLC.DAT, ενώ ο φάκελος εξόδου γίνεται NULLD.DAT, οι δυο εντολές DEFINE άλλαξαν για να αντανακλούν τους φακέλους αυτούς. Δηλαδή, τα #define infile "NULLC.DAT" και #define outfile "NULLD.DAT" χρησιμοποιούνται τώρα για να προσδιορίσουν τις αλληλουχίες NULLC.DAT και NULLD.DAT στις μεταβλητές infile και outfile για χρήση στο πρόγραμμα. Το σχήμα 2.4 δίνει ένα κατάλογο του κώδικα για τις λειτουργίες 'main' , 'nullit' και 'tally' για το πρόγραμμα NULLD.C. Αφού ο NULLC.C αναπτύχθηκε για να συμπιέσει αλληλουχίες (σειρές) διαστημάτων, ο NULLD.C αναπτύχθηκε για την αποκατάσταση των διαστημάτων από τη στιγμή που συναντάται ο χαρακτήρας που δείχνει συμπίεση. Έτσι, η εντολή IF [BUFFER[I]==125] ελέγχει για την δεξιά αγκύλη που δείχνει τη συμπίεση. Όταν συναντηθεί, η εντολή o[j++]=""; θέτει την j-στή θέση του προσωρινού καταχωρητή εξόδου σ' ένα διάστημα και αυξάνει τον αριθμοδείκτη j κατά ένα.

```
/* NULLC.C - Program to demonstrate null surpression (actually ASCII 32     */

#include <stdio.h>
#define infile "NULL.DAT"      /* define name of I/O files suggested for     */
#define outfile "NULLC.DAT"    /* use in this program                        */
      char  o[256],            /* output buffer                              */
      filename[13],            /* store filename                             */
      buffer[256];             /* input buffer - max string length           */

int   t=0,                     /*                                            */
      t1=0,
      n=0,
      n1=0,

FILE  *input,
      *output;

main()
{
   filein();
   while( fgets(buffer,256,input) != NULL )
   {
      n = strlen(buffer);                  /* get length of string          */
      tally( nullit() );                   /* check for nulls, & tally      */
   }
   summary();
   return 0;
}

nullit()
{
   int   i=0                               /* set indices                   */
         j=0,                              /* buffer position               */
         k=1;                              /* null count                    */

   unsigned char   a,
                   b;

   for(i=0;i < n; i++)                     /*step through record            */

      a = buffer[i];                       /* extract a character           */
      if ((a ==' ') && (a == buffer[i+1]))/* two nulls in a row             */

         b=a;                              /* set next = current            */
         k++;                              /* increment count               */
         continue;

      if  ( k> 2)                          /* have null suppression         */

         o[j] = 125;                       /* set flag                      */
         o[j+1] = k;                       /* insert null count             */
         j+=2;                             /* increment buffer position     */
         k = 1;                            /* reset count                   */

      else
      {
         if ( k== 2)                       /* only two nulls in sequence    */
         {
            o[j]= b;
            o[j+1] = b;
            j+=2;
            k=1;
         }
         else
         {
          o[j] = a;
          j++;
```

```
            }
        }
    }
    return(i);
}

tally(j)                                    /* tally the compression count   *
int   j;
{
    int    i=0;

    nl=nl+n;
    t=n-j;
    tl=tl+t;
    for(i=0; i<j; i++)
    {
       fputc(o[i],output);
    }
    return 0;
}

filein()
{
    printf("Enter non-compressed ASCII Filename.Eg, %s  :", infile);
    scanf("%13s",filename);

    if ( (input  = fopen(filename,"rb")) == NULL )
    {
       printf("ERROR: Cannot open the file: %s \n",filename);
       exit(-1);
    }
    if ( (output = fopen(outfile,"wb")) == NULL )
    {
       printf("ERROR: Cannot open the file %s for output\n",outfile);
       exit(-2);
    }
    return 0;


summary()
{
    fclose(input);
    fclose(output);
    if ( (input  = fopen(filename,"RB")) == NULL )
    {
       printf("ERROR: Cannot open the file %s for input\n",filename);
       exit(-1);
    }
    printf("File  %s Before Compression:\n",filename);

    while( fgets(buffer,256,input) != NULL )
    {
       printf("%s",buffer);
    }
    fclose(input);
    if ( (input = fopen(outfile,"rb")) == NULL )
    {
       printf("ERROR: Cannot open the file %s for output\n",outfile);
       exit(-2);
    }
    printf("\fFile %s After Compression :\n", outfile);

    while( fgets(buffer,256,input) != NULL )
    {
       printf("%s",buffer);
    }
    fclose(input);
    printf("\n%d Total Characters Eliminated From %d Or %d%%\n\n",
    tl,nl,(int)((float)((float)tl/(float)nl)*100));
    return 0;
}
```

Σχήμα 2.2 Κατάλογος προγράμματος NULLC.C

```
C:\TC>nullc
Enter non-compressed ASCII Filename.Eg,      NULL.DAT :null.dat
File  null.dat Before Compression:
1)this is a test
2)01234567890123456789012345678901234567890
3)*                                         *
4)* of null suppression                only      *

File NULLC.DAT After Compression :
1)this is a test
2)01234567890123456789012345678901234567890
3)*)1*
4)* of null suppression)only)*

68 Total Characters Eliminated From 183 Or 37%

c:\TC>
```

Σχήμα 2.3 Εκτέλεση του προγράμματος NULLC.C

2.1.5. Εκτέλεση του προγράμματος

Η εκτέλεση του NULLD.C διαβάζει τον προηγούμενα συμπιεσμένο φάκελο, δημιουργεί έναν αποσυμπιεσμένο φάκελο και δείχνει τα περιεχόμενα του συμπιεσμένου και του αποσυμπιεσμένου φακέλου. Το σχήμα 2.5 δείχνει την εκτέλεση του εκτελεστού φακέλου προγράμματος NULLD που δημιουργήθηκε με τη χρήση του μεταγλωττιστή Borland Turbo C.

Στην εξέταση του αποσυμπιεσμένου φακέλου που φαίνεται στο χαμηλότερο τμήμα του Σχήματος 2.5 θα παρατηρήσετε ότι ταιριάζει με το επάνω τμήμα του Σχήματος 2.3. Επιπρόσθετα, σημειώστε ότι οι χαρακτήρες που εισάγονται είναι ίσοι με τους χαρακτήρες που αφαιρέθηκαν προηγούμενα.

2.1.6. Περιορισμοί

Αφού μια αλληλουχία συμπίεσης δυο χαρακτήρων πάνω εξάγεται από την αποσυμπίεση μέχρι και 255 μετρημένων σε σειρά (αλληλουχία) κενών, δεν είναι δυνατές διασώσεις, εκτός αν βρεθούν τρία ή περισσότερα κενά σε σειρά. Έτσι, μια αλληλουχία δυο κενών δεν θα πρέπει να τοποθετηθεί στη συμπιεσμένη μορφή καταστολής κενού. Αυτό συμβαίνει επειδή δεν θα δημιουργηθούν διασώσεις, ενώ η διαδικασία συμπίεσης και αποσυμπίεσης απαιτεί ένα τμήμα χρόνου επεξεργασίας. Επιπρόσθετα, αν κάποιος εφαρμόζει διάφορες τεχνικές συμπίεσης δεδομένων, θα παρατηρήσουμε ότι δυο μηδενικά που μετρήθηκαν σε σειρά μπορούν να συμπιεσθούν αποτελεσματικά με τη διαδικασία κωδικοποίησης Diatomic. Αυτή η τεχνική συμπίεσης έχει ως αποτέλεσμα μια 100% μείωση δεδομένων για την κατάσταση αλληλουχίας δυο κενών, εκεί όπου η τεχνική καταστολής κενού είναι αναποτελεσματική.

Ένας κύριος περιορισμός που σχετίζεται με τη χρήση της καταστολής κενού αφορά την επιλογή του ειδικού χαρακτήρα ένδειξης συμπίεσης. Αν ένας ακαθόριστος χαρακτήρας διατίθεται από το σετ χαρακτήρων που χρησιμοποιείτε, κατόπιν οι διασώσεις αρχίζουν να αθροίζονται (αυξάνονται) όταν συναντώνται τρία ή περισσότερα συνεχόμενα κενά. Αν πρέπει να χρησιμοποιήσετε ένα εκτεταμένο σετ χαρακτήρων που καθορίζεται πρώτα από τον χαρακτήρα shift out (SO), θα χρειαστείτε

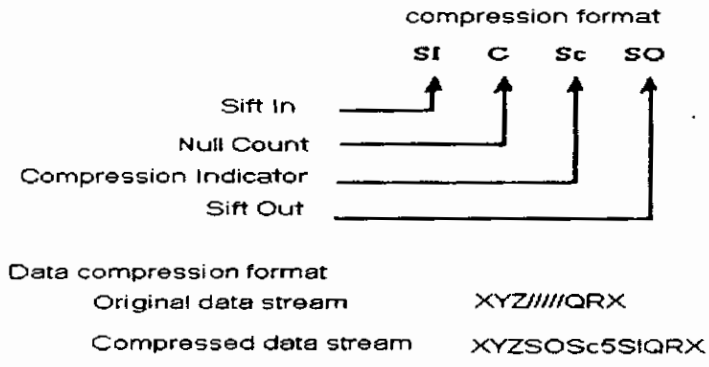
```
main()
{
   filein();
   while( fgets(buffer,256,input) != NULL )
   {
      n = strlen(buffer);                    /* get length of string
      tally( nullit() );                     /* check for nulls, & tally
   }
   summary();
   return 0;
}

nullit()
{
   int     i=0,                              /* set indices
           k=0,                              /* null count
           l=0,                              /* loop for output buffer
           j=0;                              /* buffer position

   for(i=0;8 <n; i++)                         * step through record
   {
      if (buffer[i] == 12                     * compression indicator ?
      {
          k = buffer[i+1];                    * get null count

          for (l=0; l<k; l                    * set nulls in output buffer
          {
            o[j++]=' ';                       * set to null
          }
           i++;                               * skip over null count
      }
      else
      {
         o[j++] = buffer[i];                  * just a non null characte

   }
   return;

tally(                                        * tally the compression coun
int
{
          i=0;

   n1=n1+n;
   t=n-j;
   t1=t1-t;
   for(i=0; i<j; i++)
   {
      fputc(o[i],output);
   }
   return 0;
}
```

Σχήμα 2.4 Κύριες λειτουργίες του προγράμματος NULLD.C

μια αλληλουχία τεσσάρων χαρακτήρων για την κωδικοποίηση ενός τρεξίματος (εκτέλεσης) μηδενικών. Το σχήμα 2.6 δείχνει τη μορφή συμπίεσης και ένα παράδειγμα καταστολής κενού με τη χρήση ενός

```
C:\TC>nulld
Enter compressed ASCII Filename.Eg,  NULLC.DAT  :nullc.dat
Contents of compressed file nullc.dat :
1)this is a test
2)01234567890123456789012345678901234567890123456789
3)*)1*
4)* of null suppression)only)*

Decompressed file contents :

1)this is a test
2)01234567890123456789012345678901234567890123456789
3)*                                                 *
4)* of null suppression                    only     *

68 Total Characters Inserted

C:\TC>
```

Σχήμα2.5. Εκτέλεση του προγράμματος NULLD.C



Σχήμα 2.6 Καταστολή κενού με τη χρήση ενός εκτεταμένου κώδικα

εκτεταμένου κώδικα.

Όπως παρατηρήθηκε από το παράδειγμα του σχήματος 2.6, η καταστολή κενού είναι εφαρμόσιμη μόνο για μια αλληλουχία πέντε ή περισσότερων κενών, όταν πρέπει να χρησιμοποιηθεί ένας εκτεταμένος κώδικας για το σχηματισμό ενός ειδικού χαρακτήρα που να δείχνει τη συμπίεση.

Για να ληφθεί υπόψη η τεχνική insert και delete για τη χρήση του χαρακτήρα που δείχνει τη συμπίεση, θα πρέπει να γνωρίζετε τη συχνότητα εμφάνισης του επιλεγμένου χαρακτήρα για την εισαγωγή και απαλοιφή, σε σύγκριση με το δυνητικό σώσιμο που μπορεί να προστεθεί από την καταστολή κενού. Αν η συχνότητα εμφάνισης υπερβαίνει το δυνητικό ποσοστό της μείωσης δεδομένων, ο αλγόριθμός σας θα δημιουργήσει επέκταση των δεδομένων ακόμα και αν στην πραγματικότητα θα κατέστειλε όλες τις εμφανίσεις τριών ή περισσότερων κενών.

Ενώ η καταστολή κενού θεωρείται ως μια στοιχειώδης τεχνική συμπίεσης δεδομένων, είναι πολύ εύκολο να εφαρμοστεί και οι συνέπειές της μπορεί να είναι ουσιαστικές. Έχουν αναφερθεί ωφέλειες απόδοσης μεταξύ 30 και 50% για έναν αριθμό εγκαταστάσεων υπολογιστών που άλλαξαν από την αλληλουχία ελέγχου 2780 δισύγχρονης μετάδοσης που δεν συμπιέζει δεδομένα στην αλληλουχία 3780 που εκτελεί καταστολή κενού.

2.1.7. Παραλλαγές της τεχνικής

Δύο παραλλαγές της καταστολής κενού μπορούν να χρησιμοποιηθούν για τη συμπίεση τμημάτων εγγράφων που περιέχουν προκαθορισμένες ή μεταβλητές οδοντώσεις. Σε μία περίπτωση, μπορεί να είναι ευεργετικό να διατηρηθεί μια ομάδα χαρακτήρων από το σετ χαρακτήρων για την αναπαράσταση κάποιων προκαθορισμένων αριθμών διαστημάτων ή κενών. Έτσι, ένας χαρακτήρας μπορεί

κατόπιν να αναπαριστάνει την οδόντωση σε ένα γράμμα πέντε διαστημάτων, ενώ ένας δεύτερος χαρακτήρας θα μπορούσε να χρησιμοποιηθεί για την αναπαράσταση 20 διαστημάτων που απαιτούνται για τη στηλοθέτηση στην έναρξη μιας στήλης μέσα σ' ένα έγγραφο.

Αφού οι προκαθορισμένες οδοντώσεις παριστάνουν θέσεις στηλοθέτησης, μια δεύτερη παραλλαγή καταστολής κενού εξάγεται από την εφαρμογή του χαρακτήρα tab (στηλοθέτη). Αν οι στηλοθέτες προκαθοριστούν, χρειάζεται μόνο να αντικαταστήσετε μια σειρά διαστημάτων ή κενών από το χαρακτήρα στηλοθέτη για να δηλώσετε ότι ο επόμενος χαρακτήρας αρχίζει σε μια συγκεκριμένη στήλη της γραμμής, και όλες οι στήλες μεταξύ του τελευταίου χαρακτήρα και της τοποθεσίας όπου αρχίζει ο επόμενος χαρακτήρας είναι διαστήματα ή κενοί χαρακτήρες. Για να επεξηγήσουμε την αντίληψη αυτή, ας υποθέσουμε ότι ένα τμήμα του κειμένου που επιθυμούμε να μεταδώσουμε έχει ως εξής:

Τώρα είναι η ώρα να εξετάσουμε τη σχέση των αμυντικών δαπανών στην οικονομία. Για τα έτη 1980 έως 1984 η ανάλυσή μας δείχνει		
Έτος	Όπλα	Βούτυρο
XXXX	YYYY	ZZZZ

Σημειώστε ότι υπάρχουν τέσσερις ευκρινείς θέσεις στηλοθέτη στο έγγραφο αυτό - η οδόντωση μιας παραγράφου και οι τρεις θέσεις των στηλών. Έτσι, ένας στηλοθέτης που ακολουθείται από το χαρακτήρα 'Ν' θα μπορούσε να χρησιμοποιηθεί για τον καθορισμό της αρχής της παραγράφου στην κατάλληλή της θέση. Αφού η οδόντωση εμφανίζεται

πριν την πρώτη θέση στηλοθέτη για τον καθορισμό του 'Ε' στα έτη θα απαιτούνταν να εφαρμοστούν δυο στηλοθέτες. Παρόμοια, το 'Ο' στα όπλα θα πρέπει να προηγείται από τρεις χαρακτήρες στηλοθέτη κ.λπ. Καθώς ο αριθμός της μοναδικής οδόντωσης και των θέσεων καθορισμού στηλών αυξάνει σ' ένα έγγραφο ή μεταξύ διαφορετικών εγγραφων, ο αριθμός των χαρακτήρων στηλοθετών που χρειάζεται να εφαρμοστούν για να δηλώσουν μια προκαθορισμένη θέση θα μπορούσε να έχει ως αποτέλεσμα την επέκταση των δεδομένων αντί για τη συμπίεσή τους. Για να εμποδιστεί η εμφάνιση τέτοιων καταστάσεων, καθώς επίσης για να εξαλειφθούν η ανάγκη προηγούμενης γνώσης σχετικά με την οδόντωση και των τοποθεσιών στηλών, μια διαδικασία μεταβλητού στηλοθέτη μπορεί να εφαρμοστεί. Χρησιμοποιώντας μεταβλητούς στηλοθέτες απλώς υποκαθιστάς ένα χαρακτήρα στηλοθέτη και την τοποθεσία της στήλης προς στηλοθέτηση στη θέση του διαστήματος μεταξύ των στηλών. Επιστρέφοντας στο προηγούμενο παράδειγμα, αν το 'έτος' (year) ξεκινούσε. στη στήλη 15 ενώ τα 'όπλα' (guns) και το 'βούτυρο' (butter) άρχιζαν στις στήλες 30 και 45 αντίστοιχα, οι επικεφαλίδες της στήλης γραμμής θα αντικαθιστούνταν από την αλληλουχία Ts15YeraTs3OGunsTs45Butter, όπου Ts είναι ο χαρακτήρας στηλοθέτησης.

Μια από τις κύριες εφαρμογές για την αντικατάσταση των αλληλουχιών κενών με στηλοθέτες είναι οι μεταδόσεις CICS υπολογιστών μεγάλων ταχυτήτων. Διάφοροι άσχετοι με αυτήν κατασκευαστές λογισμικού εμπορεύονται προϊόντα που διακόπτουν τις προς τα έξω μεταδόσεις CICS σε τερματικά και, μέσα σε πολλές άλλες λειτουργίες, αντικαθιστούν τις αλληλουχίες κενών με σειρές στηλοθετών. Με τη μετάδοση ενός μικρότερου ποσού δεδομένων

και η ικανότητα του συστήματος μετάδοσης καθώς επίσης και οι χρόνοι απόκρισης του χρήστη βελτιώνονται.

2.2. Χαρτογράφηση των bit

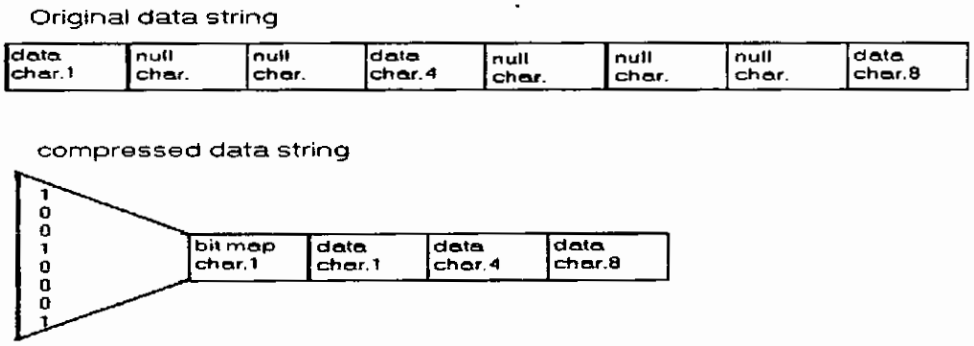
Αυτή η τεχνική συμπίεσης είναι αποτελεσματική όταν τα προς λειτουργία δεδομένα απαρτίζονται από μια υψηλή αναλογία συγκεκριμένων τύπων δεδομένων, όπως τα αριθμητικά, ή μια μεγάλη αναλογία ενός συγκεκριμένου χαρακτήρα, όπως τα κενά. Καθώς δηλώνει και το όνομα, μια χαρτογράφηση bit εφαρμόζεται για να δείξει την παρουσία ή απουσία χαρακτήρων δεδομένων ή το γεγονός ότι συγκεκριμένοι χαρακτήρες δεδομένων έχουν τεθεί σε λειτουργία από πριν και πρέπει να λειτουργήσουν ξανά για να επιστρέψουν τα δεδομένα στην αρχική μορφή τους.

2.2.1. Διαδικασία κωδικοποίησης

Για να εξεταστεί η τεχνική χαρτογράφησης δεδομένων και οι εφαρμογές της, πρώτα θα δούμε πώς μπορεί να εφαρμοστεί για να υλοποιήσει μια εκδοχή της καταστολής κενού. Στο αριστερό τμήμα του Σχήματος 2.7 φαίνεται ένα τμήμα ρεύματος δεδομένων που αποτελείται από τρεις χαρακτήρες δεδομένων και πέντε κενά. Εδώ,

τα πέντε κενά αντιπροσωπεύουν το 62,5% του περιεχομένου της αλληλουχίας και είναι σκορπισμένα μέσα στο ρεύμα δεδομένων σε μια τυχαία σειρά. Αφού η καταστολή κενού είναι αποτελεσματική μόνο όταν μετρούνται τρία ή περισσότερα κενά σε σειρά, η χρήση της θα μείωνε μόνο την αλληλουχία σε μήκος από οκτώ σε επτά χαρακτήρες.

Μέσω της χρήσης μιας χαρτογράφησης bit προσαρτημένης μπροστά από την αλληλουχία, μπορούμε να δείξουμε την παρουσία ή απουσία των κενών και με αυτόν τον τρόπο να μειώσουμε το μέγεθος της αλληλουχίας δεδομένων. Στο χαμηλότερο τμήμα του Σχήματος 2.7 φαίνεται η εφαρμογή ενός χαρακτήρα χαρτογράφησης bit, όπου όλα τα κενά πέφτουν από την αλληλουχία δεδομένων και το bit που αντιστοιχεί στη θέση κενού τίθεται στο μηδέν, ενώ η θέση bit στο χάρτη που ανταποκρίνεται σε ένα μη-κενό ή ένα χαρακτήρα δεδομένων τίθεται στο ένα.



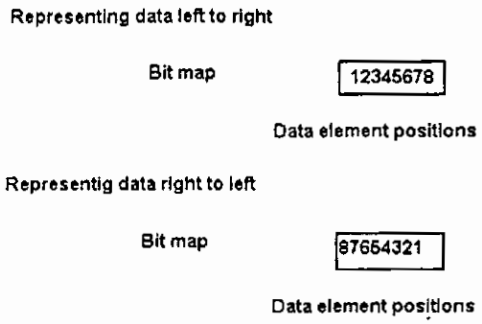
Σχήμα 2.7 Η διαδικασία χαρτογράφησης bit. Σε ένα τυπικό ρεύμα δεδομένων, υπάρχει μια μεγάλη πιθανότητα ένας ή περισσότεροι χαρακτήρες να επαναλαμβάνονται. Η χρησιμοποίηση ενός ένα χαρακτήρα να εξυπηρετεί ως χάρτης bit, μπορεί να εξυπηρετήσει ώστε

να εξαλειφθεί η υψηλή συχνότητα εμφάνισης χαρακτήρων από ένα ρεύμα δεδομένων.

Συγκρίνοντας την αλληλουχία συμπιεσμένων δεδομένων με την αλληλουχία αρχικών δεδομένων, οι οκτώ χαρακτήρες δεδομένων που περιλαμβάνουν κενά έχουν μειωθεί σε τέσσερις χαρακτήρες. τρεις χαρακτήρες δεδομένων και το χαρακτήρα χάρτη bit. Αυτό έχει ως αποτέλεσμα ένα λόγο συμπίεσης 2:1 γι' αυτή τη συγκεκριμένη εφαρμογή.

2.2.2. Θεωρήσεις για το hardware

Ο χαρακτήρας χάρτη bit που φαίνεται στο Σχήμα 2.7 δηλώνει τις μη- κενές θέσεις χαρακτήρων δεδομένων ανά τοποθεσία, από αριστερά προς δεξιά. Αντιστρέφοντας τη διάταξη του χάρτη bit, οι θέσεις στοιχείων δεδομένων μπορούν να δειχτούν από δεξιά προς τα αριστερά. Το σχήμα 2.8 δείχνει τις δυο διαφορετικές μεθόδους στο σχηματισμό του χάρτη bit για την παρουσίαση της αλληλουχίας συμπιεσμένων δεδομένων. Με τη χρήση της τεχνικής τοποθέτησης στοιχείων δεδομένων χάρτη bit να φαίνεται στο χαμηλότερο τμήμα του Σχήματος 2.8, ο χαρακτήρας χάρτη bit που απορρέει από το αρχικό ρεύμα δεδομένων όπως φαίνεται στο Σχήμα 2.7 θα έφτανε το 10001001. Το σετ οδηγιών της υπόψη διάταξης hardware για την εκτέλεση της τεχνικής καταστολής χάρτη bit θα καθορίσει τη



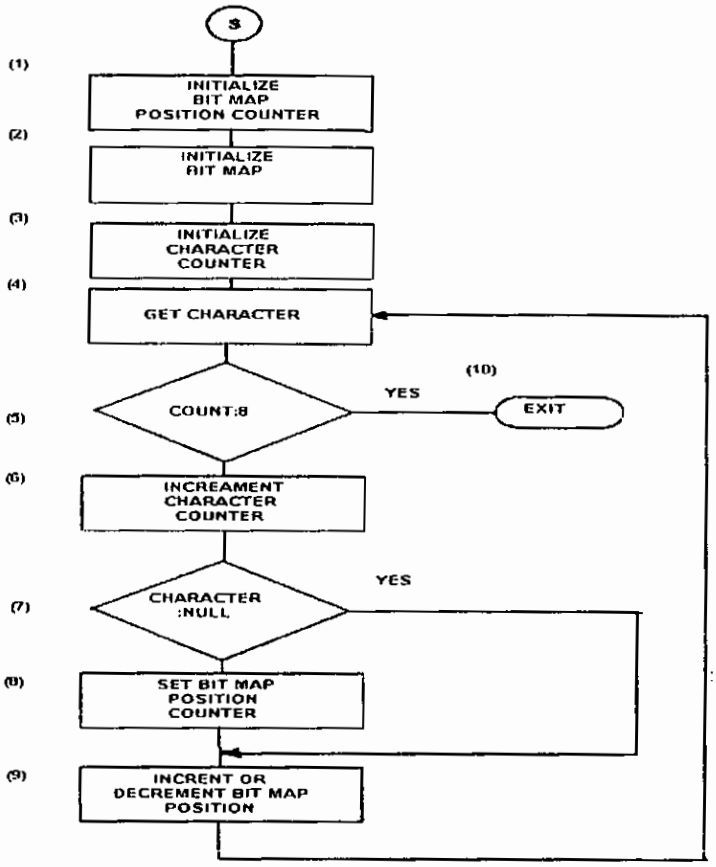
Σχήμα 2.8 Τοποθέτηση στοιχείου χάρτη bit. Δυο μέθοδοι μπορούν να εφαρμοστούν για να παρουσιάσουν την αλληλουχία συμπιεσμένων δεδομένων στο χάρτη bit - τα δεδομένα που παρουσιάζονται από αριστερά προς τα δεξιά και από δεξιά προς τα αριστερά.

μέθοδο της τοποθέτησης στοιχείων χάρτη bit που θα εφαρμοστεί. Αυτό μπορεί να εξηγηθεί εύκολα εξετάζοντας πρώτα ένα διάγραμμα ροής των λειτουργιών που έχουν να εκτελεστούν στην αρχική αλληλουχία δεδομένων ώστε να κατασκευαστεί ο χάρτης bit και η αλληλουχία συμπιεσμένων δεδομένων.

Η διαδικασία καταστολής χάρτη bit φαίνεται λειτουργικά στο Σχήμα 2.9. Η ρουτίνα λογισμικού (software) για τη συμπίεση δεδομένων πρέπει πρώτα να τοποθετήσει αρχικές τιμές στο μετρητή τοποθέτησης χάρτη bit (1), στο χάρτη bit (2) και σ' ένα μετρητή χαρακτήρων (3). Αφού επικρατήσει ένας χαρακτήρας (4), ο μετρητής χαρακτήρα συγκρίνεται με οκτώ (5). Αν εμφανιστεί κάποιο ταίριασμα, οκτώ εισερχόμενοι χαρακτήρες έχουν επεξεργαστεί και μπορούμε να βγούμε από τη ρουτίνα (10). Αν δεν εμφανιστεί ταίριασμα, ο μετρητής χαρακτήρων αυξάνεται (6) και ο υπό εξέταση χαρακτήρας συγκρίνεται με ένα

μηδενικό χαρακτήρα (7). Αν ο υπό εξέταση χαρακτήρας δεν είναι μηδενικός, η τοποθέτηση χάρτη bit τίθεται ίση στο δυαδικό ένα (8). Αν ο χαρακτήρας είναι μηδενικός (κενό), η λειτουργία αυτή (8) παρακάμπτεται. Κατόπιν, η τοποθέτηση χάρτη bit είτε αυξάνεται είτε μειώνεται (9), έτσι ώστε ο χάρτης bit προετοιμάζεται να τεθεί σε μια μηδενική ή στην επόμενη θέση bit αν ο επόμενος χαρακτήρας που εξετάζεται είναι μηδενικός (κενό). Τέλος, αφού έχουν επεξεργαστεί οκτώ χαρακτήρες, η μέτρηση εξισώνεται με το οκτώ (5) και παίρνεται ο κλάδος εξόδου της ρουτίνας (10).

Από ένα σημείο άποψης hardware, η μέθοδος που χρησιμοποιείται για την απόδοση των λειτουργιών που φαίνονται στα block (8) και (9) του Σχήματος 2.9 εξαρτάται από τον τρόπο και τις λογικές οδηγίες που είναι διαθέσιμες για να τις χρησιμοποιήσει ο προγραμματιστής. Αυτή η αλληλοσυσχέτιση μπορεί να θεωρηθεί με την εμφάνιση του αποτελέσματος στο χαρακτήρα χάρτη bit καθώς εξετάζονται διαδοχικοί χαρακτήρες δεδομένων. Στο Σχήμα 2.10 φαίνεται το φαινόμενο στο χάρτη bit και η 'μάσκα' καθώς εξετάζεται μια πρόοδος των χαρακτήρων δεδομένων. Εδώ, η μάσκα είναι απλά μια δυαδική μονάδα που μετατοπίζεται (shift) μέσω των θέσεων χάρτη των 8-bit και λογικά `OR'd με το χάρτη bit όταν ο χαρακτήρας δεδομένων δεν είναι μηδενικός (κενό). Εξετάζοντας τη μάσκα, μπορούμε να παρατηρήσουμε ότι μια λογική λειτουργία ή μια λειτουργία αριθμητικού προς τα αριστερά shift απαιτείται αν επιθυμούμε να τοποθετήσουμε το χάρτη μας bit έτσι ώστε το προς το δεξί μας χέρι bit να δείχνει την παρουσία ή απουσία ενός μηδενικού χαρακτήρα στο πρώτο στοιχείο της αρχικής αλληλουχίας δεδομένων. Έτσι, από την άποψη του hardware, η οδηγία shift που είναι διαθέσιμη θα είναι ο κυβερνών



Σχήμα 2.9. Διάγραμμα ροής λειτουργίας καταστολής χάρτη bit.

παράγοντας, σύμφωνα και με το πώς τα στοιχεία χάρτη bit τοποθετούνται. Παρόλο που οι περισσότεροι μικροεπεξεργαστές, μίνι-υπολογιστές και βεβαίως όλοι οι μεγάλοι υπολογιστές έχουν και δεξιά και αριστερά λειτουργίες μετατόπισης (shift), μερικοί μικροεπεξεργαστές ίσως έχουν περιορισμένη ικανότητα μετατόπισης. Τέτοια ικανότητα θα πρέπει να εξετάζεται πριν να επιχειρηθεί η εφαρμογή της τεχνικής αυτής.

2.2.3. Αποτελεσματικότητα καταστολής

Στο προηγούμενο παράδειγμα ο χαρακτήρας χάρτη bit περιείχε οκτώ bits. Ενώ το παράδειγμα έδειξε μια 50% μείωση στους χαρακτήρες από την αλληλουχία αρχικών δεδομένων προς την αλληλουχία συμπιεσμένων δεδομένων, σκεφτείτε τι θα συνέβαινε στην ικανότητα συμπίεσης όταν το ποσοστό κενών στην αλληλουχία δεδομένων μειώνεται. Ο Πίνακας 2.1 δείχνει το λόγο συμπίεσης που

Data character	Initial bit map	Mask	Bit map ⊕ mask (or bit map if null)
	00000000	00000001	00000001
Data	00000001	00000010	00000001
Null	00000001	00000100	00000101
Data	00000101	00001000	00001101
Data	00001101	00010000	00001101
Null	00001101	00100000	00001101
Null	00001101	01000000	00001101
Null	00001101	10000000	10001101
Data			

Σχήμα 2.10 Η διαδικασία μάσκας χάρτη bit. Ο χαρακτήρας μάσκας είναι μια δυαδική μονάδα που μετατοπίζεται μέσα σε όλες τις

θέσεις bit και λογικώς OR'd με το χάρτη bit όταν ο χαρακτήρας δεδομένων δεν είναι κενό.

βασίζεται στο ποσοστό κενών που περιέχονται στην αλληλουχία για ένα χάρτη των 8-bit. Όταν δεν υπάρχουν μηδενικοί (κενοί) χαρακτήρες, η προκύπτουσα αλληλουχία δεδομένων αυξάνεται σε μέγεθος κατά ένα χαρακτήρα ως αποτέλεσμα της πρόσθεσης του χαρακτήρα χάρτη bit, δημιουργώντας ένα λόγο συμπίεσης 0.888, αυτό σημαίνει ότι στη χειρότερη περίπτωση όπου δεν υπάρχουν κενοί χαρακτήρες για να συμπιεστούν, ένα επιπλέον 12,5% δεδομένων θα προκύψει από την εφαρμογή αυτής της μεθόδου συμπίεσης.

Μπορούμε να αναπτύξουμε ένα μαθηματικό μοντέλο αποτελεσματικότητας καταστολής ως ακολούθως. Αν ρ είναι η πιθανότητα κάθε δοσμένος χαρακτήρας να είναι κενό, ο αναμενόμενος αριθμός κενών σε μια αλληλουχία μήκους S χαρακτήρων είναι Sp. Με τη χρήση της συμπίεσης κενού αυτό θα κωδικοποιούνταν σαν μια αλληλουχία μήκους :

$$S*(1-p)+[S/8\ ]$$

και επομένως ο λόγος συμπίεσης είναι

$$\left(\frac{1}{S}\left\{S(1-p)-\left[\frac{S}{8}\right]\right\}\right)^{-1}\approx\left((1-p)+\frac{1}{8}\right)^{-1}$$

για μεγάλες τιμές του S.

Null percentage	Resultant string size	Compression ratio
0.0	9	0.888
12.5	8	1.000
25.0	7	1.143
37.5	6	1.334
50.0	5	1.600
62.5	4	2.000
75.0	3	2.667
87.5	2	4.000
100.0	1	8.000

Πίνακας 2.1. Αποτελεσματικότητα συμπίεσης και ποσόστωση κενών

2.2.4. Παραλλαγές του χάρτη bit

Στην προηγούμενη συζήτηση της διαδικασίας χάρτη bit, υποθέσαμε ότι είτε ένα κενό είτε άλλος χαρακτήρας που εμφανίζεται σε μεγάλη αναλογία στο υπόλοιπο των δεδομένων πρέπει να κατασταλεί. Για κάποιες εφαρμογές, δεν υπάρχει κάποιος συγκεκριμένος χαρακτήρας που να μετράται περισσότερο συχνά από άλλους χαρακτήρες ωστόσο, σε ορισμένες περιπτώσεις κάποιος μπορεί να μετρήσει μια κατάσταση όπου ένας συγκεκριμένος τύπος δεδομένων, όπως αριθμοί, εμφανίζεται συχνά. Μια εφαρμογή όπου μια τέτοια κατάσταση θα μπορούσε να υπάρχει είναι η περιοχή επεξεργασίας ελέγχου όπου αριθμητικές αναγνώσεις διάφορου εξοπλισμού μεταδίδονται σε μια κεντρική θέση για επεξεργασία και σήματα ελέγχου επιστρέφουν στις διατάξεις βάσει κάποιων

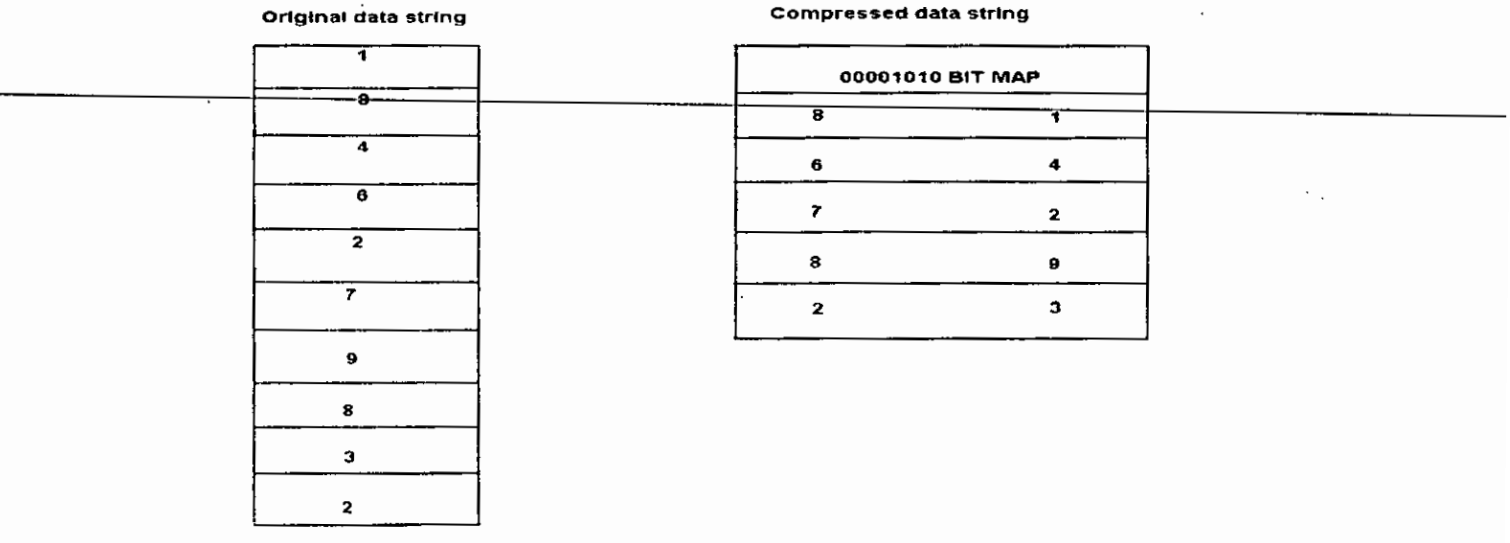
προκαθορισμένων κριτηρίων. Ανάλογα με τον κώδικα μετάδοσης που εφαρμόζεται, μπορούν να επιτευχθούν ορισμένες οικονομίες με τη χρήση της τεχνικής χάρτη bit. Αν τα προς μετάδοση δεδομένα είναι σε έναν επαυξημένο δυαδικό-δεκαδικό κώδικα εναλλαγής (Extended Binary Coded Decimal Interchange Code - EBCDIC), τότε οι πρώτες τέσσερις θέσεις bit κάθε αριθμητικού χαρακτήρα είναι όλες μονάδες. 'Ετσι, ο χαρακτήρας χάρτη bit θα μπορούσε να εφαρμοστεί για να δηλώσει τον αριθμό των σε δέσμη χαρακτήρων στην συμπιεσμένη αλληλουχία, με κάθε χαρακτήρα να περιέχει δυο ψηφία με τις αρχικές τέσσερις θέσεις bit να έχουν αφαιρεθεί. Η τεχνική αυτή φαίνεται στο Σχήμα 2.11.

2.2.5. Περιορισμοί τεχνικής

'Ενας βασικός περιορισμός της τεχνικής χάρτη bit είναι ότι είναι εφαρμόσιμη σε δεδομένα που έχουν κανονισμένες μονάδες μεγέθους, όπως χαρακτήρες, bytes ή λέξεις. Όταν χρησιμοποιείται για να καταστείλει ένα συγκεκριμένο χαρακτήρα, όπως κενό, ο λόγος συμπίεσης της τεχνικής αυτής είναι άμεσα ανάλογος με το ποσοστό εμφάνισης του χαρακτήρα αυτού στο αρχικό ρεύμα δεδομένων. 'Ετσι, εάν κάποιος χαρακτήρας σε μια αλληλουχία δεδομένων εμφανίζεται στο 30% του χρόνου ενώ ο δεύτερος περισσότερο συχνά μετρούμενος χαρακτήρας εμφανίζεται, ας πούμε, στο 25% του χρόνου, η τεχνική αυτή αγνοεί το υψηλό ποσοστό εμφάνισης του δεύτερου χαρακτήρα ή οποιουσδήποτε άλλους χαρακτήρες. Υπάρχει μια τεχνική γνωστή ως

κωδικοποίηση εκτέλεσης μήκους που μπορεί να εφαρμοστεί για να επωφεληθεί από το συνεχόμενο πλεονασμό εμφάνισης όλων των χαρακτήρων σε ένα ρεύμα δεδομένων.

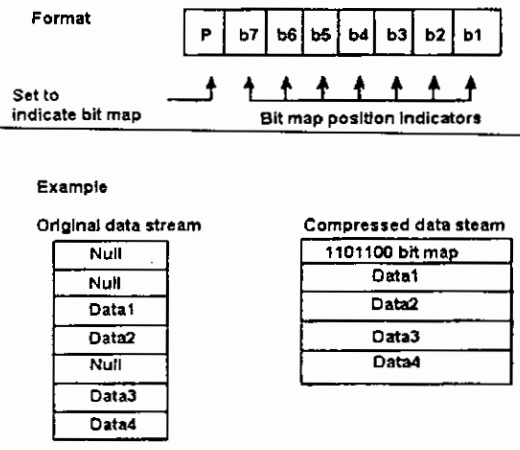
Δυο άλλοι περιορισμοί που σχετίζονται με την τεχνική χάρτη bit περιλαμβάνουν την απαίτηση να λειτουργεί πάνω σε έναν τύπο χαρακτήρα μόνο, όπως κενά ή ψηφία, και την ανάπτυξη μιας μεθόδου αναγνώρισης ότι ένας χαρακτήρας αναπαριστάνει ένα χάρτη bit αντί για δεδομένα. Ας εξετάσουμε πώς μπορούμε να υπερπηδήσουμε το δεύτερο περιορισμό όταν χρησιμοποιούμε δεδομένα ASCII 7-επιπέδου και πώς μπορούμε να αναπτύξουμε μια μέθοδο για να υπερπηδήσουμε τον περιορισμό όταν χρησιμοποιούμε άλλα σετ χαρακτήρων.



Σχήμα 2.11 Καταστολή ημι-ψηφίου. Στην τεχνική καταστολής ημι-ψηφίου, τα περιεχόμενα του χάρτη bit καθορίζουν τον αριθμό ψηφίων που ακολουθούν, πακεταρισμένα δύο ανά χαρακτήρα

Αν χρησιμοποιούμε 7-επίπεδο ASCII μπορούμε να αφαιρέσουμε την ισότητα των bit και να χρησιμοποιήσουμε αυτή την όγδοη θέση bit ως έναν δείκτη χάρτη bit. Παρόλο που η τεχνική αυτή μειώνει το μέγιστο λόγο συμπίεσης που μπορεί να επιτευχθεί μέσα από τη χρήση της τεχνικής χάρτη bit, επίσης μειώνει την πιθανότητα επέκτασης των δεδομένων. Για να δείξουμε πώς μπορούμε να αποφύγουμε την επέκταση των δεδομένων ας εξετάσουμε τη λειτουργία αυτής της αναθεωρημένης τεχνικής.

Το Σχήμα. 2.12 δείχνει το σχηματισμό ενός δείκτη χάρτη bit μέσω της τοποθέτησης της θέσης bit 8 η οποία στην 7επίπεδη ASCII χρησιμοποιείται ως bit ισότητας. Στο χαμηλότερο τμήμα του Σχήματος 2.12 φαίνεται το επακόλουθο συμπιεσμένο ρεύμα δεδομένων,



Σχήμα 2.12 Χρήση του bit ισότητας ως δείκτη χάρτη bit

βασισμένο πάνω σ' ένα αρχικό ρεύμα δεδομένων επτά χαρακτήρων. Τώρα ας υποθέσουμε ότι η σύνθεση των επόμενων επτά χαρακτήρων αλλάζει, ώστε να μην μετρηθούν κενά. Κάτω από αυτή την

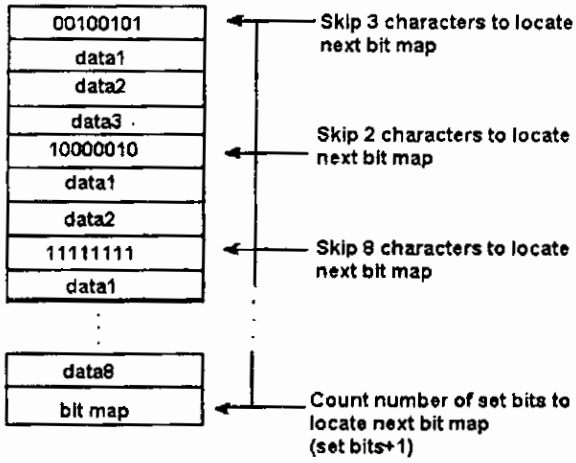
αναθεωρημένη τεχνική οι χαρακτήρες απλά θέτονται στο ρεύμα των συμπιεσμένων δεδομένων χωρίς την αναγκαιότητα πρόσθεσης ενός χάρτη bit ο οποίος θα μπορούσε να έχει ως αποτέλεσμα την επέκταση των δεδομένων. Εδώ τα δεδομένα αναγνωρίζονται ως δεδομένα αφού η θέση ισότητας bit δεν έχει τεθεί. Άλλο ένα πλεονέκτημα που σχετίζεται μ' αυτή την αναθεωρημένη τεχνική συμπίεσης είναι το γεγονός ότι η τοποθέτηση της προηγούμενης θέσης ισότητας bit έχει ως αποτέλεσμα την αυτόματη αναγνώριση του χαρακτήρα ως η 7-θέση χάρτη bit. Αυτό σας επιτρέπει ελαχιστοποίηση της κωδικοποίησης για να κρατηθούν τα ίχνη των χαρτών bit ενάντια στα δεδομένα.

Αν χρησιμοποιείτε κάθε τοποθεσία bit μέσα σ' ένα χαρακτήρα, όπως με την EBCDIC 8-επιπέδων, πρέπει κατόπιν να κρατήσετε τα ίχνη της σχέσης μεταξύ των χαρακτήρων χάρτη bit και των χαρακτήρων δεδομένων. Για να γίνει κάτι τέτοιο, μπορείτε απλά να μετρήσετε τον αριθμό των σετ των bits στον πρώτο χάρτη bit, ο οποίος κατόπιν σας λέει τον αριθμό των χαρακτήρων που πρέπει να παραληφθούν πριν την ανάγνωση του επόμενου χάρτη bit. Το Σχήμα 2.13 δείχνει τη χρήση της μέτρησης των σετ των bits στο χάρτη bit για τον εντοπισμό της τοποθεσίας του επόμενου χάρτη bit στο ρεύμα δεδομένων.

2.3. Μήκος εκτελεσης

Η κωδικοποίηση μήκους εκτέλεσης είναι μια μέθοδος συμπίεσης δεδομένων που θα μειώσει με φυσικό τρόπο οποιοδήποτε τύπο επαναλαμβανόμενης αλληλουχίας χαρακτήρων, από τη στιγμή που η

αλληλουχία αυτή φτάσει ένα προκαθορισμένο επίπεδο εμφάνισης. Για την ιδιαίτερη περίπτωση όπου ο κενός χαρακτήρας είναι ο επαναλαμβανόμενος χαρακτήρας, η συμπίεση μήκους εκτέλεσης μπορεί να θεωρηθεί ως υπέρθεση της καταστολής κενού (Rubin, 1976~ Ruth και Kreutzer, 1972).



Σχήμα 2.13 Εντοπισμός χαρτών bit. Οι επόμενοι χάρτες bit μπορούν να εντοπιστούν στο ρεύμα δεδομένων με την πρόσθεση του 1 στο μέτρημα του αριθμού των σετ των bits στον προηγούμενο χάρτη bit.

2.3.1. Λειτουργία

Με έναν τρόπο παρόμοιο με τη μέθοδο που χρησιμοποιήθηκε για την επίδειξη της καταστολής κενού, η εφαρμογή της κωδικοποίησης μήκους εκτέλεσης κανονικά απαιτεί τη χρήση ενός ειδικού χαρακτήρα

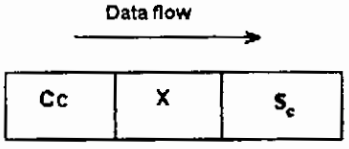
που δηλώνει ότι αυτός ο τύπος συμπίεσης έχει εμφανιστεί. Μια εξαίρεση στη χρήση ενός ειδικού χαρακτήρα δείκτη της συμπίεσης είναι η μέθοδος συμπίεσης δεδομένων Microcom Networking Protocol (MNP) Class 5. Οταν χρησιμοποιείται ένας χαρακτήρας που δείχνει τη συμπίεση, συνήθως ακολουθείται από έναν από τους επαναλαμβανόμενους χαρακτήρες ο οποίος βρισκόταν στην αριθμημένη αλληλουχία των γεμάτων επαναλήψεις χαρακτήρων. Τέλος, ένας χαρακτήρας απαρίθμησης δηλώνει τον αριθμό των φορών που έχει εμφανιστεί ο επαναλαμβανόμενος χαρακτήρας στην αλληλουχία.

Όταν εφαρμόζονται κώδικες όπως ο ASCII ή ο EBCDIC, μια καλή επιλογή για τον ειδικό χαρακτήρα είναι ένας που δεν θα εμφανίζεται στην αλληλουχία δεδομένων. Για καθένα από τους κώδικες αυτούς υπάρχουν πολυάριθμοι μη προσδιορισμένοι χαρακτήρες με μοναδικές παραστάσεις bit που μπορούν να χρησιμοποιηθούν. Για καταστάσεις όπου το σετ χαρακτήρων δεν περιέχει μη χρησιμοποιημένο χαρακτήρα, όπως είναι ο κώδικας Baudot 5-επίπεδου (bit), η τεχνική αυτή μπορεί ακόμα να χρησιμοποιηθεί με την επιλογή ενός χαρακτήρα ο οποίος μπορεί να μη χρησιμοποιείται δυο φορές διαδοχικά, όπως μια αλλαγή (shift) γράμματος ή αλλαγή (shift) φιγούρας, για να δείξει ότι έχει εμφανιστεί συμπίεση.

2.3.2. Διαδικασία κωδικοποίησης

Η διαδικασία συμπίεσης εκτέλεσης μήκους έχει ως αποτέλεσμα μια αλληλουχία επαναλαμβανόμενων χαρακτήρων να μετατρέπεται σε αλληλουχία συμπιεσμένων δεδομένων, όπως φαίνεται στο Σχήμα 2.14 (Aronson, 1977). Με τρεις χαρακτήρες να απαιτούνται για να δηλώσουν τη συμπίεση, η κωδικοποίηση εκτέλεσης μήκους είναι αποτελεσματική μόνο όταν μια αλληλουχία δεδομένων περιέχει μια σειρά περισσότερων επαναλαμβανόμενων χαρακτήρων δεδομένων.

Τρία παραδείγματα της εφαρμογής της κωδικοποίηση ς εκτέλεσης μήκους πάνω σε σειρές επαναλαμβανόμενου χαρακτήρα παρουσιάζονται στον Πίνακα 2.2. Σημειώστε ότι το Sc δηλώνει τον ειδικό χαρακτήρα που χρησιμοποιείται για να δείξει την εμφάνιση της κωδικοποίησης εκτέλεσης μήκους, ενώ το σύμβολο ..... χρησιμοποιείται για να δείξει την παρουσία ενός κενού χαρακτήρα.



Σχήμα 2.14 Κωδικοποίηση εκτέλεσης μήκους, γενική μορφή συμπίεσης. Στην κωδικοποίηση εκτέλεσης μήκους, ένας ειδικός χαρακτήρας, επαναλαμβανόμενος χαρακτήρας δεδομένων και μέτρηση χαρακτήρων απαιτούνται για να δείξουν τις παραμέτρους συμπίεσης.

Sc = Ειδικός χαρακτήρας που δείχνει τη συμπίεση που ακολουθεί.

X = Οποιοσδήποτε επαναλαμβανόμενος χαρακτήρας δεδομένων.

Cc = Μέτρηση χαρακτήρων. Η μέτρηση αυτή είναι ο αριθμός φορών που επαναλαμβάνεται ο συμπιεσμένος χαρακτήρας.

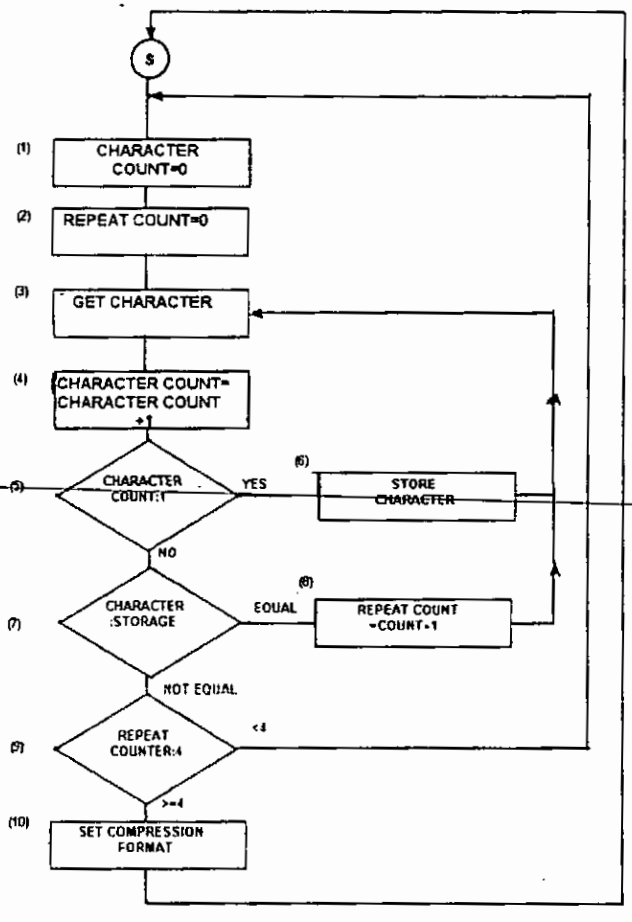
Original data string	Encoded data string
$******55.72	$$S_c$*655.72
---------	S_c-9
Guns[illegible]Butter	GunsS_c[illegible]Butter

Πίνακας 2.2 Εφαρμογή της κωδικοποίησης μήκους εκτέλεσης

Με τη μορφή της καταστολής κενού να απαιτεί δυο χαρακτήρες, η εφαρμογή της συμπίεσης εκτέλεσης μήκους για την καταστολή των κενών έχει ως αποτέλεσμα να δημιουργείται ένας επιπλέον χαρακτήρας στο ρεύμα των συμπιεσμένων δεδομένων. Ενώ αυτό δεν είναι σημαντικό όταν συμπιέζονται μακρές αλληλουχίες κενών, πολυάριθμες κοντές αλληλουχίες θα μπορούσαν να προκύψουν σε μια πλεονάζουσα ποσότητα συμπιεσμένων δεδομένων. Αυτό σημαίνει ότι θα πρέπει να σκεφτείτε τη χρήση μιας μείξης διάφορων αλγόριθμων για να εκτελέσετε συμπίεση δεδομένων.

Τα κύρια βήματα στη διαδικασία κωδικοποίησης εκτέλεσης μήκους φαίνονται στο Σχήμα 2.15 μέσα από τη χρήση ενός διαγράμματος ροής συστημάτων. Αρχικά, ένας μετρητής χαρακτήρων (1) και ένας μετρητής επανάληψης χαρακτήρων (2) τίθενται στο μηδέν. Κατόπιν εξαχθεί ένας χαρακτήρας στην αλληλουχία αρχικών

δεδομένων (3), ο μετρητής χαρακτήρων αυξάνει (4) κατά ένα. ο μετρητής χαρακτήρων κατόπιν συγκρίνεται με το ένα (5). Στον πρώτο κύκλο, η σύγκριση αυτή πάντα είναι αληθινή και ο χαρακτήρας κατόπιν τοποθετείται σε μια περιοχή προσωρινού καταχωρητή (προσωρινή αποθήκευση) (6) για μεταγενέστερη επεξεργασία αν η αλληλουχία



Σχήμα 2.15 Βασική διαδικασία κωδικοποίησης μήκους εκτέλεσης

αρχικών δεδομένων βρεθεί να περιέχει τέσσερις ή περισσότερους επαναλαμβανόμενους χαρακτήρες δεδομένων. Για τον δεύτερο και

επακόλουθους κύκλους, ο χαρακτήρας που εξάγεται από την αλληλουχία αρχικών δεδομένων (3) συγκρίνεται με το χαρακτήρα που τοποθετήθηκε προς αποθήκευση (7). Αν ο τωρινός χαρακτήρας είναι ίσος με το χαρακτήρα που έχει αποθηκευτεί, η συμπίεση μπορεί να είναι δυνατή εάν μετρηθούν τέσσερις ή περισσότεροι ταυτόσημοι χαρακτήρες σε σειρά. Έτσι, όταν ο χαρακτήρας ισούται με τον αποθηκευμένο χαρακτήρα, ο μετρητής επανάληψης (8) αυξάνεται ανά ένα και άλλος ένας χαρακτήρας εξάγεται από την αλληλουχία αρχικών δεδομένων (3). Εάν ο παρών χαρακτήρας που εξετάζεται δεν ισούται με τον αποθηκευμένο χαρακτήρα (7), ο μετρητής επανάληψης συγκρίνεται με το τέσσερα (9). Εάν είναι λιγότερο από τέσσερα, δεν αξίζει να γίνει συμπίεση, αφού τρεις χαρακτήρες πρέπει να χρησιμοποιηθούν για να κωδικοποιήσουν συμπιεσμένα δεδομένα. 'Οταν ο μετρητής επανάληψης ισούται με ή είναι μεγαλύτερος από τέσσερα (9) η μορφή συμπίεσης (10) μπορεί πλέον να τεθεί.

2.3.3. Ειδικές σκέψεις

Στο βασικό διάγραμμα ροής κωδικοποίησης που φαίνεται στο Σχήμα 2.15, έγινε η υπόθεση ότι ο μετρητής επανάληψης ήταν ικανός να έχει μια απεριόριστη ακτίνα τιμών. Στην πραγματικότητα, η μέγιστη τιμή που μπορεί να περιέχει ο μετρητής επαναλήψεων είναι μια λειτουργία του επιπέδου κώδικα χαρακτήρων που εφαρμόστηκε. Για έναν 8-επίπεδο κώδικα χαρακτήρων (οκτώ bits ανά χαρακτήρα), μπορεί να παρουσιαστεί ένα μέγιστο μεταξύ 255 και 260

επαναλαμβανόμενων χαρακτήρων από τον μετρητή χαρακτήρων. Η ακριβής τιμή θα εξαρτηθεί από το πώς εφαρμόζεται ο μετρητής χαρακτήρων. Στις περισσότερες καταστάσεις, η πραγματική τιμή μετρητή χαρακτήρων χρησιμοποιείται ως ο αριθμός των επαναλαμβανόμενων χαρακτήρων. Με τον τρόπο αυτό, η μέγιστη τιμή του μετρητή είναι 2^8-1 ή 255. Αφού η μορφή συμπίεσης που φαίνεται στο Σχήμα 2.14 εμφανίζεται μόνο όταν μετρηθούν τέσσερις ή περισσότεροι επαναλαμβανόμενοι χαρακτήρες, η παρουσία ενός χαρακτήρα μέτρησης χαρακτήρα από μόνη της υπονοεί ότι υπάρχουν τέσσερις ή περισσότεροι επαναλαμβανόμενοι χαρακτήρες. Έτσι, ένας μετρητής χαρακτήρων με όλα τα bits μηδέν μπορεί να χρησιμοποιηθεί για να δείξει τέσσερις επαναλαμβανόμενους χαρακτήρες, ενώ ένας μετρητής χαρακτήρων με όλα τα bits να τίθενται στο 1 θα έδειχνε 260 επαναλαμβανόμενους χαρακτήρε . Αφού η μέθοδος εφαρμογής του μετρητή επανάληψης καθοριστεί, το διάγραμμα ροής στο Σχήμα 2.15 πρέπει να τροποποιηθεί για να προστεθεί μια επιπρόσθετη σύγκριση μετρητή επανάληψης για να δοκιμαστεί η μέγιστη τιμή που επιτρέπεται να αποθηκευτεί στο μετρητή χαρακτήρων.

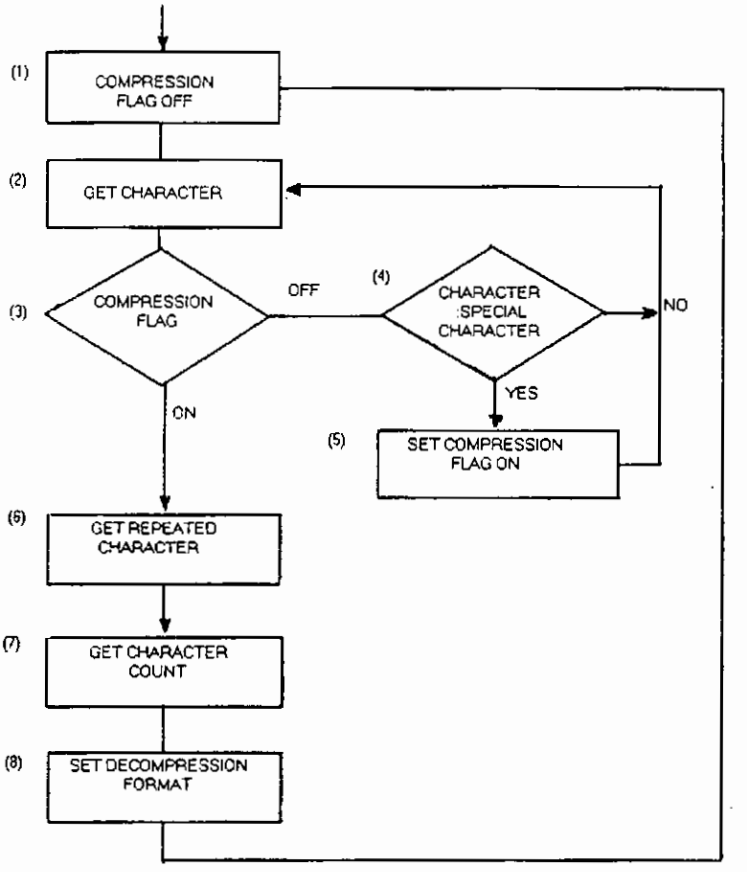
2.3.4. Αποκωδικοποίηση

Οι λειτουργίες που είναι αναγκαίες για την αποσυμπίεση συμπιεσμένων δεδομένων, σύμφωνα με τη διαδικασία κωδικοποίησης μήκους εκτέλεσης, φαίνονται στο Σχήμα 2.16 σε μορφή

διαγράμματος ροής. Στην αρχή της διαδικασίας αποσυμπίεσης, ένας δείκτης συμπίεσης κλείνεται (1) και ένας χαρακτήρας εξάγεται από την αλληλουχία των συμπιεσμένων δεδομένων (2). Κατόπιν, αν ο δείκτης συμπίεσης είναι κλειστός (3), ο χαρακτήρας συγκρίνεται με τον ειδικό χαρακτήρα ένδειξης συμπίεσης μήκους εκτέλεσης (4), για να καθορίσει εάν εμφανίστηκε συμπίεση μήκους εκτέλεσης. Εάν ο χαρακτήρας δεν είναι ο ειδικός χαρακτήρας, λαμβάνεται ο επόμενος χαρακτήρας (2). Αν ο χαρακτήρας είναι ο χαρακτήρας ένδειξης συμπίεσης μήκους εκτέλεσης, ανοίγει ο δείκτης συμπίεσης (5) και ο επόμενος χαρακτήρας λαμβάνεται (2). Στο επόμενο πέρασμα, αφού είναι ανοιχτός ο δείκτης συμπίεσης (3), ο επόμενος χαρακτήρας που λαμβάνεται (6) είναι ο επαναλαμβανόμενος χαρακτήρας δεδομένων, ενώ ο επόμενος χαρακτήρας 97) περιέχει το μέτρημα χαρακτήρων. Αφού ληφθούν αυτοί οι χαρακτήρες, μπορεί να ξεκινήσει η μορφή αποσυμπίεσης (8).

2.3.5. Χρησιμοποίηση

Μια από τις περισσότερο δημοφιλείς χρήσεις της κωδικοποίησης μήκους εκτέλεσης είναι το υποσύνολο που είναι γνωστό ως καταστολή κενού. Αυτή η τεχνική συμπίεσης πρωταρχικά συναντάται στο πρωτόκολλο IBM 3780 BISYNC. Η συμπίεση χώρου είναι ένα καθορισμένο χαρακτηριστικό του πρωτοκόλλου αυτού όταν λειτουργεί η διάταξη 3780 σε γραμμικό τρόπο με μη-διαυγή δεδομένα. Εδώ, κάθε



Σχήμα 2.16. Διαδικασία αποκωδικοποίησης μήκους εκτέλεσης ομάδα των δύο ή περισσότερων συνεχόμενων χαρακτήρων διαστήματος, μέχρι 63, αντικαθιστάται από έναν χαρακτήρα IGS αν ο κώδικας μετάδοσης είναι ο EBCDIC ή ένα GS χαρακτήρα αν ο κώδικας είναι ο ASCII. Και ο ένας και ο άλλος χαρακτήρας ακολουθούνται από ένα χαρακτήρα μέτρησης διαστήματος που καθορίζει τον αριθμό διαστημάτων που απομακρύνθηκαν. Για την κατάσταση όπου 64 ή περισσότεροι συνεχόμενοι χαρακτήρες διαστήματος εμφανίζονται, εισάγονται ένας συμπληρωματικός IGS ή GS χαρακτήρας και ένας χαρακτήρας μέτρησης διαστήματος.

Στα υπολογιστικά συστήματα Honeywell και σε μερικά Bull, χρησιμοποιείται μια παραλλαγή της συμπίεσης μήκους εκτέλεσης στο λογισμικό των γενικών απομακρυσμένων τερματικών συστημάτων τους (general remote terminal system - GRTS) στους επεξεργαστές που επικοινωνούν με απομακρυσμένα τερματικά κάτω από το πρωτόκολλο απομακρυσμένου υπολογιστή (remote computer - RC). Επιπρόσθετα, ο ίδιος τύπος συμπίεσης χρησιμοποιείται και στο ξεχωριστό σύστημα ταινίας προς ταινία (stand alone tape-to-tape system - SATTS) της Honeywell για τη μετάδοση κυλίνδρων μαγνητικής ταινίας μεταξύ τοποθεσιών. Στην εκδοχή αυτή της κωδικοποίησης μήκους εκτέλεσης, μια εγγραφή εξετάζεται για μια σειρά τριών ή περισσότερων εμφανίσεων του ίδιου χαρακτήρα δεδομένων. Όταν προκύψει μια τέτοια κατάσταση, οι σειρές συμπιέζονται και μια αλληλουχία επαναλαμβανόμενων χαρακτήρων μετατρέπεται σε μια αλληλουχία τριών χαρακτήρων, όπως φαίνεται στο Σχήμα 2.17.

Η περισσότερο δημοφιλής υλοποίηση της κωδικοποίησης μήκους εκτέλεσης στην πραγματικότητα αντιπροσωπεύει μια ελαφρά παραλλαγή της προηγουμένως περιγραφείσας τεχνικής συμπίεσης δεδομένων. Η παραλλαγή της κωδικοποίησης μήκους εκτέλεσης που έχει επιτύχει μια εγκατεστημένη βάση περίπου ενός εκατομμυρίου προϊόντων είναι η τεχνική κωδικοποίησης μήκους εκτέλεσης εφαρμοσμένη από την Τάξη 5 του Πρωτοκόλλου Microcom Networking (MNP), που έχει πατενταριστεί προς χρήση από περισσότερους από 50 κατασκευαστές μόντεμ διακοπτόμενου δικτύου.

Η συμπίεση δεδομένων MNP Τάξης 5 χρησιμοποιεί ένα συνδυασμό δυο τεχνικών συμπίεσης δεδομένων - την κωδικοποίηση

μήκους εκτέλεσης και την κωδικοποίηση προσαρμόσιμης συχνότητας.

Σχήμα 2.17 Η παραλλαγή κωδικοποίησης μήκους εκτέλεσης Honeywell και Bull. Η κωδικοποίηση μήκους εκτέλεσης όπως εφαρμόζεται στα υπολογιστικά συστήματα Honeywell και Bull διαφέρει ελαφρά από τους περισσότερους άλλους κατασκευαστές υπολογιστών.

X = Οποιοσδήποτε επαναλαμβανόμενος χαρακτήρας δεδομένων.

US = Ο χαρακτήρας ASCII (0011111).

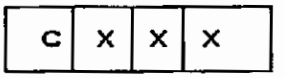
Cc = Ένα δυαδικό μέτρημα των 6-bit. Ο χαρακτήρας BCD που παρουσιάζεται από το δυαδικό μέτρημα πρέπει να μεταφραστεί σε ASCII για μετάδοση στο υποσύστημα επικοινωνιών. Το μέτρημα αυτό είναι ο αριθμός φορών που πρόκειται να επαναληφθεί ο συμπιεσμένος χαρακτήρας (μέγιστος 63)

Κάτω από τη μέθοδο κωδικοποίησης μήκους εκτέλεσης NMP Τάξης 5, ένα μέτρημα επανάληψης εισάγεται στο ρεύμα δεδομένων για να εκφράζει τον αριθμό των επαναλαμβανόμενων οκτάδων που ακολουθούν τις τρεις πρώτες εμφανίσεις μιας αλληλουχίας. Αφού η παραλλαγή MNP της κωδικοποίησης μήκους εκτέλεσης στέλνει τις πρώτες τρεις κωδικοποίησης μήκους εκτέλεσης επαναλαμβανόμενες οκτάδες, αυτοί οι επαναλαμβανόμενοι χαρακτήρες εξυπηρετούν ως ένας δείκτης της έναρξης της κωδικοποίησης μήκους εκτέλεσης και, ως αποτέλεσμα, εξαφανίζουν την αναγκαιότητα χρησιμοποίησης ενός ειδικού

χαρακτήρα δείκτη της συμπίεσης. Το επάνω τμήμα του Σχήματος 2.18 δείχνει τη μορφή της κωδικοποίησης μήκους εκτέλεσης MNP τάξης 5.

Το χαμηλότερο τμήμα του Σχήματος 2.18 περιέχει πέντε παραδείγματα της λειτουργίας κωδικοποίησης μήκους εκτέλεσης MNP Τάξης 5 πάνω σε διαφορετικές αλληλουχίες. Σημειώστε ότι κάτω από την εκδοχή αυτή της κωδικοποίησης μήκους εκτέλεσης, μια επαναλαμβανόμενη σειρά ακριβώς Τριών χαρακτήρων έχει ως αποτέλεσμα τη χρήση ενός μετρητή επανάληψης μηδέν να εισάγεται στο ρεύμα των συμπιεσμένων δεδομένων. Επιπρόσθετα, η μέγιστη μέτρηση επανάληψης που υποστηρίζεται από το MNP περιορίζεται σε 250 δεκαδικά. 'Ετσι, αυτό το τμήμα της συμπίεσης δεδομένων MNP Τάξης 5 μπορεί να έχει ως αποτέλεσμα την επέκταση των δεδομένων όταν μετρούνται σειρές τριών επαναλαμβανόμενων χαρακτήρων στο ρεύμα των αρχικών δεδομένων. Επιπρόσθετα, η MNP εκδοχή της κωδικοποίησης μήκους εκτέλεσης απαιτεί τέσσερις χαρακτήρες, ενώ η προηγούμενα περιγραφόμενη μέθοδος κωδικοποίησης μήκους εκτέλεσης χρησιμοποιεί τρεις χαρακτήρες. Τέλος, η συμβατική κωδικοποίηση μήκους εκτέλεσης μπορεί να εφαρμοστεί για να συμπιέσει έως 256 επαναλαμβανόμενους χαρακτήρες, ενώ η MNP τάξης 5 περιορίζεται σε 250. Αυτά τα μειονεκτήματα πρέπει να ζυγιστούν ενάντια στο γεγονός ότι η εκδοχή MNP Τάξης 5 της κωδικοποίησης μήκους εκτέλεσης εξαφανίζει την αναγκαιότητα χρησιμοποίησης ενός ειδικού χαρακτήρα δείκτη συμπίεσης, αφού οι τρεις επαναλαμβανόμενοι χαρακτήρες μαζί δείχνουν ότι έχει εμφανιστεί συμπίεση και αναγνωρίζουν το χαρακτήρα που συμπιέστηκε.

Format



Where
X=repeated character
C=repetition count

Examples

Original data stream	Compressed data stream
A	A
BB	BB
CCC	0CCC
DDDD	4DDD
EEEE	6EEE

Σχήμα 2.18 Κωδικοποίηση μήκους εκτέλεσης MNP Τάξης 5. Αφού τρεις επαναλαμβανόμενοι χαρακτήρες δείξουν κωδικοποίηση μήκους εκτέλεσης, η μορφή συμπίεσης μήκους εκτέλεσης MNP Τειξης 5 δεν απαιτεί τη χρήση ενός ειδικού χαρακτήρα ένδειξης συμπίεσης.

2.3.6. Αποτελεσματικότητα

Η αποτελεσματικότητα της κωδικοποίησης μήκους εκτέλεσης εξαρτάται από τον αριθμό των εμφανίσεων των επαναλαμβανόμενων χαρακτήρων στα δεδομένα που πρόκειται να συμπιεστούν, το μέσο μήκος του επαναλαμβανόμενου χαρακτήρα και την τεχνική που εφαρμόστηκε για την συμπίεση. Ένας δεύτερος παράγοντας που επηρεάζει την αποτελεσματικότητα της κωδικοποίησης μήκους εκτέλεσης είναι ο μηχανισμός που χρησιμοποιείται για την υλοποίηση αυτής της τεχνικής συμπίεσης δεδομένων. Ο Πίνακας 3.5 δείχνει το λόγο συμπίεσης για τρεις

συνηθισμένες μεθόδους με τις οποίες εφαρμόζεται η κωδικοποίηση μήκους εκτέλεσης. Η πρώτη στήλη δείχνει το λόγο συμπίεσης όταν ένας ειδικός χαρακτήρας συμπίεσης ακολουθείται από το συμπιεσμένο χαρακτήρα και ένα μέτρημα, χρησιμοποιώντας μια σειρά τριών χαρακτήρων για να δηλώσει την εμφάνιση της συμπίεσης μήκους εκτέλεσης. Στη δεύτερη στήλη φαίνεται ο λόγος συμπίεσης που βγαίνει από τη χρήση ενός σετ εκτεταμένου κώδικα (extended code set - ECS). Στην κατάσταση αυτή οι χαρακτήρες SO και SI πρέπει να προστεθούν στην αρχή (SO) και στο τέλος (SI) της προηγούμενα περιγραφείσας αλληλουχίας τριών χαρακτήρων, με αποτέλεσμα να απαιτούνται πέντε χαρακτήρες για την κωδικοποίηση μιας αλληλουχίας επαναλαμβανόμενων χαρακτήρων. Η τρίτη στήλη, που ονομάζεται MNP Μήκος Εκτέλεσης, δείχνει το λόγο συμπίεσης όταν χρησιμοποιείται η μέθοδος MNP Τάξης 5 κωδικοποίησης μήκους εκτέλεσης. Κάτω από την τεχνική αυτή, αλληλουχίες τριών ή περισσότερων επαναλαμβανόμενων χαρακτήρων κωδικοποιούνται ως τέσσερις χαρακτήρες.

Όπως φαίνεται στον Πίνακα 3.5, η συμβατική μέθοδος της κωδικοποίησης μήκους εκτέλεσης παρέχει το μεγαλύτερο λόγο συμπίεσης. Αυτό συμβαίνει επειδή απαιτεί το μικρότερο αριθμό χαρακτήρων για να εφαρμοστεί. Η MNP Τάξης 5 είναι η επόμενη περισσότερο αποτελεσματική, καθώς απαιτεί τέσσερις χαρακτήρες, ενώ η χρήση ενός σετ εκτεταμένου χαρακτήρα είναι η λιγότερο αποτελεσματική μέθοδος για εφαρμογή της κωδικοποίησης μήκους εκτέλεσης.

Table 3.5 Efficiency of run-length encoding methods

Number of repeated characters	Compression ratio		
	Run length	ECS run length	MNP run length
3	1.0000	0.6000	0.7500
4	1.3333	0.8000	1.0000
5	1.6667	1.0000	1.2500
6	2.0000	1.2000	1.5000
7	2.3333	1.4000	1.7500
8	2.6667	1.6000	2.0000
9	3.0000	1.8000	2.2500
10	3.3333	2.0000	2.5000
11	3.6667	2.2000	2.7500
12	4.0000	2.4000	3.0000
13	4.3333	2.6000	3.2500
14	4.6667	2.8000	3.5000
15	5.0000	3.0000	3.7500
16	5.3333	3.2000	4.0000
17	5.6667	3.4000	4.2500
18	6.0000	3.6000	4.5000
19	6.3333	3.8000	4.7500
20	6.6667	4.0000	5.0000

Πίνακας 3.5 Αποτελεσματικότητα των μεθόδων κωδικοποίησης μήκους εκτέλεσης.

2.3.7. Παραδείγματα προγραμματισμού

Στην εσωτερική του λειτουργία, ο IBM PC χρησιμοποιεί έναν κώδικα ASCII των 8-bit εκτεταμένου χαρακτήρα. Αυτός ο κώδικας εκτεταμένου χαρακτήρα έχει ως αποτέλεσμα την εκχώρηση των ευκρινών χαρακτήρων σε τιμές ASCII από 128 έως 255. Αφού κάθε χαρακτήρας από την ASCII τιμή 0 έως 255 καθορίζεται και μπορεί να εμφανιστεί όταν μεταδίδονται δεδομένα από έναν IBM PC σε άλλο

υπολογιστικό σύστημα, μπορείτε κανονικά να εφαρμόσετε τους ASCII SO (shift out) και SI (shift in) χαρακτήρες στην ανάπτυξη ενός μέτρου συμπίεσης σχεδιασμένου να λειτουργήσει σε δεδομένα ASCII οποτεδήποτε υπάρχει πιθανότητα εμφάνισης για κάθε χαρακτήρα στο σετ χαρακτήρων.

Ο χαρακτήρας SO χρησιμοποιείται για να βγάλει έξω (shift out) το παρόν σετ χαρακτήρων ASCII, με αποτέλεσμα την ικανότητα του χρήστη να ανακαθορίσει κάθε χαρακτήρα στο σετ χαρακτήρων. Παρόμοια, ο χαρακτήρας SI χρησιμοποιείται για να ξαναεισάγει πίσω στο καθορισμένο σετ χαρακτήρων ASCII.

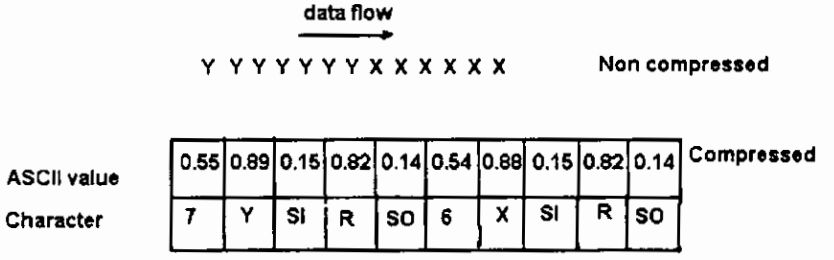
Με τη χρήση των χαρακτήρων SO και SI στην ASCII λαμβάνετε ένα σετ είτε 128 είτε 256 νέων χαρακτήρων, ανάλογα με το αν χρησιμοποιείτε ένα σύστημα που χρησιμοποιεί έναν 7-bit ή έναν εκτεταμένο 8-bit ASCII κώδικα. Αυτό το νέο σετ χαρακτήρων μπορεί κατόπιν να χρησιμοποιηθεί για να δηλώσει χαρακτήρες ένδειξης συμπίεσης. Το σχήμα 2.19 δείχνει τη χρησιμοποίηση των ASCII χαρακτήρων SO και SI για την εξαγωγή ενός νέου σετ χαρακτήρων όπου η τιμή ASCII 082 (συμβατικός ASCII R) χρησιμοποιείται για να δηλώσει κωδικοποίηση μήκους εκτέλεσης. Στο παράδειγμα αυτό, μια αλληλουχία από έξι X υποτέθηκε ότι ακολουθείται από μια αλληλουχία από εφτά Y. Αφού η ASCII τιμή 082 σ' έναν προσφάτως καθορισμένο ASCII κώδικα θα χρησιμοποιηθεί για να δείξει μια συμπίεση εκτέλεσης μήκους, πρέπει πρώτα να κάνετε shift out (SO) στο νέο κώδικα, να εκδώσετε το χαρακτήρα που δείχνει τη συμπίεση (R) και κατόπιν να ξανακάνετε shift (SI) στον κανονικό ASCII κώδικα για να μεταδώσετε το χαρακτήρα που συμπιέστηκε (X) και την ποσότητα των X χαρακτήρων που συμπιέστηκαν (6). Εξαιτίας της απαίτησης για shift out

από το σετ χαρακτήρων για να εκδοθεί ο χαρακτήρας ένδειξης συμπίεσης και κατόπιν για shift ξανά στο κανονικό σετ χαρακτήρων ASCII, δυο συμπληρωματικοί χαρακτήρες απαιτούνται για να δηλώσουν μια κωδικοποιημένη αλληλουχία μήκους εκτέλεσης. Επιπρόσθετα μ' αυτή την τεχνική που απαιτεί δυο επιπλέον χαρακτήρες, η χρήση ενός κώδικα shift out 14 χρησιμοποιείται για να ανοίξει η ρύθμιση της διάταξης διπλού βάθους των περισσότερων εκτυπωτών κουκκίδων (dot matrix), ενώ η θέση shift in 15 χρησιμοποιείται για να ανοίξει η ρύθμιση της διάταξης συμπιεσμένου χαρακτήρα τέτοιων εκτυπωτών, κάνοντας τη γραφική αναπαράσταση της τεχνικής αυτής κουραστική.

Βάσει των προηγούμενων πληροφοριών, καθορίστηκε ότι για τα παραδείγματα που παρουσιάζονται, η χρήση ενός απλού χαρακτήρα στο σέτ χαρακτήρων ASCII θα μπορούσε ικανοποιητικά να εξυπηρετήσει ως σημαία (δείκτης) συμπίεσης, με επιπρόσθετο ότι πραγματικά σώζονται δυο χαρακτήρες στην αναπαράσταση της συμπίεσης δεδομένων που βασίζεται στη χρήση της κωδικοποίησης μήκους εκτέλεσης.

2.3.8. Πρόγραμμα συμπίεσης

Το Σχήμα 2.20 περιέχει τον κατάλογο ενός προγράμματος BASIC που δείχνει τον κώδικα που απαιτείται για την πραγματοποίηση συμπίεσης μήκους εκτέλεσης.



Σχήμα 2.19 Χρησιμοποίηση χαρακτήρων SO και SI. Η χρήση των SO και SI παρέχει ένα νέο σετ χαρακτήρων που μπορεί να χρησιμοποιηθεί για να δείξει διαφορετικές τεχνικές συμπίεσης, σ' αυτο το παράδειγμα, η τιμή ASCII 082 χρησιμοποιείται για να δηλώσει κωδικοποίηση μήκους εκτέλεσης.

Στο πρόγραμμα RUNLENC.BAS, η τιμή ASCII 125 (δεξιά αγκύλη, }) χρησιμοποιήθηκε ως ο χαρακτήρας ένδειξης συμπίεσης, ο οποίος κατόπιν ακολουθήθηκε από το χαρακτήρα ASCII που συμπιέστηκε και την επαναληπτική μέτρησή του σε δεκαδική αρίθμηση. 'Ετσι, κάθε αλληλουχία που υπερβαίνει τους τρεις επαναλαμβανόμενους χαρακτήρες θα υπόκειται σε συμπίεση.

Διάφορες εντολές στον κατάλογο του προγράμματος που περιέχονται στο Σχήμα 2.20 δικαιολογούν συζήτηση για τους αναγνώστες εκείνους που δεν είναι εξοικειωμένοι με την εκδοχή BASICA IBM PS της προγραμματικής γλώσσας BASIC. Η εντολή LINE INPUT στη γραμμή 130 έχει ως αποτέλεσμα μια ολόκληρη γραμμή από έναν ακολουθητικό φάκελο να αναγιγνώσκεται και να τοποθετείται στη μεταβλητή X$ της αλληλουχίας. Στη γραμμή 140 καθορίζεται το μήκος της αλληλουχίας που αναπαριστάνει μια γραμμή στο φάκελο δεδομένων. Το μήκος της αλληλουχίας κατόπιν

χρησιμοποιείται στο βρόγχο FOR-NEXT που αναπηδά από τις γραμμές 240 μέχρι 310 για να επεξεργαστεί την αλληλουχία για επαναλαμβανόμενους χαρακτήρες. Οι λειτουργίες MID$ στις γραμμές 250 και 260 εξάγουν τους 10 και 10+1 χαρακτήρες από την αλληλουχία και συγκρίνουν τους χαρακτήρες αυτούς τον ένα με τον άλλο. Όταν είναι ίσοι, ο επαναλαμβανόμενος χαρακτήρας σώζεται (γραμμή 330) και το μέτρημα των επαναλαμβανόμενων χαρακτήρων προσαυξάνεται (γραμμή 340). 'Οταν η επαναλαμβανόμενη αλληλουχία χαρακτήρων σπάσει, η γραμμή 260 είναι FALSE (ψευδής) και μια προκύπτει μια σύγκριση της επαναλαμβανόμενης μέτρησης (γραμμές 270 και 280). Όταν η μέτρηση υπερβαίνει τα τρία (γραμμή 270) τα δεδομένα συμπιέζονται από τον κώδικα που περιέχεται στις γραμμές 360 μέχρι 400. Αν η μέτρηση ισούται με το τρία δεν υπάρχει πλεονέκτημα που μπορεί να κερδιθεί από την συμπίεση μήκους εκτέλεσης και η ρουτίνα που αναπηδούσε από τις γραμμές 420 μέχρι 450 απλά προσθέτει τους χαρακτήρες εισόδου στον καταχωρητή εξόδου. Όταν οι χαρακτήρες 1ος και 1ος+1 δεν είναι ίσοι, 0 1ος χαρακτήρας στον καταχωρητή εισόδου απλά τοποθετείται στον καταχωρητή εξόδου (γραμμή 290). Οι γραμμές 900 μέχρι 9999 δεν είναι πραγματικά τμήμα της διαδικασίας κωδικοποίησης μήκους εκτέλεσης και περιλαμβάνονται μόνο για να διευκολύνουν λειτουργίες του φακέλου και σύγκριση των καταχωρητών εισόδου και εξόδου ώστε να εξαχθεί ένα μέτρο της αποτελεσματικότητας της τεχνικής αυτής όταν εφαρμόζεται σε ένα φάκελο δεδομένων που περιέχει μια ποικιλία επαναλαμβανόμενων αλληλουχιών δεδομένων.

Το Σχήμα 2.21 δείχνει μια δειγματική εκτέλεση του προγράμματος RUNLENC.BAS που χρησιμοποιεί ένα φάκελο ASCII ονομαζόμενο

RUNLENC.DAT ως είσοδο στο πρόγραμμα. Σημειώστε ότι το RUNLENC.BAS επίτηδες έχει γραφτεί έτσι ώστε πρώτα να δίνει κατάλογο των περιεχομένων του φακέλου πριν τη συμπίεσή του, που φαίνεται στις γραμμές 1 έως 8 στο επάνω τμήμα του Σχήματος 2.21. Κατόπιν, το πρόγραμμα δίνει έναν κατάλογο του φακέλου αφού συμπιεστούν τα περιεχόμενά του, βασισμένο πάνω στην εφαρμογή της κωδικοποίησης μήκους εκτέλεσης στα δεδομένα που περιέχονται στο φάκελο.

Θα πρέπει, να σημειωθεί ότι δεκαδικές σειριακές τιμές που εκτείνονται κάτω από την ASCII 32 επίτηδες παραλείφθηκαν από τη συμπερίληψή τους στο δοκιμαστικό φάκελο,. αφού θα μπορούσαν να προκαλέσουν την εμφάνιση ανεπιθύμητων χαρακτήρων επιστροφής, τροφοδοτήσεων γραμμής και άλλων μη εκτυπώσιμων χαρακτήρων, κάτι που θα έκανε μια επίδειξη αυτής της τεχνικής συμπίεσης δύσκολο να κατανοηθεί. Μπορούν, ωστόσο, να είναι αρκετά κατάλληλες σε κανονική συμπίεση αλληλουχίας και εφαρμογές αποσυμπίεσης.

2.3.9. Πρόγραμμα γλώσσας C

Για να διευκολυνθεί η χρήση προγραμμάτων από τους αναγνώστες που προτιμούν να χρησιμοποιούν τη γλώσσα C, το RUNLENC.BAS ξαναγράφτηκε με τη χρήση της Borland's Turbo C. Το επακολουθούμενο πρόγραμμα, χρησιμοποιεί τις εντολές DEFINE για να εκχωρήσει την αλληλουχία 'RUNLENC.DAT' στη μεταβλητή 'infile'

```
10 REM RUNLENC.BAS PROGRAM
20 DIM O$(132)
30 WIDTH 80:CLS
40 '**********MAIN ROUTINE**********************
50 '* THIS ROUTINE READS RECORDS FROM AN ASCII *
60 '* FILE INTO A STRING CALLED X$ WHICH IS    *
70 '* THEN PASSED TO SUBROUTINES FOR COMPRESSION
80 '*************************************************
90 PRINT "ENTER ASCII FILENAME. EG, RUNLEN.DAT"
100 INPUT F$: OPEN F$ FOR INPUT AS #2
105 OPEN "RUNLENC.DAT" FOR OUTPUT AS #3
110 PRINT "PATIENCE - INPUT PROCESSING"
120 IF EOF(2) THEN GOTO 900
130 LINE INPUT #2, X$
140 N= LEN(X$)'
150 GOSUB 180
160 GOSUB 900
170 GOTO 120
180 '*********RUN LENGTH ENCODING SUBROUTINE*********
190 '* THIS ROUTINE PROCESSES RECORDS FROM X$   *
200 '* AND COMPRESSES OUT REPETITIVE CHARACTERS$
210 '* USING O$ AS THE OUTPUT BUFFER.          *
220 '*************************************************
230 K=1:J=1                          'RESET INDICES
240 FOR I= 1 TO N                    'STEP THRU RECORD
250 A$= MID$(X$, I, 1)               'EXTRACT A CHAR
260 IF A$= MID$(X$,I+1,1) THEN 330 'SAME AS NEXT?
270 IF K>3 THEN 360                  'COMPRESS
280 IF K=3 THEN 320                  'DON'T COMPRESS
290 O$(J)=A$                         'STUFF IN OUTPUT BUFFER
300 J=J+1                            'BUMP BUFFER INDEX
310 NEXT I                           'GO BACK FOR MORE
320 RETURN                           'END OF STRING
330 B$=A$                            'SAVE REPEATED CHAR
340 K=K+1                            'BUMP COUNT
350 GOTO 310                         'KEEP LOOKING
355 '*************************************************
360 'INSERT COMPRESSION NOTATION IN OUTPUT BUFFER
365 '*************************************************
370 O$(J)=CHR$(255)                 'SET FLAG FOR RUN-LENGTH
380 O$(J+1)=B$                       'INSERT REPEATED CHAR
390 O$(J+2)=CHR$(K)                  'INSERT COUNT
400 J=J+3:K=1                        'RESET INDEX
410 GOTO 310
420 O$(J)=B$                         'STUFF 1ST REPEAT CHAR
430 O$(J+1)=B$                       'STUFF 2ND REPEAT CHAR
440 J=J+2:K=1                        'RESET INDEX
450 GOTO 310
```

```
900 '******TALLY THE COMPRESSION COUNT & WRITE BUFFER******
910 '* DISPLAY BEFORE & AFTER RESULTS OF COMPRESSION      *
920 '* AND SHOW THE NET RESULTS OBTAINED BY EACH METHOD   *
930 '*************************************************************
931 N1=N1+N                              'TALLY INPUT CHAR COUNT
932 T=N-J+1                              'NET DIFFERENCE IN BUFFERS
936 T1=T1+T                              'SAVE COUNT FOR SUMMARY
940 FOR I= 1 TO J-1
950 PRINT #3, O$(I);
960 NEXT I
965 PRINT #3, ""
970 RETURN
9000 CLOSE: OPEN F$ FOR INPUT AS #2
9010 PRINT "FILE ";F$;" BEFORE COMPRESSION:"
9020 LINE INPUT #2,X$
9030 IF EOF(2) THEN 9060
9040 PRINT X$
9050 GOTO 9020
9060 PRINT X$:OPEN "RUNLENC.DAT" FOR INPUT AS #3
9070 PRINT "FILE ";F$;" AFTER COMPRESSION:"
9080 LINE INPUT #3,O$
9090 IF EOF(3) THEN 9998
9100 PRINT O$
9110 GOTO 9080
9998 PRINT O$:PRINT T1;" TOTAL CHARACTERS ELIMINATED FROM ";
9999 PRINT N1;"OR ";INT((T1/N1)*100);"%":CLOSE:END
```

Σχήμα 2.20 Κατάλογος προγράμματος RUNLENC.BAS

'infile' και την αλληλουχία 'RUNLENC.DAT' στη μεταβλητή 'outfile' για να παράσχει συνέπεια με τα προηγουμένως αναπτυγμένα προγράμματα γλώσσας C.

Το Σχήμα 2.22 δίνει έναν κατάλογο των εντολών στο πρόγραμμα RUNLEN.C. Σημειώστε ότι όταν χρησιμοποιούνταν πιθανές μεταβλητές ετικέτες, παρόμοιες μεταβλητές ετικέτες χρησιμοποιούνταν στο ισοδύναμο πρόγραμμα BASIC. Επιπρόσθετα, το πρόγραμμα γλώσσας C κάνει ελεύθερη χρήση των σχολίων προς διευκόλυνση κατανόησης της λειτουργίας του. Η εκτέλεση του RUNLENC.C θα δημιουργήσει παρόμοια αποτελέσματα με την εκτέλεση του

RUNLENC.BAS που φαίνεται στο Σχήμα 2.21. Η διαφορά μεταξύ της εκτέλεσης των RUNLENC.BAS και RUNLENC.C περιορίζεται στο μέτρημα των χαρακτήρων στο φάκελο. Το πρόγραμμα στη γλώσσα C μετρά την τροφοδότηση γραμμής και τους χαρακτήρες επαναφορέα κυλίνδρου εκτύπωσης σε καθε γραμμή, καθώς επίσης και το χαρακτήρα

```
ENTER ASCII FILENAME. EG. RUNLEN.DAT
? RUNLEN.DAT
PATIENCE - INPUT PROCESSING
FILE RUNLEN.DAT BEFORE COMPRESSION:
1 BEGIN**************************************
2 RRRRRRRRRRRRRRRRRRRRRRRRRRRRRRR
3 EEEEEEEEEEEEEEEEEEEEEEEEEEEEEEE
4 PPPPPPPPPPPPPPPPPPPPPPPPPPPPPPP
5 EEEEEEEEEEEEEEEEEEEEEEEEEEEEEEE
6 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
7 TTTTTTTTTTTTTTTTTTTTTTTTTTTTTTT
8 ****************************************END
FILE RUNLEN.DAT AFTER COMPRESSION:
1 BEGIN)**
2 )R!
3 )E"
4 )P#
5 )E$
6 )A%
7 )T&
8 )*%END
 261  TOTAL CHARACTERS ELIMINATED FROM  309 OR  84 %
Ok
```

Σχήμα 2.21 Δειγματική εκτέλεση του RUNLENC.BAS

τέλους φακέλου που τοποθετείται από το λειτουργικό σύστημα στο τέλος του φακέλου. 'Ετσι, ο φάκελος των οκτώ γραμμών που φαίνεται στο Σχήμα 2.21 όταν συμπιέζεται από το RUNLENC.C θα δείχνει '261 Συνολικούς Χαρακτήρες Περιορισμένους από 326' αντί για `309', αφού το πρόγραμμα BASICA δεν μετρά τις τροφοδοτήσεις γραμμών, τις επαναφορές κυλίνδρου εκτύπωσης και το χαρακτήρα τέλους φακέλου.

2.3.10. Μετατροπές προς θεώρηση

Ο χαρακτήρας ASCII 125 χρησιμοποιήθηκε ως χαρακτήρας ένδειξης συμπίεσης εξαιτίας της παράστασής του ως δεξιάς αγκύλης στους περισσότερους εκτυπωτές. Κανονικά, αν τα δεδομένα σου από την πηγή δεν περιλαμβάνουν χαρακτήρες πέρα από τον ASCII 127, τότε ένας χαρακτήρας στο εκτεταμένο σετ χαρακτήρων ASCII, όπως ο ASCII 129 ή άλλος πέρα από τον ASCII 127, θα χρησιμοποιηθεί για να δηλώσει την εμφάνιση της κωδικοποίησης μήκους εκτέλεσης. Για το προηγούμενο παράδειγμα, ο ASCII 129 επίτηδες εξαιρέθηκε, επειδή η εμφάνισή του σε μια οθόνη ως ο χαρακτήρας ii θα τυπωνόταν σε μερικούς εκτυπωτές ως ο χαρακτήρας £ (αγγλική λίρα), ενώ άλλοι εκτυπωτές απλώς αγνοούν χαρακτήρες πέρα από τον ASCII 127. Για τη σωστή εκτύπωση χαρακτήρων πέρα του ASCII 127 με τη χρήση ενός IBM PC απαιτείται να υπάρχει ένας εκτυπωτής ικανός να εκτυπώνει το εκτεταμένο σετ χαρακτήρων ASCII. Επιπρόσθετα, ένα πρόγραμμα ειδικού λειτουργικού συστήματος δίσκου (disc operating system - DOS) που ονομάζεται GrAFTABL, το οποίο είναι διαθέσιμο κάτω από το DOS 3.0 και νεώτερες εκδόσεις του λειτουργικού συστήματος, πρέπει να φορτωθεί στον υπολογιστή πριν τα προς εκτύπωση δεδομένα. Εξαιτίας αυτού, ο χαρακτήρας ASCII 125 χρησιμοποιήθηκε για σκοπούς επίδειξης ως ο χαρακτήρας ένδειξης συμπίεσης.

Αν ένας χαρακτήρας πέρα από τον ASCII 127 χρησιμοποιηθεί για να δείξει την εμφάνιση συμπίεσης και αυτός ο χαρακτήρας κανονικά παρουσιάζεται στα δεδομένα σας, θα έχει ως αποτέλεσμα μια ψευδή ένδειξη συμπίεσης. Για να εμποδίσετε μια συσκευή λήψης να ερμηνεύσει λάθος το χαρακτήρας ως ένδειξη ότι εμφανίστηκε συμπίεση μήκους εκτέλεσης, το πρόγραμμα μπορεί να τροποποιηθεί ώστε να στέλνει δυο τέτοιους χαρακτήρες οποτεδήποτε εμφανίζεται φυσιολογικά σ' ένα ρεύμα δεδομένων ένας χαρακτήρας ένδειξης συμπίεσης. Τότε, στη συσκευή λήψης το πρόγραμμα αποσυμπίεσης θα εξετάζει πρώτα κάθε χαρακτήρα για την εμφάνιση ενός χαρακτήρα ένδειξης συμπίεσης, όμως, όταν μετρηθεί δεν θα καθορίζει αμέσως ότι έχει εμφανιστεί κωδικοποίηση μήκους εκτέλεσης. Το πρόγραμμα κατόπιν θα εξετάζει τον επόμενο χαρακτήρα για να εξακριβώσει αν ο χαρακτήρας είναι επίσης ένας χαρακτήρας ένδειξης συμπίεσης. Αν είναι, αυτός θα εξυπηρετήσει ως ενδείκτης ότι εμφανίστηκε φυσιολογικά στα δεδομένα ένας χαρακτήρας ένδειξης συμπίεσης, με αποτέλεσμα την απομάκρυνση του δεύτερου χαρακτήρα ένδειξης συμπίεσης από τον λήπτη. Η τεχνική που μόλις περιγράφτηκε ονομάζεται περισσότερο τυπικά ως η μέθοδος εισόδου και απαλοιφής χαρακτήρα.

2.3.11. Αποσυμπίεση

Στο Σχήμα 2.23, οι αναγνώστες θα βρουν τον κατάλογο του προγράμματος RUNLEND.BAS, το οποίο είναι το πρόγραμμα που αναπτύχθηκε για την αποσυμπίεση δεδομένων που είχαν

προηγουμένως συμπιεστεί με το πρόγραμμα RUNLENC.BAS. Σε όσο το δυνατό μεγαλύτερο μέρος, οι μεταβλητές του προγράμματος και οι τρόποι κωδικοποίησης έχουν κρατηθεί οι ίδιοι μεταξύ των προγραμμάτων συμπίεσης και αποσυμπίεσης που παρουσιάζονται στο βιβλίο αυτό, για να διευκολυνθεί η χρήση τους και η επεξήγησή τους.

Παρόμοια με το προηγούμενα εξεταζόμενο πρόγραμμα συμπίεσης, το πρόγραμμα αυτό επεξεργάζεται τα δεδομένα σε βάση γραμμής με γραμμή. Η εντολή LINE INPUT στη γραμμή 130 διαβάζει μια γραμμή δεδομένων από το φάκελο που χρησιμοποιήθηκε για είσοδος. Κατόπιν, στη γραμμή 140 καθορίζεται το μήκος της γραμμής.

Κατόπιν τίθεται σε ενέργεια η υπορουτίνα που αναπηδούσε στις γραμμές 180 και 320. Στην υπορουτίνα αυτή η αλληλουχία που παριστάνει μια γραμμή από το φάκελο εισόδου εξετάζεται σε μια βάση από χαρακτήρα σε χαρακτήρα, με τη χρήση της λειτουργίας MID$ στη γραμμή 250 για την εξαγωγή ενός χαρακτήρα τη φορά από την αλληλουχία. Στη γραμμή 260, κάθε εξαγόμενος χαρακτήρας συγκρίνεται με την τιμή χαρακτήρα 125 που είναι ο χαρακτήρας δεξιάς αγκύλης, για να καθοριστεί αν εμφανίστηκε ένας χαρακτήρας ένδειξης συμπίεσης. Αν έχει εμφανιστεί, εμφανίζεται ένας κλάδος στη γραμμή 360 όπου το επαναλαμβανόμενο μέτρημα και ο επαναλαμβανόμενος χαρακτήρας εξάγονται από την αλληλουχία στις γραμμές 370 και 380. Κατόπιν, εξάγεται ένας αριθμοδείκτης (κατάλογος) βασισμένος στην αριθμητική αξία του K$, με τη χρήση της λειτουργίας ASC στη γραμμή 390. Αυτό ακολουθείται από τον βρόγχο FOR-NEXT που αναπηδά στις γραμμές από 400 έως 420, που τοποθετεί τον επαναλαμβανόμενο χαρακτήρα στον καταχωρητή εξόδου τον απαιτούμενο αριθμό φορών για να ταιριάξει με το χαρακτήρα μέτρησης.

Κατόπιν οι αριθμοδείκτες J και I αυξάνονται και το πρόγραμμα διακλαδώνεται πάλι στη γραμμή 250.

Αν δεν εμφανιστεί στα δεδομένα ένας χαρακτήρας ένδειξης συμπίεσης, εκτελείται η γραμμή 290. Η γραμμή αυτή προκαλεί την αφαίρεση του χαρακτήρα από την αλληλουχία και την μετάθεσή του κατευθείαν στον καταχωρητή εξόδου. Κατόπιν, ο αριθμοδείκτης J αυξάνεται κατά 1 στη γραμμή 300 και το όριο του αρχικού FOR-NEXT-βρόγχου ελέγχει για να καθοριστεί αν το τέλος του βρόγχου έφτασε στη γραμμή 310.

Οι εντολές από τη γραμμή 900 έως το τέλος του προγράμματος περιελήφθησαν για να καταμετρήσουν το μέτρημα της αποσυμπίεσης και να δείξουν τα αποτελέσματα, πριν και μετά του προγράμματος. Έτσι, αυτό το τμήμα του προγράμματος περιελήφθηκε μόνο για σκοπούς επίδειξης.

Το σχήμα 2.24 περιέχει μια δειγματική εκτέλεση του προγράμματος RUNLEND.BAS. Θα σημειώσετε ότι ο φάκελος δεδομένων RUNLENC.DAT χρησιμοποιήθηκε ως είσοδος στο πρόγραμμα. Αυτός ο φάκελος δεδομένων δημιουργήθηκε από την εκτέλεση του προγράμματος RUNLENC.BAS και οι πρώτες οκτώ αριθμημένες γραμμές στο Σχήμα 2.24 ανταποκρίνονται στις οκτώ χαμηλότερες αριθμημένες γραμμές του Σχήματος 2.21.

Αφού το πρόγραμμα αποσυμπίεσης επιστρέφει τα συμπιεσμένα δεδομένα στην αρχική μορφή τους, οι οκτώ αριθμημένες γραμμές στο κάτω μέρος του Σχήματος 2.24 είναι ακριβώς ίδιες με τις οκτώ αριθμημένες γραμμές στην κορυφή του Σχήματος 2.21.

```
/* RUNLENC.C - Program to demonstrate  run length compression                */

#include <stdio.h>
#define infile "RUNLEN.DAT"
#define outfile "RUNLENC.DAT"

char  o[256],                  /* output buffer                               */
      filename[13],            /* store filename                              */
      buffer[256];             /* input buffer - max string length            */

int   t=0,                     /* initialize counters                         */
      t1=0,
      n=0,
      n1=0;


FILE  *input,
      *output;

main()
{
   fileop();
   while( fgets(buffer,256,input) != NULL )
   {
      n = strlen(buffer);                    /* get length of string         */
      tally( runlength() );
   }
   summary();
   return 0;
}

runlength()
{
   int    i=0                              /* set indices                     */
          j=0,
          k=1;

   unsigned char   a,
                   b;

   for(i=0;i < n; i++)                     /*step through record              */
   {
      a = buffer[i];                       /* extract a character             */

      if (a == buffer[i+1] )               /* same as next?                   */
      {
         b=a;
         k++;
         continue;
      }
      if  ( k > 3)                         /* have run length compression     */
      {
         o[j] = 125;                       /* set flag for run length         */
         o[j+1] = b;;                      /* insert repeated character       */
         O[j+2] = k;                       /* insert count                    */
         j+=3;
         k=1;                              /* reset count                     */
      }
      else
      {
         if ( k== 3)                       /* when only three repeating       */
         {
            o[j]   = b;
            o[j+1] = b;
            j+=2;
            k=1;
         }
         else                              /* next character different        */
         {
            o[j] = a;
            j++;
         }
      }
```

```
    }
    return(j);
}

tally(j)                                    /* tally the compression count   */
int   j;
{
    int   i=0;

    nl=nl+n;
    t=n-j;
    tl=tl+t;
    for(i=0; i<j; i++)
    {
       fputc(o[i],output);
    }
    return 0;
}

fileop()
{
      printf("Enter ASCII Filename.Eg, %s  :", infile);
      scanf("%13s",filename);
      if ( (input  = fopen(filename,"rb")) == NULL )
       {
       printf("ERROR: Cannot open the file: %s \n",filename);
       exit(-1);
       }
       if ( (output = fopen(outfile,"wb")) == NULL )
       }
       printf("ERROR: Cannot open the file %s for output\n",outfile);
       exit(-2);
       }
       return 0;
}

summary()
{
   fclose(input);
   fclose(output);

   if ( (input  = fopen(filename,"rb")) == NULL )
   {
      printf("ERROR: Cannot open the file %s for input\n",filename);
      exit(-1);
   }
   printf("File  %s Before Compression:\n",filename);
   while( fgets(buffer,256,input) != NULL )
   {
      printf("%s",buffer);
   }

   fclose(input);

   if ( (input = fopen("runlenc.dat","rb")) == NULL )
   {
      printf("ERROR: Cannot open the file %s for output\n",outfile);
      exit(-2);
   }

   printf("\nFile After Compression :\n");

   while( fgets(buffer,256,input) != NULL )
   {
      printf("%s",buffer);
   }

   fclose(input);

   printf("\n%d Total Characters Eliminated From %d Or %d%%\n\n",
   tl,nl,(int)((float)((float)tl/(float)nl)*100));
   return 0;
}
```

Σχήμα 2.22 Κατάλογος προγράμματος RUNLENC.C

```
10 REM RUNLEND.BAS PROGRAM
20 DIM O$(132)
30 WIDTH 80:CLS
40 '***********MAIN ROUTINE**********************
50 '* THIS ROUTINE READS RECORDS FROM AN ASCII *
60 '* FILE INTO A STRING CALLED X$ WHICH IS    *
70 '* THEN PASSED TO DECOMPRESSION SUBROUTINE  *
80 '*********************************************
90 PRINT "ENTER ASCII FILENAME. EG. RUNLENC.DAT"
100 INPUT F$: OPEN F$ FOR INPUT AS #2
105 OPEN "RUNLEND.DAT" FOR OUTPUT AS #3
110 PRINT "PATIENCE - INPUT PROCESSING"
120 IF EOF(2) THEN GOTO 9000
130 LINE INPUT #2, X$
140 N= LEN(X$)
150 GOSUB 180
160 GOSUB 900
170 GOTO 120
180 '*****RUN LENGTH DECODING SUBROUTINE********
190 '* THIS ROUTINE PROCESSES RECORDS FROM X$  *
200 '* AND DECOMPRESSES RUN-ENCODED CHARACTERS *
210 '* USING O$ AS THE OUTPUT BUFFER.          *
220 '*********************************************
230 K=1:J=1                          'RESET INDICES
240 FOR I= 1 TO N                    'STEP THRU RECORD
250 A$= MID$(X$,I,1)                 'EXTRACT A CHAR
260 IF A$= CHR$(125) THEN 360        'COMPRESSION FLAG?
290 O$(J)=A$                         'STUFF IN OUTPUT BUFFER
300 J=J+1                            'BUMP BUFFER INDEX
310 NEXT I                           'GO BACK FOR MORE
320 RETURN                           'END OF STRING
355 '*********************************************************
360 'DECODE COMPRESSION NOTATION TO OUTPUT BUFFER
365 '*********************************************************
370 K$= MID$(X$,I+2,1)               'GET REPEAT COUNT
380 A$= MID$(X$,I+1,1)               'GET REPEAT CHAR
390 K= ASC(K$)                       'SET UP INDEX
400 FOR L= J TO J+K                  'SET OUTPUT LOOP
410 O$(L)= A$                        'STUFF REPEAT CHAR
420 NEXT L                           'KEEP GOING
430 J= L                             'BUMP OUTPUT INDEX
440 I= I+3                           'BUMP INPUT INDEX
450 GOTO 250                         'DONE
```

Σχήμα 2.23 Κατάλογος προγράμματος RUNLEND.BAS.

```
900 '*****TALLY THE DECOMPRESSION COUNT & WRITE BUFFER*****
910 '* DISPLAY BEFORE & AFTER RESULTS OF DECOMPRESSION    *
920 '* AND SHOW THE NET RESULTS OBTAINED BY EACH METHOD   *
930 '*************************************************************
931 N1=N1+N                          'TALLY INPUT CHAR COUNT
932 T=N-J+1                          'NET DIFFERENCE IN BUFFERS
936 T1=T1-T                          'SAVE COUNT FOR SUMMARY
940 FOR I= 1 TO J-1
950 PRINT #3, O$(I);
960 NEXT I
965 PRINT #3, ""
970 RETURN
9000 CLOSE: OPEN F$ FOR INPUT AS #2
9010 PRINT "FILE ";F$;" BEFORE DECOMPRESSION:"
9020 LINE INPUT #2,X$
9030 IF EOF(2) THEN 9060
9040 PRINT X$
9050 GOTO 9020
9060 PRINT X$:OPEN "BYTED.DAT" FOR INPUT AS #3
9070 PRINT "FILE ";F$;" AFTER DECOMPRESSION:"
9080 LINE INPUT #3,O$
9090 IF EOF(3) THEN 9998
9100 PRINT O$
9110 GOTO 9080
9998 PRINT O$:PRINT T1;" TOTAL CHARACTERS INSERTED"
9999 CLOSE:END
```

Σχήμα 2.23 συνέχεια

2.3.12. Πρόγραμμα RUNLEND.C

Βασισμένο πάνω στις προηγουμένως αναφερθείσες συνθήκες ονομασίας, το RUNLEND.C δηλώνει το πρόγραμμα γλώσσας C που δημιουργήθηκε για να δείξει την αποσυμπίεση μήκους εκτέλεσης. Αφού οι δηλώσεις του προγράμματος στο RUNLEND.C μοιάζουν πολύ με τις εντολές που περιέχονται στο RUNLENC.C, θα περιορίσουμε την εξέτασή μας του κώδικα του προγράμματος στις βασικές λειτουργίες του

```
ENTER ASCII FILENAME. EG. RUNLENC.DAT
? RUNLENC.DAT
PATIENCE - INPUT PROCESSING
FILE RUNLENC.DAT BEFORE DECOMPRESSION:
1 BEGIN)**
2 )R!
3 )E"
4 )P#
5 )E$
6 )A%
7 )T&
8 )*%END
FILE RUNLENC.DAT AFTER DECOMPRESSION:
1 BEGIN**********************************************
2 RRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRR
3 EEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEE
4 PPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPP
5 EEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEE
6 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
7 TTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTT
8 **********************************************END
 276  TOTAL CHARACTERS INSERTED
Ok
```

Σχήμα 2.24 Δειγματική εκτέλεση του προγράμματος RUNLEND.BAS.

'main' (κύρια), 'runlength' (μήκος εκτέλεσης) και 'tally' (απαρίθμηση, καταμέτρηση), των οποίων οι εντολές βρίσκονται στο Σχήμα 2.25.

Χρησιμοποιώντας τις προηγούμενες συνθήκες, οι δυο καθοριστικές εντολές στην επικεφαλίδα του προγράμματος, που δεν φαίνονται στο Σχήμα 2.25, εκχωρούν την αλληλουχία 'RUNLENC.DAT' στη μεταβλητή 'infile'. και την αλληλουχία 'RUNLEND.DAT' στη μεταβλητή 'outfile'. Η λειτουργία 'runlength' εκτελεί την πραγματική αποσυμπίεση. Η εντολή 'FOR I' επαναλαμβάνει κατά κύκλους κάθε χαρακτήρα στον καταχωρητή, ψάχνοντας για την εμφάνιση του χαρακτήρα ASCII του οποίου η τιμή είναι 125. Όταν εμφανιστεί ένα ταίριασμα, η θέση του επόμενου χαρακτήρα αναγνωρίζεται ως ο

χαρακτήρας συμπίεσης και εκχωρείται στη μεταβλητή 'α', ενώ ο χαρακτήρας που μετατοπίζεται κατά δυο θέσεις στον καταχωρητή δηλώνει το μέτρημα χαρακτήρων και εκχωρείται στη μεταβλητή 'k'. κατόπιν, ο βρόγχος 'for 1' τοποθετεί τις εμφανίσεις 'k' του συμπιεσμένου χαρακτήρα στις κατάλληλες θέσεις στον καταχωρητή εξόδου μέσω της χρήσης της εντολής 'o[j++]=α;'. Αφού η εμφάνιση του ASCII 125 έχει ως αποτέλεσμα τη χρήση των δυο επόμενων χαρακτήρων στο ρεύμα δεδομένων, η εντολή 'I+=2;' εκτοπίζει τον αριθμοκώδικα (index) τη θέση της παράταξης του καταχωρητή κατά 2. Το υπόλοιπο του κώδικα στη λειτουργία 'tally' δηλώνει κωδικοποίηση κατά πολύ παρόμοια με την κωδικοποίηση που χρησιμοποιείται στο RUNLENC.C και πρέπει να είναι αυτοεπεξηγηματική σε άτομα με ένα περιορισμένο υπόβαθρο στη C. Έτσι, θα παραλείψουμε τη συζήτηση αυτής της λειτουργίας του προγράμματος.

Η εκτέλεση του RUNLEND.C με τη χρήση του φακέλου RUNLENC.DAT για είσοδο παράγει το ίδιο αποτέλεσμα όπως η χρήση του RUNLEND.BAS. Έτσι, οι αναγνώστες που εκτελούν το RUNLEND.C θα λάβουν μια εμφάνιση ισοδύναμη με .αυτή που φαίνεται στο σχήμα 2.23.

```
main()
{
   fileop();

   while( fgets(buffer,256,input) != NULL )
   {
      n = strlen(buffer);
      tally( runlength() );
   }
   summary();
   return 0;
}

runlength()
{
   int   i=0,
         j=0,
         k=0,
         l=0;

   unsigned  char  a,
                   b;

   for(i=0; i < n; i++)
   {
      if ( buffer[i] == 125 )
      {
         a = buffer[i+1];
         k = buffer[i+2];

         for (l=0; l<k; l++)
         {
            o[j++] = a;
         }
         i+=2;
      }
      else
      {
         o[j++] = buffer[i];
      }
   }
   return(j);
}

tally(j)
int   j;
{
   int   i=0;

   nl=nl+n;
   t=n-j;
   tl=tl-t;
   for(i=0; i<j; i++)
   {
      fputc(o[i],output);
   }
   return 0;
}
```

Σχήμα 2.25 Οι λειτουργίες 'main', 'runlength' και 'tally' του RUNLEND.C

3. Συμπίεση εικόνας

3.1. Περίληψη γραφικών

Οι διατάξεις γραφικών φακέλων βασίζονται στη μέθοδο με την οποία τα προγράμματα γραφικών δημιουργούν, αποθηκεύουν και επιδεικνύουν εικόνες. Τα προγράμματα γραφικών και οι διατάξεις φακέλων που απορρέουν από αυτά μπορούν να διαχωριστούν σε μια από τις δυο κατηγορίες - κοκκιδοπλαίσιο (raster) και διάνυσμα (vector).

3.1.1. Προγράμματα raster

Ένα πρόγραμμα διάταξης raster εργάζεται με μια σειρά στοιχείων εικόνας, που ονομάζονται pels ή pixels, και η διάταξη φακέλου που απορρέει από αυτό συνήθως αναφέρεται ως χαρτογράφηση κουκίδων (bitmap). Ένα πρόγραμμα διάταξης raster χωρίζει την περιοχή της εικόνας σε πολύ μικρά σημεία, συνήθως 1/300 μιας τετραγωνικής ίντσας, και αποθηκεύει τα δεδομένα για κάθε σημείο. Έτσι, μια

τετραγωνική ίντσα σε μια ανάλυση 300 κουκίδων ανά ίντσα (dot per inch / dpi) θα απαιτεί 90.000 σημεία.

Κάθε σημείο σε μια εικόνα χαρτογραφημένων κουκίδων ή raster μπορεί να έχει δυο ή περισσότερες καταστάσεις. Αν η εικόνα είναι ασπρόμαυρη, κάθε σημείο μπορεί να αναπαρασταθεί με ένα δυαδικό ψηφίο (bit). Αν η εικόνα είναι σε κλίμακα του γκρίζου ή έγχρωμη, θα απαιτηθούν δύο ή περισσότερα bits για την αναπαράσταση κάθε σημείου.

Κοινές εικόνες της κλίμακας του γκρίζου έχουν είτε 16 είτε 256 αποχρώσεις του γκρίζου, που απαιτούν είτε τέσσερα είτε οκτώ bits για να αναπαραστήσουν την πιθανή γκρι απόχρωση κάθε σημείου. Οι έγχρωμες εικόνες μπορούν να εκτείνονται από 16 χρώματα, ή τέσσερα bits ανά σημείο, μέχρι 16.7 εκατομμύρια χρώματα, που απαιτούν 24 bits ανά σημείο.

Ο αριθμός των δυαδικών ψηφίων που χρησιμοποιείται για να αναπαραστήσει την κλίμακα του γκρίζου ή το χρώμα ενός σημείου συνήθως αναφέρεται ως βάθος χρώματος μιας εικόνας. Παρόλο που περισσότερα από 24 bits μπορούν να χρησιμοποιηθούν για να αναπαραστήσουν το χρώμα κάθε σημείου, υπάρχει ένα πρακτικό μέγιστο όριο όπου οποιαδήποτε ακτίνα χρωμάτων πέρα από αυτή που απαιτείται με τη χρήση των 24 bits συνήθως δεν μπορεί να ανιχνευτεί από το ανθρώπινο μάτι.

3.1.2. Προγράμματα ανύσματος (vector)

'Ενας δεύτερος τύπος προγράμματος γραφικής εικόνας και διάταξης φακέλων που απορρέουν από αυτό είναι τα προγράμματα ανύσματος που παράγουν φακέλους ανύσματος. Ένα πρόγραμμα γραφικών vector παράγει σχήματα που δημιουργούνται από γραμμικά τμήματα. Παραδείγματα προγραμμάτων γραφικών vector περιλαμβάνουν το σχεδιασμό με τη βοήθεια υπολογιστή (computer aided design / CAD) και προγράμματα δημιουργίας χαρτών και χειρισμών.

3.1.3. Raster εναντίον vector

Οι εικόνες raster είναι ανεξάρτητες σχήματος και επιτρέπουν την είσοδο, χειρισμό και έξοδο των εικόνων που θα μπορούσε να είναι δύσκολο να γίνουν, αν όχι αδύνατο, με τη χρήση προγράμματος εικόνας vector. Για το λόγο αυτό σαρωτές ειδώλων (scanners), ψηφιακές κάμερες και ψηφιοποιητικές παρεμβάσεις (digitizer pads) παρέχουν είσοδο εικόνας με χαρτογράφηση κουκίδων ή raster. Η έξοδος των προγραμμάτων εικόνας raster μπορεί να επιτευχθεί ευκολότερα και γρηγορότερα σε ένα monitor υπολογιστή ή σ' έναν εκτυπωτή γραφικών απ' όσο ένα πρόγραμμα εικόνας vector, εφόσον δεν είναι απαραίτητη

καμία μετατροπή από vector σε raster. Στο κεφάλαιο αυτό θα επικεντρώσουμε την προσοχή μας πάνω στις εικόνες raster.

3.1.4. Αξία της συμπίεσης εικόνας.

Για να έχετε μια εκτίμηση για την αξία της συμπίεσης εικόνας, σκεφτείτε τις απαιτήσεις αποθήκευσης μιας έγχρωμης εικόνας 3" X 5" που σαρώνεται (σκανάρεται) με τη χρήση ενός scanner ικανού να αναγνωρίζει 56 χρώματα στα 300 dpi. Η έγχρωμη εικόνα των 3" X 5" απαρτίζεται από 15 τετραγωνικές ίντσες, με 90.000 σημεία ανα ίντσα. 'Ετσι, 90.000 X 15 ή 1.350.000 bits απαιτούνται για την αναπαράσταση της εικόνας, χωρίς να λαμβάνεται υπόψη το βάθος χρωμάτων της. Από τη στιγμή που οκτώ bits (ή ένα byte) απαιτούνται για να εξαχθεί ένα βάθος χρώματος των 256, αυτό έχει ως αποτέλεσμα μια ελάχιστη απαίτηση αποθήκευσης δεδομένων από 1.35 Mbytes για την έγχρωμη εικόνα των 3" X 5" που αναφέρθηκε προηγουμένως. Ο λόγος που έχει χρησιμοποιηθεί ο όρος «ελάχιστη» είναι ότι ο φάκελος της εικόνας πρέπει να περιέχει μια επικεφαλίδα που να δίνει τη δυνατότητα στο πρόγραμμα να δηλώσει τον τύπο της αποθηκευμένης εικόνας καθώς επίσης και πληροφορίες σχετικά με το μέγεθός της, την ανάλυσή της και το βάθος χρωμάτων της. Έτσι, ο πραγματικός φάκελος θα απαιτούσε μερικά επιπρόσθετα bytes αποθήκευσης, παρόλο που η επιπρόσθετη αποθήκευση θα ήταν σχετικά μικρή σε σύγκριση με το συνολικό ποσό αποθήκευσης που απαιτείται. Χωρίς συμπίεση

μπορούμε να αποθηκεύσουμε μόνο μία εικόνα 3" X 5" σε μια δισκέτα των 3 '/2 ιντσών των 1.44 Mbyte. Αν κάποιος αρκετά τυχερός παλαιότερα είχε ένα σκληρό δίσκο των 200 Mbyte, η αποθήκευση λιγότερων από 150 εικόνων 3" X 5" θα απαιτούσε ολόκληρη την ικανότητα αποθήκευσης του δίσκου του. Παρόμοια, ο χρόνος μετάδοσης που απαιτείται για να σταλούν ή να ληφθούν μη-συμπιεσμένες εικόνες μπορεί να είναι πολύ μακρύς. Για παράδειγμα, στα 9600 bps (1200 χαρακτήρες ή bytes ανά δευτερόλεπτο) η έγχρωμη εικόνα των 3" X 5" θα απαιτούσε 1125 δευτερόλεπτα ή πάνω από 18 λεπτά για να σταλεί ή να ληφθεί. Για τους λόγους αυτούς, η συμπίεση μπορεί να θεωρηθεί ως αναγκαιότητα όταν δουλεύουμε με εικόνες. Ευτυχώς, οι διατάξεις φακέλων που σχετίζονται με πολλά δημοφιλή γραφικά εικόνων καθορίζουν τη χρήση ενός ή περισσοτέρων τύπων συμπίεσης που επιτρέπουν την αποθήκευση και μετάδοση πολλών τύπων εικόνων σε μια μορφή συμπίεσης.

Ο πίνακας 3.1 καταγράφει σε λίστα εννέα δημοφιλείς τύπους φακέλων εικόνων, την επέκτασή τους φακέλων και την υποστήριξη χρώματος.

Description	File extension	Colors
OS/2 bit map	BMP	B&W/Color
Windows bit map	BMP	B&W/Color
Windows Clipboard	CLP	B&W/Color
Paint Brush	PCX/PCC	B&W/Color
GEM environment	IMG	B&W/Color
Dr. Halo	PIC	Color
Graphics Interchange Format	GIF	B&W/Grayscale/Color
Tag Image File Format	TIF	B&W/Grayscale/Color
Joint Photographic Experts Group	JPG	Grayscale/Color

Πίνακας 3.1 Δημοφιλή φορμάτ αρχείων εικόνας.

3.2. Τεχνικές συμπίεσης εικόνας

Μια ευρεία ποικιλία τεχνικών συμπίεσης έχουν αναπτυχθεί τα τελευταία 20 χρόνια, για να μειωθεί το μέγεθος των φακέλων εικόνων χαρτογραφημένων κουκίδων. Μερικές τεχνικές παρουσιάζουν την εφαρμογή των τροποποιημένων αλγορίθμων συμπίεσης χωρίς απώλειες, ενώ άλλες τεχνικές παρουσιάζουν την εφαρμογή των αλγορίθμων που προκαλούν απώλειες.

Μπορούμε να ταξινομήσουμε τις τεχνικές συμπίεσης εικόνας σε ένα ελάχιστο πέντε γενικών κατηγοριών που φαίνονται στον Πίνακα 3.2.

Τεχνικές συμπίεσης εικόνων

Με απώλειες	Χωρίς απώλειες
Βάσει χαρακτήρων	Δυαδικά γραφικά
Στατιστικές	Μοντέλο αντικειμένου
Βασισμένη σε λεξικό	

Πίνακας 3.2

Θα πρέπει να σημειώσουμε ότι αυτό το ταξινομικό σχήμα δεν είναι μοναδικό, άλλα σχήματα μπορούν να αναπτυχθούν για την ομαδοποίηση τεχνικών συμπίεσης που εφαρμόζονται σε εικόνες. Στην εξέταση των καταχωρήσεων του Πίνακα 3.2, σημειώστε ότι οι

κατηγορίες υποδιαιρούνται βάσει του εάν παρέχουν μια πλήρως ανακτώμενη (χωρίς απώλειες) εικόνα ή όχι.

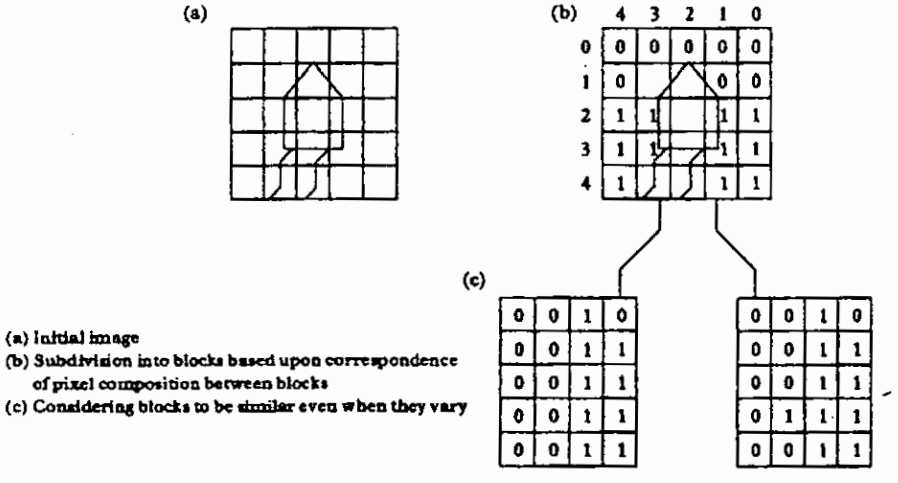
3.2.1. Βάσει χαρακτήρων

Οι τεχνικές συμπίεσης που βασίζονται στους χαρακτήρες λειτουργούν βάσει της ομαδοποίησης των pixels ως μια ολότητα byte. Έτσι, τα συμπαγή φόντα στα οποία τα μοντέλα των pixels επαναλαμβάνονται για κάμποσες ομαδοποιήσεις bytes θα πρέπει να είναι κατάλληλα για τη μείωση από μια τεχνική συμπίεσης βασισμένη στους χαρακτήρες.

3.2.2. Στατιστικές

Το Σχήμα 3.1 α δείχνει μια φτωχά σχεδιασμένη εικόνα ενός σπιτιού που υποτίθεται ότι βρίσκεται σε μια περιοχή όπου το περισσότερο από το φόντο παρουσιάζει είτε γαλάζιο ουρανό ή πράσινη γη, εκτός από ένα δρόμο που οδηγεί στο σπίτι. Υποθέστε ότι τα μηδέν χρησιμοποιούνται για να δηλώσουν το φόντο του ομοιόμορφου ουρανού που περιλαμβάνεται μέσα σε ένα μπλοκ από pixels, ενώ τα 1 χρησιμοποιούνται για να παρουσιάσουν ένα μπλοκ από pixels που καλύπτουν ένα ομοιόμορφο πράσινο γρασίδι. Κατόπιν, το Σχήμα

3.1 b μπορεί να θεωρηθεί ότι αναπαριστάνει την υποδιαίρεση της εικόνας σε μπλοκ από pixels που έχουν τα ίδια ή παρόμοια



Σχήμα 3.1

χαρακτηριστικά στα pixel. Αν τα μπλοκ έχουν τα ίδια χαρακτηριστικά στα pixel, με κάθε pixel ακριβώς το ίδιο με τα άλλα για να συμπεριληφθεί η αλληλουχία βάθους χρώματός τους, κατόπιν η επαναλαμβανόμενη αλληλουχία μπορεί να μην εμφανιστεί βασισμένη σε bytes. Αντίθετα, ομάδες από bytes που να περιλαμβάνουν το βάθος χρώματος των pixels μπορεί να επαναληφθεί για μια αλληλουχία. Έτσι, μια μέθοδος συμπίεσης βασισμένη στη στατιστική ή στην εντροπία, όπως η κωδικοποίηση Huftman ή η αριθμητική κωδικοποίηση, θα μπορούσε να εφαρμόστεί στην εικόνα και θα ήταν περισσότερο από πιθανό ότι θα παρήγαγε καλύτερα αποτελέσματα.

3.2.3. Κωδικοποίηση βασισμένη σε λεξικό

Από τη στιγμή που οι περισσότερες εικόνες απαρτίζονται από επαναλαμβανόμενες αλληλουχίες pixel, μια μέθοδος κωδικοποίησης βασισμένη σε λεξικό, όπως οποιοσδήποτε από τους Lempel-Ziv αλγόριθμους ή παράγωγα, μπορεί να εφαρμοστεί σ' ένα φάκελο εικόνας. Στην πραγματικότητα, μια από τις πιο δημοφιλείς διατάξεις φακέλου εικόνας, η προδιαγραφή CompuServe's GIF, χρησιμοποιεί μια τροποποιημένη φόρμα της LZW συμπίεσης για τη μείωση του μεγέθους της αποθηκευμένης εικόνας. Στην εφαρμογή της LZW συμπίεσης η προδιαγραφή GIF πρώτα προκαλεί το διαχωρισμό του βάθους χρώματος από τις τιμές pixel που παριστάνουν την εικόνα, πριν αρχίζει να εφαρμόζει τη συμπίεση στα pixel. Αργότερα στο κεφάλαιο αυτό θα εξετάσουμε την προδιαγραφή GIF με λεπτομέρεια.

3.2.4. Δυαδικά γραφικά

Υπάρχει ένας μεγάλος αριθμός τεχνικών συμπίεσης που έχουν ειδικά αναπτυχθεί για να λειτουργήσουν με εικόνες. Μερικές από τις τεχνικές αυτές, όπως ο τροποποιημένος κώδικας Huftman CCITT Group 3, η κωδικοποίηση μήκους εκτέλεσης και η κωδικοποίηση σχετικής διεύθυνσης, μπορούν να θεωρηθούν ότι παρουσιάζουν

τεχνικές συμπίεσης δυαδικών γραφικών χωρίς απώλειες. Άλλες τεχνικές συμπίεσης που εμπίπτουν στην κατηγορία δυαδικών γραφικών έχουν απώλειες. Παραδείγματα των τελευταίων περιλαμβάνουν την προφητική κωδικοποίηση, τη Προσαρμόσιμη Συμπίεση Εικόνας Διπλού Επιπέδου (Adaptive Bilevel Image Compression - ABIC) και την τεχνική συμπίεσης εικόνας της Ομάδας Ενωμένων Φωτογραφικών Ειδικών (Joint Photographic Experts Group - JPEG).

3.2.5. Προφητική κωδικοποίηση

Η προφητική κωδικοποίηση χρησιμοποιεί προηγούμενα κωδικοποιημένα τμήματα μιας εικόνας για να προβλέψει τη σύνθεση των μελλοντικών τμημάτων, με αποτέλεσμα μια συμπίεση με απώλειες. Το σχήμα αυτό περιγράφτηκε για πρώτη φορά το 1976 (Preuss, 1976) και εφαρμόστηκε σε φαξ που κωδικοποιούσαν δυαδικά δεδομένα.

Κάτω από την προφητική κωδικοποίηση, τα δεδομένα σαρώνονται σε διάταξη raster και σε κάθε θέση pixel γίνεται μια πρόβλεψη, βασισμένη στη σύνθεση των γειτνιαζόντων προηγούμενα επεξεργασμένων pixels. Αν η πρόβλεψη μπορεί να γίνει «τέλεια», όπως σε μια περιοχή φόντου, τότε το pixel κωδικοποιείται ως μηδέν. Διαφορετικά, η πραγματική τιμή του pixel κωδικοποιείται. Το αποτέλεσμα της διαδικασίας πρόβλεψης είναι δέσμες κωδικοποιημένων

σε μηδέν pixels για ομογενείς περιοχές. Τα pixels εκείνα μπορούν κατόπιν να μειωθούν μεσώ της χρήσης μιας άλλης τεχνικής συμπίεσης, όπως η κωδικοποίηση μήκους εκτέλεσης ή αυτή της σχετικής διεύθυνσης.

3.2.6. Προσαρμόσιμη Συμπίεση Εικόνας Διπλού Επιπέδου

Η Προσαρμόσιμη Συμπίεση Εικόνας Διπλού Επιπέδου (Adaptive Bilevel Image Compression - ABIC) αναφέρεται σε μια τάξη τεχνικών συμπίεσης που αναπτύχθηκαν στην IBM (Mitchel και Pennebaken, 1988) και επί του παρόντος τυποποιείται από την Ομάδα Ενωμένης Δυαδικής Εικόνας (Joint Binary Image Group - JBIG). Η ABIC χρησιμοποιεί έναν κωδικοποιητή πρόβλεψης δυο διαστάσεων για τον καθορισμό των σχετικών πιθανοτήτων του 0 ή του 1 στο επόμενο pixel που σαρώνεται. Η υποθετική πιθανότητα κατόπιν χρησιμοποιείται από έναν αριθμητικό κωδικοποιητή για την ελαχιστοποίηση του μεγέθους του λόγου μετάδοσης μιας εικόνας κοντά στην εντροπία της πηγής. Ο εκτιμητής πιθανότητας και αριθμητικός κωδικοποιητής πατενταρίστηκε από την IBM και αναφέρεται ως ένας Q- κωδικοποιητής. Η JBIG επεκτείνεται πάνω στη δουλειά της IBM με την εισαγωγή προσαρμόσιμων περιγραμμάτων. Εδώ το περίγραμμα παρουσιάζει τη σύνθεση μιας ομάδας από pixels που μπορούν να χρησιμοποιηθούν για τον καθορισμό μιας σχέσης προς το τρέχον pixel που μοντελοποιείται. Για παράδειγμα, ένα περίγραμμα μπορεί να

παριστάνει τη σύνθεση τριών pixels στην ίδια γραμμή που υπερβαίνει το pixel που έχει μοντελοποιηθεί και πέντε pixels κεντραρισμένα πάνω από εκείνο το pixel στην προηγούμενη γραμμή. Η τυποποίηση JBIG αναπτύχθηκε βασισμένη πάνω στις αρχικές προσπάθειες έρευνας της IBM για τη μετάδοση δυαδικών ή δυο επιπέδων κειμένων μεταξύ σταθμών εργασίας.

3.2.7. JPEG

Ο αλγοριθμος JPEG συμπίεσης και αποσυμπίεσης παρουσιάζει μια σειρά συνδεμένων βημάτων που περιγράφονται σε ένα επίσημο κείμενο περιεκτικών τυποποιήσεων που διατίθεται από το Αμερικανικό Ινστιτούτο Τυποποιήσεων (American National Standards Institute - ANSI).

Ο βασικός αλγόριθμος JPEG μπορεί να θεωρηθεί ως η βάση για την οποία υπάρχει ένας αριθμός επεκτάσεων. Δυο παραδείγματα JPEG επεκτάσεων περιλαμβάνουν την προοδευτική κωδικοποίηση που έχει ως αποτέλεσμα τη σταδιακή δόμηση ή αποδόμηση μιας εικόνας και τη χρήση μιας αριθμητικής κωδικοποίησης προς αντικατάσταση της κωδικοποίησης Hoftman που καθορίζεται στον βασικό αλγόριθμο.

Αρχικά ο JPEG ομαδοποιεί τα pixel σε μπλοκ των 8 X 8, οργανωμένα ως συστατικά χρώματος και φωτός (ένταση ή λαμπρότητα). Όταν εφαρμόζεται σε μια τυποποιημένη εικόνα σε μόνιτορ υπολογιστή που συντίθεται από κόκκινα, πράσινα και μπλε

(RGB) pixels, αυτό έχει ως αποτέλεσμα έναν YUV μετασχηματισμό, όπου το Y παριστάνει την ένταση και τα U και V παριστάνουν τις τιμές χρώματος. Σε μια φόρμα YUV η ίδια εικόνα απαιτεί λιγότερη αποθήκευση από ό,τι σε μια φόρμα RGB παρόλα αυτά, δεν υπάρχει αισθητή απώλεια της ποιότητας της εικόνας.

Αφού τα pixels σε μια εικόνα ομαδοποιηθούν σε μπλοκ, μια διαδικασία μετατροπής διακεκριμένου συνημίτονου (DCT) μετατρέπει τα μπλοκ των YUV pixels σε σετ συντελεστές που παριστάνουν τα απομονωμένα συστατικά συχνότητας των χρωμάτων και τις εντάσεις του μπλοκ. Κατά τη διάρκεια της DCT διαδικασίας, κάθε μπλοκ μετατρέπεται σε ένα σετ 64 συντελεστών - ένα συντελεστή dc (συνεχούς ρεύματος) και 63 συντελεστές ac (εναλλασσόμενου ρεύματος). Η πράξη αυτή μετατοπίζει το μπλοκ των pixels από την περιοχή χώρου στην περιοχή συχνότητας, με αποτέλεσμα η περισσότερη από τη μετατρεπόμενη ενέργεια του μπλοκ να συγκεντρώνεται στις χαμηλότερες συχνότητες. Πριν από το βήμα κβαντοποίησης ο συντελεστής dc κωδικοποιείται διαφορικά ανάλογα με το προηγούμενο μπλοκ.

Το βήμα κβαντοποίησης έχει ως αποτέλεσμα τη χρήση ενός πίνακα 64 κβαντοποιημένων τιμών εναντίον των οποίων συγκρίνονται οι 63 ac συνιστώσες του μπλοκ. Από τη στιγμή που τα περισσότερα ταιριάσματα δεν είναι ακριβή αλλά κατά προσέγγιση, η πράξη αυτή έχει ως αποτέλεσμα σύνιστώσα απώλειας της GPEG. Για παράδειγμα, θεωρήστε τη σύνθεση δυο υπο-μπλοκ που φαίνονται στο Σχήμα 3.1 c. Κατά τη διάρκεια της διαδικασίας κβαντοποίησης μικρές διαφορές μεταξύ των μπλοκ ίσως να μην έχουν επίδραση στη χρήση μιας διαφορετικής τιμής κβαντοποίησης που τέθηκε για κάθε μπλοκ. 'Ετσι, τα

δυο υπο-μπλοκ θα μπορούσαν να έχουν την ίδια κωδικοποιημένη τιμή παρόλα αυτά, στο παράδειγμα αυτό η αποσυμπίεση θα είχε ως αποτέλεσμα την απώλεια της σωστικής τιμής ενός pixel. Επιπρόσθετα, αφού η κβαντοποίηση εφαρμόζεται στο μητρώο συχνότητας που δημιουργήθηκε από τη λειτουργία CDT, η διαδικασία κβαντοποίησης μπορεί να ποικίλει για να μειωθεί το μέγεθος της απορρέουσας ροής δεδομένων. Για παράδειγμα, αφού η ανθρώπινη όραση είναι λιγότερο ευαίσθητη στο χρώμα απ' ό,τι στην ένταση, μπορεί να μειωθεί ο αριθμός των τιμών κβαντοποίησης ή των βημάτων που χρησιμοποιήθηκαν για την παράσταση του χρώματος, με αποτέλεσμα έναν υψηλότερο βαθμό συμπίεσης για τα συστατικά του χρώματος και ταυτόχρονο περιορισμό της οπτικά αισθητής υποβάθμισης της εικόνας. Παρόμοια, από τη στιγμή που η ανθρώπινη όραση είναι λιγότερο ευαίσθητη στις λεπτομέρειες υψηλής συχνότητας απ' ό,τι σ' αυτές χαμηλής συχνότητας, τα βήματα που χρησιμοποιούνται για την παράσταση των τιμών υψηλής συχνότητας μπορεί να αυξηθούν, κάτι που αυξάνει την συμπιεστότητα της εικόνας.

Η χρήση της εκτέλεσης τρεξίματος και της κωδικοποίησης Huftman περαιτέρω μειώνει το μέγεθος της εικόνας. Και ένας στατικός προκαθορισμένος πίνακας Huftman και ένας πίνακας Huftman δυο διελεύσεων που δημιουργήθηκαν για να παραστήσουν τη συγκεκριμένη εικόνα υποστηρίζονται από τον JPEG. Το τελικό βήμα της διαδικασίας JPEG είναι η δεσμοποίηση των δεδομένων, στην οποία οι αλληλουχίες των bit που παράγονται από τον κωδικοποιητή Huftman ομαδοποιούνται σε bytes.

3.2.8. Μοντέλο αντικειμένου

Η ικανότητα μιας εικόνας να μοντελοποιηθεί μπορεί να μειώσει σημαντικά τις περιττές επαναλήψεις. Παραδείγματα των τεχνικών συμπίεσης μοντέλου αντικειμένου περιλαμβάνουν κωδικοποίηση ταιριάσματος μοντέλου και κωδικοποίηση βασισμένη σε κλασματικά αντικείμενα.

3.2.9. Ταίριασμα μοντέλου

Η κωδικοποίηση ταιριάσματος μοντέλου έχει ως αποτέλεσμα την υποδιαίρεση μιας εικόνας σε μπλοκ που έχουν παρόμοια μοντέλα ή σχήματα. Κατόπιν, μόνο ένας αναγνωριστής μπλοκ και μπλοκ θέση, των μπλοκ, των οποίων η σύνθεση ταιριάζει σε ένα προηγούμενο αναγνωρισμένο μπλοκ, απαιτούν κωδικοποίηση. Για παράδειγμα, θεωρήστε το Σχήμα 3.1 b. Εδώ η εικόνα έχει σπαστεί σε 25 μπλοκ και για ευκολία, για σκοπούς επίδειξης, 19 αναπαριστάνουν αλληλουχίες pixels που αντιγράφτηκαν πιστά μ' έναν παρόμοιο τρόπο σε άλλα μπλοκ. Μια τεχνική PMC θα αποθήκευε τα περιεχόμενα ενός μπλοκ που αριθμείται 0 και ενός μπλοκ που αριθμείται 1. Κατόπιν, για την κωδικοποίηση του μπλοκ (0, 3) η αλληλουχία 0, 0, 3 θα χρησιμοποιούταν, με τον πρώτο αριθμό

να δηλώνει το γεγονός ότι το μπλοκ στη θέση 0, 3 ταιριάζει με τα περιεχόμενα του μπλοκ 0.

Μια σύγκριση των περιεχομένων του pixel μεταξύ των μπλοκ έχει ως αποτέλεσμα μια αύξηση στο μέγεθος του μπλοκ, ελαττώνοντας την πιθανότητα ενός ακριβούς ταιριάσματος της σύνθεσης των pixels. 'Ετσι, υπάρχει μια ανταλλαγή μεταξύ των προσπαθειών να ταιριαστούν μεγαλύτερα μπλοκ και της ικανότητας να γίνει κάτι τέτοιο. Μια μέθοδος η οποία απαλύνει μερικά την ανταλλαγή αυτή, είναι να υποτεθεί ότι τα περιεχόμενα σε pixel των μπλοκ ταιριάζουν ακόμα και όταν αυτό δεν είναι φυσικά δυνατό, οποτεδήποτε η διαφορά μεταξύ των μπλοκ είναι κάτω από ένα συγκεκριμένο κατώφλι. Για παράδειγμα, το Σχήμα 3.1 c δείχνει τη σύνθεση δυο γειτονικών μπλοκ εικόνας. Παρόλο που δεν ταιριάζουν ακριβώς, μπορεί να σχηματιστεί ο αλγόριθμος PMC ώστε να θεωρήσει ότι τα μπλοκ ταιριάζουν. Ενώ αυτή η μέθοδος έχει ως αποτέλεσμα μια αδυναμία για πλήρη ακριβή αντιγραφή του περιεχομένου pixel της εικόνας, αυξάνει την ικανότητα του αλγορίθμου να εξηγεί έναν περισσότερο αποτελεσματικό λόγο (αναλογία) συμπίεσης.

3.2.10. Κλασματική κωδικοποίηση

Η κλασματική κωδικοποίηση των εικόνων βασίζεται πάνω στην εργασία του Γάλλου Benoit Mandelbrot, του οποίου το κλασικό βιβλίο *Η Κλασματική Γεωμετρία Της Φύσης* (Mandelbrot, 1982) μπορεί να

θεωρηθεί ως το άνοιγμα ενός καινούργιου κλάδου των μαθηματικών. Ο Mandelbrot επινόησε τη λέξη 'κλασματικό' για να περιγράψει αντικείμενα που είναι 'σπασμένα' για τα οποία μπορούν να χρησιμοποιηθούν μαθηματικοί τύποι για να αναπαραστήσουν μια εικόνα.

Βασισμένη πάνω στην εργασία του Mandelbrot, μια εταιρεία με έδρα την Ατλάντα της Γεωργία, η Iterated Systems (Συνεχώς Επαναλαμβανόμενα Συστήματα), ανάπτυξε λογισμικό που μπορεί να χρησιμοποιηθεί για τη συμπίεση εικόνων, επιτυγχάνοντας αναλογίες συμπίεσης μέχρι ή και μεγαλύτερες από 10.000:1. Η τεχνική που χρησιμοποιήθηκε από την Iterated Systems ξεκινά με το σπάσιμο μιας ψηφιοποιημένης εικόνας σε μπλοκ ή τμήματα. Κάθε τμήμα συγκρίνεται προς μια βιβλιοθήκη κωδικών συστήματος συνεχώς επαναλαμβανόμενης λειτουργίας (IFS) που αναπαράγει αντίστοιχα κλάσματα, μειώνοντας σημαντικά το μέγεθος της βιβλιοθήκης. Οι κώδικες IFS παράγουν κλάσματα που συγκρίνονται προς την σύνθεση pixel ενός μπλοκ. Όταν το κλάσμα παρέχει μια κοντινή προσέγγιση του περιεχομένου του μπλοκ, ο κώδικας αντικαθιστά το μπλοκ. Παρόλο που αυτή η τεχνική συμπίεσης μπορεί να επιτύχει ένα λόγο συμπίεσης κάμποσων τάξεων μεγέθους ή μεγαλύτερο από άλλες τεχνικές, ο χρόνος υπολογισμού που απαιτείται για την κωδικοποίηση μιας εικόνας μπορεί να υπερβεί τις 100 ώρες ή και περισσότερο. 'Ετσι, η τεχνική αυτή δεν είναι πρακτική για τις εφαρμογές συμπίεσης που απαιτούν ταχύτητα, παρόλα αυτά, η αποσυμπίεση μπορεί να εμφανιστεί σχετικά γρήγορα, κάνοντας έτσι την τεχνική αυτή κατάλληλη για την κωδικοποίηση και διανομή μεγάλων αριθμών εικόνων ή εικόνων που σχετίζονται με μια μεγάλη βάση δεδομένων.

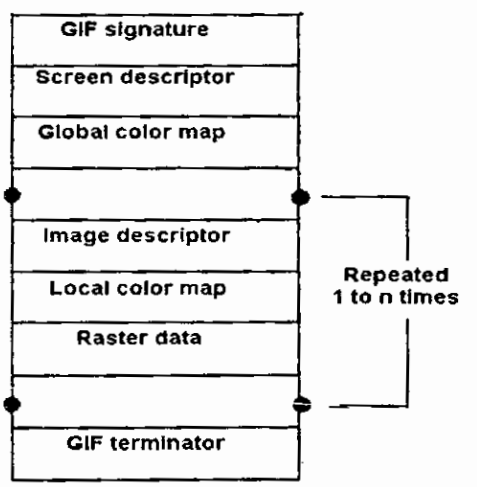
3.3. Διατάξεις φακέλων GIF

Υπάρχουν δυο διατάξεις GIF προσδιορισμένες προς το παρόν - η GIF87 που βασίζεται στην τυποποίηση CompuSeive που προτάθηκε το 1987, και η GIF89 που παρουσιάζει μια τροποποίηση στην αρχική τυποποίηση. Η τελευταία αναφέρεται περισσότερο τυπικά ως GIF89a, δηλώνοντας την 'Εκδοση GIF 89α. Και η GIF και η 'Διάταξη Διανταλλαγής Γραφικών' (Graphics Interchange Format) είναι σήματα κατατεθέντα της CompuSeive, Inc., μια εταιρεία H&R Block.

Η αρχική GIF προδιαγραφή αναπτύχθηκε το 1987 όταν η ικανότητα να επιδειχθούν περισσότερα από 16 χρώματα σε ένα μόνιτορ θεωρείτο ότι είναι το μακρινό μέλλον. Εξαιτίας αυτού, η τυποποίηση του 1987, καθώς επίσης και η αναθεώρησή της του 1989, περιορίζονται σε 256 χρώματα που προέρχονται από μια παλέτα των 16 εκατομμυρίων. Ενώ αυτό περιορίζει την ικανότητα της GIF να αναπαραστήσει αληθινά εκπληκτικά γραφικά, επίσης περιορίζει το μέγεθος ενός φακέλου που απαιτείται για την αποθήκευση μιας εικόνας. Αυτό με τη σειρά του μειώνει το χρόνο που απαιτείται για τη μετάδοση GIF κωδικοποιημένων εικόνων.

Μια τροποποιημένη μορφή της LZW συμπίεσης δομείται και στις δυο GIF τυποποιήσεις. Αυτό παρέχει μια μέθοδο μη απώλειας ή αντιστρεπτής συμπίεσης. Συγκριτικά, η JPEG, μία τεχνική που θα περιγραφτεί αργότερα στο κεφάλαιο αυτό, υποστηρίζει μια διαδικασία

συμπίεσης με απώλειες. Παρόλο που αυτό έχει ως αποτέλεσμα την υποβάθμιση της ποιότητας της εικόνας, επίσης κάνει δυνατή τη σημαντική μείωση του μεγέθους των φακέλων. Η πραγματική υποβάθμιση της ποιότητας της εικόνας ελέγχεται από το χρήστη και μια σημαντική μείωση στο μέγεθος μιας αποθηκευμένης εικόνας μπορεί να επιτευχθεί με λίγη έως καθόλου οπτικά αισθητή οπτική παραμόρφωση. Όμως, αν ο χρήστης τείνει να γίνεται άπληστος και να απαιτεί μια επιπρόσθετη μείωση στο μέγεθος της αποθηκευμένης εικόνας, η ικανότητα για οπτική αναγνώριση της εικόνας μπορεί να χειροτερέψει, όπως θα σημειώσουμε αργότερα στο κεφάλαιο αυτό. Το Σχήμα 3.3 δείχνει τη γενική διάταξη φακέλου GIF που είναι εφαρμόσιμη ουσιαστικά και στην GIF87a και στην GIF89a. Καθώς εξετάζουμε κάθε πεδίο στη διάταξη φακέλου που φαίνεται στο Σχήμα 3.3, θα σημειώσουμε τις διαφορές μεταξύ των δυο τυποποιήσεων.



Σχήμα 3.3 Γενική διάταξη φακέλου GIF

3.3.1. Υπογραφή GIF

Κάτω από την τυποποίηση GIF87a το πεδίο Υπογραφής GIF απαρτίζεται από την είσοδο GIF87a στα πρώτα έξι bytes του φακέλου. Κάτω από την τυποποίηση GIF89a το πεδίο Υπογραφής GIF μετονομάστηκε ως Επικεφαλίδα (Header), όμως, έξι bytes εξακολουθούν να χρησιμοποιούνται για την αναγνώριση των συμφραζομένων του ρεύματος δεδομένων GIF. Τα bytes 0 ως 2 συνεχίζουν να περιέχουν την κανονισμένη τιμή GIF, ενώ τα bytes 3 ως 5 αναγνωρίζουν τον αριθμό εκδοχής που χρησιμοποιείται για τη διάταξη του ρεύματος δεδομένων. Έτσι, η τυποποίηση GIF87a περιορίζεται στην υποστήριξη μιας διάταξης ρεύματος δεδομένων, ενώ η τυποποίηση GIF89a μπορεί να υποστηρίξει διαφορετικές διατάξεις ρεύματος δεδομένων.

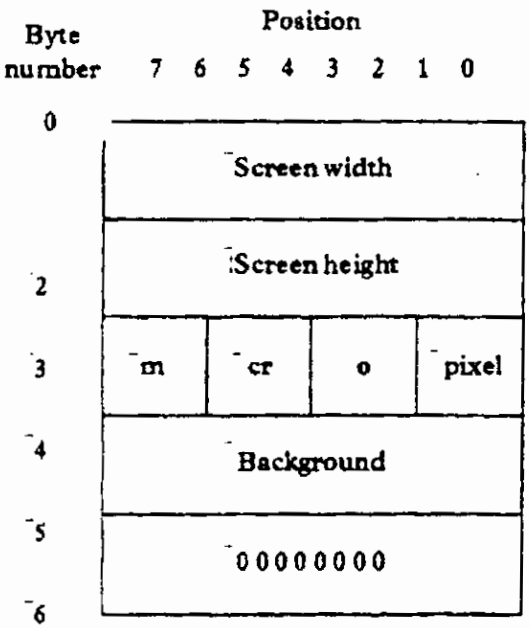
3.3.2. Περιγραφητής Οθόνης

Το πεδίο περιγραφής οθόνης απαρτίζεται από οκτώ bytes που καθορίζουν τις παραμέτρους των εικόνων GIF που περιλαμβάνονται στο φάκελο. Οι παράμετροι αυτές περιλαμβάνουν το εύρος και ύψος raster της εικόνας GIF, έναν δείκτη που καθορίζει αν ένας σφαιρικός έγχρωμος χάρτης ακολουθεί το πεδίο περιγραφητή οθόνης ή όχι, τον

αριθμό των bits της χρωματικής ανάλυσης, τον αριθμό των bits ανά pixel στην εικόνα και τον κατάλογο χρωμάτων που σχετίζεται με το φόντο της οθόνης. Η εικόνα 3.4 δείχνει τη διάταξη του πεδίου Περιγραφητή Οθονης.

Κάτω από την τυποποίηση GIF87a η τιμή ενός pixel παριστάνει το μέγιστο αριθμό χρωμάτων που χρησιμοποιούνται για την αναπαράσταση της εικόνας. Από τη στιγμή που για το cr (carriage return (?) - επαναφορέας κυλίνδρου εκτύπωσης) χρησιμοποιούνται τρία bits, η ακτίνα τιμών από 0 ως 7 παριστάνει ένα έως οκτώ bits που υποστηρίζουν δύο (μαύρο και άσπρο) μέχρι 256 χρώματα. Επίσης, κάτω από την τυποποίηση GIF87a, το bit 3 της λέξης 4 καθώς επίσης και όλα τα bit στη λέξη 6 διατηρήθηκαν για μελλοντική χρήση. Κάτω από την τυποποίηση GIF89a, το bit 3 της λέξης 4 γίνεται μια σημαία (δείκτης) ταξινόμησης. 'Οταν τίθεται στο 1, δείχνει ότι έχει διαταχθεί ένας επακόλουθος Σφαιρικός Χρωματικός Πίνακας με ελαττούμενη σημασία,με το πιο σημαντικό χρώμα να εμφανίζεται πρώτο στον πίνακα. 'Οταν η σημαία ταξινόμησης τίθεται στο 0, δείχνει ότι ο Σφαιρικός Χρωματικός Πίνακας δεν ταξινομείται. Μια δεύτερη αλλαγή κάτω από την τυποποίηση GIF89a αφορά τη χρήση της λέξης 6, που έχει κρατηθεί για μελλοντική χρήση κάτω από την τυποποίηση GIF87a. Κάτω από την τυποποίηση GIF89a η λέξη 6 περιέχει το λόγο δυο διαστάσεων pixel που παριστάνει έναν παράγοντα που χρησιμοποιείται για τον υπολογισμό μιας προσέγγισης του λόγου θεώρησης του pixel στην αρχική εικόνα. 'Οταν η τιμή του πεδίου δεν είναι μηδέν, η προσέγγιση του λόγου δυο διαστάσεων εξάγεται από τον ακόλουθο τύπο:

$$\text{Λόγος δυο διαστάσεων} = \frac{\text{Λόγος δυο διαστάσεων του pixel} + 15}{64}$$



Εύρος raster σε pixels (το λιγότερο σημαντικό bit πρώτο)

Ύψος raster σε pixels (το λιγότερο σημαντικό bit πρώτο)

M = ακολουθεί ο Σφαιρικός Εγχρωμος Χάρτης

crt1 = αριθμός των bits της ανάλυσης χρώματος

pixel+1 = νούμερο bits/pixel στην εικόνα

Background (φόντο) = ενδείκτης χρώματος του φόντου της εικόνας

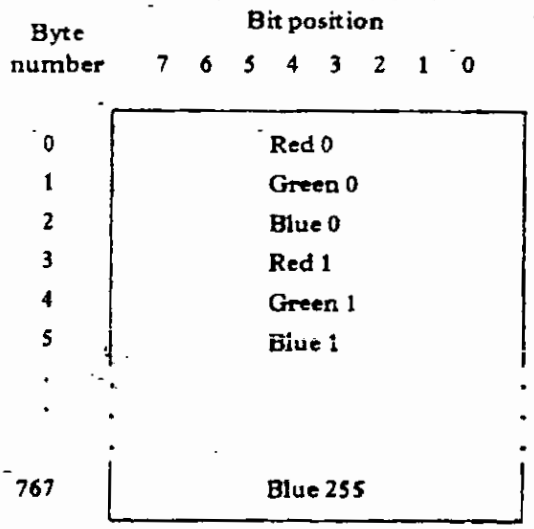
Σχήμα 3.4 Πεδίο περιγραφητή εικόνας

Όπου ο λόγος δυο διαστάσεων του pixel είναι το πηλίκο του εύρους του pixel δια του ύψους του. Η ακτίνα τιμών του πεδίου αυτού επιτρέπει τον καθορισμό από το πιο ευρύ pixel του 4:1 στο πιο ψηλό pixel του 1:4 σε προσαυξήσεις του 1 /64.

3.3.2. Σφαιρικός Έγχρωμος Χάρτης

Ο Σφαιρικός 'Εγχρωμος Χάρτης, που έχει μετονομαστεί σε Σφαιρικός 'Εγχρωμος Πίνακας κάτω από την τυποποίηση GIF89a, περιέχει μια αλληλουχία bytes που παριστάνουν τα τρίδυμα των χρωμάτων κόκκινου - πράσινου - μπλε. Το Σχήμα 3.5 δείχνει τη διάταξη του Σφαιρικού Χρωματικού Χάρτη/Πίνακα. Η παρουσία ενός Σφαιρικού Χρωματικού Χάρτη/Πίνακα δηλώνεται από μια μη-μηδενική Μ σημαία. Ο αριθμός των bytes σ' αυτόν τον χάρτη/πίνακα ισούται με: $3 \times 2^{(\text{Σφαιρικός Χρωματικός Χάρτης/Μέγεθος Πίνακα}+1)}$

Κάθε τιμή pixel εικόνας που λαμβάνεται απεικονίζεται σύμφωνα με το πλησιέστερο ταίριασμα του με ένα διαθέσιμο χρώμα της επίδειξης που βασίζεται πάνω στο χρωματικό του χάρτη.



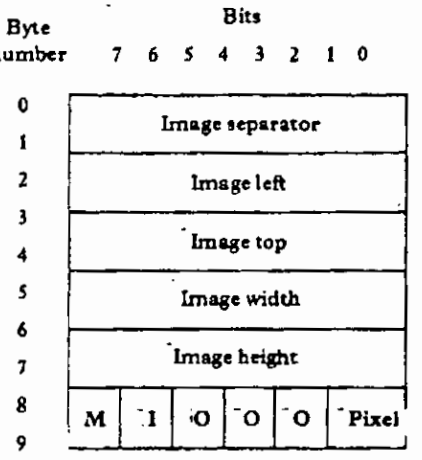
Σχήμα 3.5 Σφαιρικός χρωματικός χάρτης/πινακας

Αν δεν έχει δηλωθεί κάποιος Σφαιρικός Χρωματικός Χάρτης/Πίνακας από το M bit, αναπτύσσεται εσωτερικά ένας ερήμην χρωματικός χάρτης/πίνακας. Αυτός ο χάρτης/πίνακας αντιστοιχεί (χαρτογραφεί) κάθε πιθανό εισερχόμενο χρωματικό κατάλογο στον ίδιο hardware χρωματικό κατάλογο modulo <η>, όπου <η> είναι ο αριθμός των διαθέσιμων χρωμάτων του σκληρού δίσκου (hardware).

Τα συστατικά χρώματος παριστάνουν μια τιμή κλασματικής έντασης που κυμαίνεται από τίποτα (0) μέχρι πλήρες (255). Έτσι, το λευκό θα αναπαρασταθεί ως (255, 255, 255), το μαύρο ως (0, 0, 0) και το μεσαίο κίτρινο ως (180, 180, 0).

3.3.3. Περιγραφητής Εικόνας

Το πεδίο Περιγραφητή Εικονα απαρτίζεται από 10 bytes που καθορίζουν την πραγματική θέση μιας εικόνας μέσα στο χώρο που καθορίζεται στον Περιγραφητή Οθόνης, καθώς επίσης πώς σχηματίζεται η εικόνα, εάν θα χρησιμοποιηθεί ένας τοπικός ή σφαιρικός έγχρωμος χάρτης, και τον αριθμό των bits ανά pixel που χρησιμοποιούνται για την εικόνα. Το σχήμα 3.6 δείχνει τη διάταξη του πεδίου Περιγραφητή Εικόνας. Ο Διαχωριστής Εικόνας λειτουργεί ως ένας συγχρονιστικός χαρακτήρας προς τον ακόλουθο περιγραφητή εικόνας. Η τιμή του πεδίου αυτού καθορίζεται ως δεκαεξαδικό 2C.



Σχήμα 3.6 Πεδίο περιγραφητή εικόνας

Το πεδίο Αριστερής Εικόνας περιλαμβάνει τον αριθμό της στήλης σε pixels, της αριστερής άκρης οθόνης της εικόνας σε αντιστοιχία με το αριστερό άκρο της λογικής οθόνης. Το πεδίο Κορυφής Εικόνας καθορίζει τον αριθμό των σειρών σε pixels της άκρης κορυφής της λογικής οθόνης. Η άκρα αριστερή στήλη και η κορυφαία σειρά της λογικής οθόνης είναι και οι δυο 0. Τα πεδία Εύρους Εικόνας και Ύψους Εικόνας λειτουργούν παρόμοια με τα πεδία Αριστερής Εικόνας και Κορυφής Εικόνας, προσδιορίζοντας το εύρος και ύψος της εικόνας σε pixels.

Τα πακεταρισμένα υποπεδία στο byte 9 καθορίζουν εάν θα χρησιμοποιηθεί ένας Σφαιρικός Χρωματικός Χάρτης/Πίνακας (M=0) και θα αγνοηθεί το υποπεδίο των pixel ή αν θα χρησιμοποιηθεί ο Τοπικός Χρωματικός Χάρτης (M=1) καί θα χρησιμοποιηθεί το υποπεδίο των pixel, εάν η εικόνα είναι σχηματισμένη σε σειριακή (διαδοχική) διάταξη (I=0) ή αν η Εικόνα είναι σχηματισμένη σε διάταξη συμπλέγματος (I=1).

Κάτω από το σχηματισμό της GIF89a, το bit 5 ή το bit 9 χρησιμοποιούνται ως σημαία ταξινόμησης. Όταν τίθεται στο 0 το πεδίο αυτό δείχνει ότι δεν έχει διαταχθεί κάποιος τοπικός χρωματικός πίνακας. Όταν τίθεται σε μια τιμή 1, το πεδίο αυτό δείχνει ότι ο Τοπικός Χρωματικός Πίνακας ταξινομείται σε διάταξη φθίνουσας σημασίας, συνήθως με το πιο συχνό χρώμα να τοποθετείται πρώτο στον πίνακα. Το τελευταίο υποπεδίο στο byte 9 καθορίζει το μέγεθος του προαιρετικού Τοπικού Χρωματικού Χάρτη/Πίνακα και είναι εφαρμόσιμος μόνο όταν M=1. Η τιμή του πεδίου των τριών bit πρέπει να ανεβάζεται στη δύναμη του 2, και το 1 προστίθεται στο αποτέλεσμα ($pixel^2$+1), για τον καθορισμό του πραγματικού μεγέθους του Τοπικού Χρωματικού Χάρτη/Πίνακα.

3.3.4. Τοπικός Χρωματικός ΧάρτηςΙΠίνακας

Η παρουσία του προαιρετικού Τοπικού Χρωματικού Χάρτη/Πίνακα ορίζεται από τη θέση του M bit στο byte 9 του πεδίου Περιγραφητή Εικόνας. Αυτός ο χάρτης/πίνακας περιέχει μια αλληλουχία από bytes που παριστάνουν τα τρίδυμα χρωμάτων κόκκινου - πράσινου - μπλε των οποίων η διάταξη είναι ίδια όπως στον Σφαιρικό Χρωματικό Χάρτη/Πίνακα που δείχτηκε πριν στο Σχήμα 3.5. Αν είναι παρών ο Τοπικός Χρωματικός Χάρτης/Πίνακας, προσωρινά γίνεται ο ενεργός χρωματικός πίνακας και χρησιμοποιείται για να επεξεργαστεί την εικόνα που ακολουθεί αυτόν τον χάρτη/πίνακα.

3.3.5. Δεδομένα Raster

Το πεδίο Δεδομένων Raster καθορίζει τη διάταξη της πραγματικής εικόνας ως μια σειρά τιμών χρώματος pixel. Τα pixel αυτά αποθηκεύονται από αριστερά προς δεξιά σε αλληλουχία για μια σειρά εικόνας, με κάθε σειρά εικόνας να έχει διάταξη σε αλληλουχία από την κορυφή προς τη βάση. Η εξαίρεση στη διάταξη αυτή εμφανίζεται όταν το I bit στο byte 9 του Περιγραφητή Εικόνας τίθεται στο '1' για να ορίσει μια συμπλεγμένη εικόνα. Όταν εμφανίζεται κάτι τέτοιο, η διάταξη σειράς της εμφάνισης της εικόνας ακολουθεί μια διαδικασία τετραπλής διέλευσης (περάσματος) κατά την οποία η εικόνα γεμίζεται με σειρές που απέχουν ευρέως μεταξύ τους. Το πρώτο πέρασμα γράφει κάθε όγδοη σειρά, ξεκινώντας με την πρώτη επάνω σειρά του παραθύρου της εικόνας. Το δεύτερο πέρασμα γράφει κάθε τέταρτη σειρά, αρχίζοντας από την πέμπτη από την κορυφή σειρά. Το τρίτο πέρασμα γράφει κάθε τέταρτη σειρά ξεκινώντας από την τρίτη από την κορυφή

3.3.6. Τοπικός Χρωματικός Χάρτης/Πίνακας

Η παρουσία του προαιρετικού Τοπικού Χρωματικού Χάρτη/Πίνακα ορίζεται από τη θέση του M bit στο byte 9 του πεδίου Περιγραφητή Εικόνας. Αυτός ο χάρτης/πίνακας περιέχει μια αλληλουχία από bytes που παριστάνουν τα τρίδυμα χρωμάτων κόκκινου - πράσινου - μπλε των οποίων η διάταξη είναι ίδια όπως στον Σφαιρικό Χρωματικό Χάρτη/Πίνακα που δείχτηκε πριν στο Σχήμα 3.5. Αν είναι παρών ο Τοπικός Χρωματικός Χάρτης/Πίνακας, προσωρινά γίνεται ο ενεργός χρωματικός πίνακας και χρησιμοποιείται για να επεξεργαστεί την εικόνα που ακολουθεί αυτόν τον χάρτη/πίνακα.

3.3.7. Δεδομένα Raster

Το πεδίο Δεδομένων Raster καθορίζει τη διάταξη της πραγματικής εικόνας ως μια σειρά τιμών χρώματος pixel. Τα pixel αυτά αποθηκεύονται από αριστερά προς δεξιά σε αλληλουχία για μια σειρά εικόνας, με κάθε σειρά εικόνας να έχει διάταξη σε αλληλουχία από την κορυφή προς τη βάση. Η εξαίρεση στη διάταξη αυτή εμφανίζεται όταν το I bit στο byte 9 του Περιγραφητή Εικόνας τίθεται στο '1' για να ορίσει μια συμπλεγμένη εικόνα. Όταν εμφανίζεται κάτι τέτοιο, η διάταξη

σειράς της εμφάνισης της εικόνας ακολουθεί μια διαδικασία τετραπλής διέλευσης (περάσματος) κατά την οποία η εικόνα γεμίζεται με σειρές που απέχουν ευρέως μεταξύ τους. Το πρώτο πέρασμα γράφει κάθε όγδοη σειρά, ξεκινώντας με την πρώτη επάνω σειρά του παραθύρου της εικόνας. Το δεύτερο πέρασμα γράφει κάθε τέταρτη σειρά, αρχίζοντας από την πέμπτη από την κορυφή σειρά. Το τρίτο πέρασμα γράφει κάθε τέταρτη σειρά ξεκινώντας από την τρίτη από την κορυφή σειρά, ενώ το τέταρτο πέρασμα ολοκληρώνει την εικόνα γράφοντας ανά δεύτερη σειρά, αρχίζοντας από τη δεύτερη από την κορυφή σειρά. Το Σχήμα 3.7 δείχνει την διαδικασία σύμπλεξης.

Οι τιμές των pixel της εικόνας επεξεργάζονται ως μια σειρά χρωματικών δεικτών που χαρτογραφούνται στον χρωματικό χάρτη/πίνακα που χρησιμοποιείται. Κατόπιν επιδεικνύονται οι εξαγόμενες από τον χάρτη/πίνακα τιμές χρώματος. Οι σειρές των δεικτών pixel που περιέχονται στο πεδίο Δεδομένων Raster (ίσες με το εύρος εικόνας X ύψος εικόνας) περνούν στο ρεύμα δεδομένων εικόνας GIF μια τιμή ανά pixel, συμπιέζονται και πακετάρονται βάσει της χρήσης μιας εκδοχής της συμπίεσης LZW.

Image Row	Pass 1	Pass 2	Pass 3	Pass 4	Result
0	**1a**				**1a**
1				**4a**	**4a**
2			**2a**		**3a**
3				**4b**	**4b**
4		**2a**			**2a**
5				**4c**	**4c**
6			**3b**		**3b**
7				**4d**	**4d**
8	**1b**				**1b**
9				**4e**	**4e**
10			**3c**		**3c**
11				**4f**	**4f**

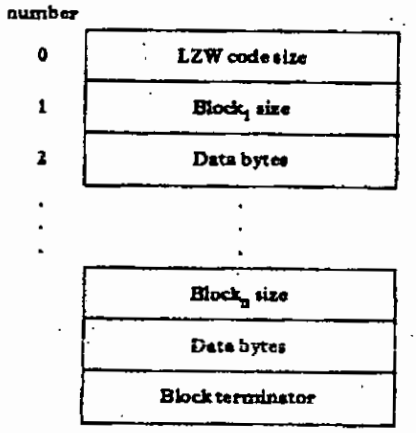
Σχήμα 3.7 Διαδικασία σύμπλεξης

3.3.8. LZW αλγόριθμος

Μια παραλλαγή του αλγόριθμου συμπίεσης LZW που περιγράφτηκε προηγουμένως στο Κεφάλαιο 6 χρησιμοποιείται για τη συμπίεση του πεδίου Δεδομένων Raster. Το πρώτο byte στο πεδίο Δεδομένων Raster παρουσιάζει το αρχικό μέγεθος κώδικα LZW το οποίο καθορίζει τον αριθμό των bytes που χρησιμοποιούνται για να παρουσιάσουν κάθε κώδικα συμπίεσης και συμβατικό (τυπικό) χαρακτήρα. Όταν έχει γεμίσει το αρχικό λεξικό, ο αλγόριθμος αυξάνει το μέγεθος του LZW κώδικα κατά ένα byte. Οι ασπρόμαυρες εικόνες που έχουν ένα χρωματικό bit πρέπει να χρησιμοποιούν ένα μέγεθος κώδικα τιμής 2 για να επιτευχθεί συμπιεστότητα, εμφανίζοντας το ελάχιστο μέγεθος κώδικα. Στην πραγματικότητα ένα bit προστίθεται σε

κάθε μέγεθος κώδικα ανά κώδικα, έως και 12 bits ανά κώδικα. Έτσι, το μέγιστο μέγεθος κώδικα είναι 12 bits, πράγμα που επιτρέπει να υποστηρίζεται ένα λεξικό 4.095 εισόδων.

Η LZW συμπίεση έχει ως αποτέλεσμα διάφορα μήκη κωδικών από τρία μέχρι 12 bits σε μήκος, τα οποία κατόπιν ανασχηματίζονται σε μια σειρά από 8-bit bytes για αποθήκευση. Το πακετάρισμα εμφανίζεται από δεξιά προς τα αριστερά, με κάθε ομάδα των 255 bytes να πακετάρεται σε μπλοκ προκαθορισμένα από μια επικεφαλίδα μεγέθους μπλοκ. Αν το τελευταίο μπλοκ δεν περιέχει 255 χαρακτήρες, τότε η επικεφαλίδα μεγέθους μπλοκ δείχνει το μέγεθος του τελευταίου μπλοκ. Το σχήμα 3.8 δείχνει το πεδίο Δεδομένων Raster το οποίο περιέχει τα αποτελέσματα της LZW λειτουργίας.



Σχήμα 3.8 Πεδίο δεδομένων Raster

Το τελευταίο byte στο πεδίο Δεδομένων Raster, το υποπεδίο Τερματιστή Μπλοκ (Block Terminator), απαρτίζεται από ένα μονό byte

με την τιμή δεκαεξαγωνικού 00. Καθώς δηλώνει το όνομά του, αυτό το μπλοκ τερματίζει μια αλληλουχία υπομπλόκ δεδομένων.

Παρόλο που ο GIF LZW αλγόριθμος ταιριάζει με τον τυποποιημένο LZW αλγόριθμο, διαφέρει από τον τελευταίο κατά διάφορους τρόπους. Πρώτα, ο GIF LZW αλγόριθμος χρησιμοποιεί έναν Καθαρό Κώδικα (Clear Code) του οποίου η τιμή είναι 2 κωδικό μέγεθος για την επαναφορά των αρχικών ενδείξεων των παραμέτρων και πινάκων συμπίεσης για την κατάσταση έναρξης. Δεύτερο, καθορίζεται ένας Κώδικας Τέλους Πληροφόρησης, ο οποίος δείχνει το τέλος του ρεύματος δεδομένων εικόνων. Η τιμή του κώδικα αυτού είναι ο Καθαρός Κώδικας +1, που εξηγεί γιατί η πρώτη διαθέσιμη τιμή του κώδικα συμπίεσης γίνεται «καθαρός κώδικας»+2. Όπως αναφέρθηκε προηγούμενα, οι κώδικες εξόδου ποικίλουν σε μήκος ξεκινώντας από το «μέγεθος κώδικα» + 1 bits ανά κώδικα, μέχρι 12 bits ανά κώδικα. Ο Πίνακας 3.3 συνοψίζει τις GIF LZW παραμέτρους και τις τιμές τους.

LZW parameter	Value
Clear Code	2 × code size
End of Information Code	clear code + 1
First available compression code value	clear code + 2
Output codes (min, max)	code size + 1, 12 bits
Terminator block	hex 00

Πίνακας 3.3

3.3.9. Επεκτάσεις GIF89

Διάφορες προαιρετικές επεκτάσεις μπορούν να ακολουθήσουν τον GIF τερματιστή κάτω από την τυποποίηση GIF89. Ένα πεδίο γραφικού Επέκτασης Ελέγχου περιλαμβάνει πληροφορίες για τον τρόπο με τον οποίο πρέπει να χειριστεί η εικόνα από τη στιγμή που εμφανίζεται, αν ή όχι η είσοδος του χρήστη αναμένεται και ένας χρόνος καθυστέρησης για αναμονή πριν τη συνέχιση της επεξεργασίας του ρεύματος δεδομένων. Άλλες επεκτάσεις περιλαμβάνουν ένα μπλοκ Επέκτασης Σχολίου που επιτρέπει το συνυπολογισμό της βασισμένης σε κείμενο πληροφόρησης που δεν είναι πέρα από την εικόνα και μια απλή Επέκταση Κειμένου που περιέχει κειμενικά δεδομένα και τις παραμέτρους που απαιτούνται για να αποδοθούν τα δεδομένα ως γραφικό. Αφού κυρίως ασχολούμαστε με τη συμπίεση GIF, οι αναγνώστες παραπέμπονται στην CompuServe FIG89, μια τυποποίηση για λεπτομερή πληροφόρηση αναφορικά με τις αμέσως προηγούμενες επεκτάσεις.

3.3.10. Προτεινόμενοι τρόποι GIF προδιαγραφών

Μια πλέον ενδιαφέρουσα πρόταση για την τροποποίηση των GIF προδιαγραφών για τη βελτίωση της αποδοτικότητας της LZW συμπίεσής τους έγινε γνωστή στην CompuSeive τον Ιούνιο 1994 από τον J.F.R. `Frank' Slinkman. Ο κ. Slinkman, ο οποίος μπορεί να βρεθεί στην CompuServe στο 72411,650, σημείωσε ότι πολλοί από τους LZW κώδικες που προστέθηκαν στον πίνακα κωδικών δεν χρησιμοποιούνται ποτέ. Αυτό κάνει τον πίνακα να γεμίζει νωρίτερα απ' ό,τι χρειάζεται και να ελαττώνει την αποδοτικότητα του σχήματος LZW συμπίεσης.

Εάν όλοι οι μη χρησιμοποιούμενοι κώδικες απλώς απομακρύνονταν κάθε φορά που γέμιζε ο πίνακας, θα δημιουργούταν μια κατάσταση κατά την οποία μερικοί νέοι και μη τοποθετημένοι για αναφορά στον πίνακα κώδικες θα τοποθετούνταν κοντά στην κορυφή του πίνακα και ο πίνακας θα γέμιζε πρώτα με εκείνους τους κώδικες που έχουν την ευκαιρία να αναφέρονταν από υψηλότερους κώδικες. Δυστυχώς, η απομάκρυνση των κωδικών εκείνων θα είχε ως αποτέλεσμα την τελική αντικατάστασή τους από επιπρόσθετους κώδικες που ποτέ δεν θα τοποθετούνταν σε αναφορά, κάτι που θα μπορούσε να παρεμποδίσει το φυσιολογικό, πλήρες άδειασμα. Για να διορθωθεί η κατάσταση αυτή, ο κ. Slinkman πρότεινε την εφαρμογή μιας λειτουργίας που ψάχνει για μη χρησιμοποιούμενους κώδικες σε

αυξανόμενα μικρότερα και υψηλότερα τμήματα του LZW πίνακα. Το Σχήμα 3.9 καταγράφει σε λίστα τον κώδικα C γλώσσας της προτεινόμενης μετατροπής του στην τυποποίηση GIF. Η λειτουργία C γλώσσας του κ. Slinkman και η πρότασή του περιέχονται στο file PRTCLR.ARC της δισκέτας διευκόλυνσης με τον τίτλο 'Graphics' (γραφικά).

Την πρώτη φορά που καλείται η λειτουργία, ελέγχει το κάτω μισό του πίνακα. Αυτή η έρευνα βασίζεται πάνω στην προϋπόθεση ότι ένας κώδικας που δεν τοποθετείται για αναφορά σε πίνακα από οποιονδήποτε από τους 2048 κώδικες στο επάνω μισό του πίνακα είναι απίθανο να χρησιμοποιείται. Από τη στιγμή που απομακρυνθούν οι μη χρησιμοποιούμενοι κώδικες στο κάτω μισό του πίνακα, οι υψηλότεροι κώδικες επανατοποθετούνται για να αντικαταστήσουν τους κώδικες που απομακρύνθηκαν. Κάθε επακόλουθη κλήση λειτουργίας ελέγχει το κάτω μισό του τμήματος του πίνακα που ξεκινά από τη σχισμή (εγκοπή) επάνω από τη νέα τοποθεσία του υψηλότερου κώδικα ο οποίος ελέγχθηκε προηγούμενα. Εάν λιγότεροι από το 1/32 των ελεγμένων κωδίκων δεν χρησιμοποιούνται στη διάρκεια οποιουδήποτε περάσματος, η διαδικασία τελειώνει με τη λειτουργία να επαναρυθμίζεται απο μόνη της και να επιστρέφει στο μηδέν, δηλώνοντας ότι ολόκληρος ο πίνακας θα μπορούσε να καθαριστεί με τον φυσιολογικό τρόπο. Επιπρόσθετα, αν η προς έλεγχο περιοχή είναι πολύ μικρή και πολύ ψηλά στον πίνακα, η λειτουργία επίσης θα επαναρυθμίστεί από μόνη της και θα επιστρέψει στο μηδέν, αφού κώδικες ψηλά στον πίνακα σχετίζονται με μεγαλύτερα μήκη απ' ότι οι κώδικες χαμηλότερα στον πίνακα. Η τεχνική αυτή παρέχει σε κάθε κώδικα μια ευκαιρία να τοποθετηθεί στον πίνακα για αναφορά από

έναν υψηλότερο κώδικα, ενώ η αναγνωρίζεται το γεγονός ότι εάν αντιμετωπίζονται λιγότεροι μη χρησιμοποιούμενοι κώδικες η συμπίεση LZW θα εκτελείται αρκετά ικανοποιητικά και ο πίνακας κωδίκων θα μπορεί να μην τροποποιηθεί.

Η καταλογοποίηση του προγράμματος C γλώσσας που περιέχεται στο Σχήμα 3.9 δείχνει έναν τρόπο για τον εντοπισμό και την απομάκρυνση μη χρησιμοποιούμενων κωδίκων LZW πίνακα. Οι αναγνώστες θα πρέπει να σημειώσουν ότι αυτή η τεχνική 'μερικού καθαρισμού LZW' παρουσιάζει μια πρόταση για μια αλλαγή στην τυποποίηση GIF και επί του παρόντος δεν είναι τμήμα της τυποποίησης GIF.

```
DEFINT A-Z
     CONST Prefixes = 0
     CONST Suffixes = 1
     DIM StringTable%(4095, 1)
     DIM Stack(4095)
     False = 0
     True = NOT False

     ze$ = CHR$(0)
     cmd$ = COMMAND$
     CLS
     CLOSE 1
     OPEN cmd$ + ".gif" FOR INPUT AS 1   'make sure it exists
     CLOSE 1
     OPEN cmd$ + ".gif" FOR BINARY ACCESS READ AS 1
     'All additions to the header processing should be a part of the
     'kernel
Kernel:
     GOSUB GIFHeader
     CLS
     GOSUB ImageProcessing
     CLOSE
     BEEP
     WHILE a$ = ""   'Wait loop to keep the image on screen until a key is
pressed
          a$ = INKEY$

     WEND
```

```
      END '****************** Program ENDS ********************
'******************* SUBROUTINES ***************************************
GIFHeader:      'Process the Gif Header Information
      DO                  'ignore leading junk bytes
            g$ = INPUT$(1, #1)
      LOOP WHILE g$ <> "G" AND NOT EOF(1)

      if$ = INPUT$(2, #1)
      Version$ = INPUT$(3, 1)
      ScreenWidth = CVI(INPUT$(2, #1))     'Convert two bytes to an integer
      ScreenHeight = CVI(INPUT$(2, #1))    ' as above
      MapDat = ASC(INPUT$(1, #1))
      GlobalMap = (MapDat AND &H80) / &H80  'Get the Global Color Map
Bit
      Cres = (MapDat AND &H70) / &H10 + 1   'Color Resolution
      R1 = (MapDat AND &H10) / &H10         'Reserved Bit
      PixelSize = (MapDat AND &H7) + 1      'Pixel Size
      BGColor = VAL(INPUT$(1, #1))          'Background Color Index
      SrtDat = ASC(INPUT$(1, #1))           'Sort and Aspect Ration
information
      Sort = (SrtDat AND &H80) / &H80       'Sorted Color Map Flag
      PixAspect! = ((SrtDat AND &H7F) + 31) / 64!   'Pixel Aspect Ratio
      colors = 2 ^ PixelSize                'Number of Possible Colors
'**** Select overall screen mode based on screen height.
' This select structure can is used to select the screen setup based upon
' the parameters from the header. These were my intial choices but are by
' no means absolute.
'
' I use two basic methods for pixel display.
'
'  1) Select a screen mode which closely matches the global screen specs
'     and then paint actual pixels as required.
'
'  2) Select a higher resolution screen mode than required then map the
'     screen coordinates to the image screen size using the basic WINDOW
'     SCREEN function. To eliminate the gaps that would result from just
'     plotting pixels, I draw filled boxes that are 1 logical pixel wide
'     and 1 logical pixel wide. In this way the BASIC function takes care
'     of the "gaps".
'
'     In the select structure, Pixel is set TRUE for real pixel drawing or
'     FALSE for Mapping and Box drawing.
'
'     The scale parameter is set to enable the palette selection math to
'     calculate properly.

      SELECT CASE ScreenWidth
            CASE 1 TO 320
                  mode = 13
                  Pixels = True
```

```
                SCREEN mode
                WINDOW
                scale = 4
            CASE 640
                SELECT CASE ScreenHeight
                    CASE 200
                        mode = 8
                        Pixels = True
                        SCREEN mode
                        WINDOW
                        scale = 1
                    CASE 201 TO 350
                        mode = 9
                        Pixels = True
                        SCREEN mode
                        WINDOW
                        scale = 1
                    CASE 351 TO 480
                        mode = 12
                        Pixels = True
                        SCREEN mode
                        WINDOW
                        scale = 4
                    CASE ELSE
                        mode = 12
                        Pixels = False
                        SCREEN mode
                        WINDOW SCREEN (1, 1)-(ScreenWidth,
ScreenHeight)
                        scale = 4
                END SELECT
            CASE 641 TO 700
                scale = 4
                mode = 12
                Pixels = True
                SCREEN mode
                WINDOW
            CASE 720
                scale = 4
                mode = 12
                Pixels = True
                SCREEN mode
                WINDOW
            CASE ELSE
                scale = 4
                mode = 12
                Pixels = False
                SCREEN mode
                WINDOW SCREEN (1, 1)-(ScreenWidth, ScreenHeight)
            END SELECT
```

```
'****** Print some diagnostic information before image processing

      PRINT "Screen Width", ScreenWidth
      PRINT "Screen Height", ScreenHeight
      PRINT "Mode=", mode,
      PRINT "Scale=", scale
      PRINT "Press SPACE BAR to continue..."

      WHILE p$ <> " "          'clear the keyboard buffer.
            p$ = INKEY$
      WEND

      WHILE INKEY$ <> ""       'wait for a key press to proceed.
           dummy$ = INKEY$
      WEND

GlobalMap:    'Get Global Color Map information
      ColorOffset = 0    'This will be used in the future for palette
fragmentation
      IF GlobalMap THEN GOSUB GetColorMap
RETURN '***************** GIFHeader Ends ************************
GetColorMap:   '************ Process Color Map Information *************

      FOR i = ColorOffset + 1 TO 2 ^ PixelSize + ColorOffset
                Red = INT(ASC(INPUT$(1, #1)) / scale)
                Green = INT(ASC(INPUT$(1, #1)) / scale)
                Blue = INT(ASC(INPUT$(1, #1)) / scale)
           SELECT CASE mode
                CASE 12, 13
                     PALETTE i - 1, 65536 * Blue + 256 * Green + Red
                     ' per QUICKBASIC manual. Colors are outstanding!
                CASE 9
                     pal = 0
                     SELECT CASE Red
                          CASE 0 TO 63                'black
                               pal = pal
                          CASE 64 TO 127               'dark red
                               pal = pal + &H20
                          CASE 128 TO 191              'medium red
                               pal = pal + &H4
                          CASE 192 TO 255              'light red
                               pal = pal + &H24
                     END SELECT
                     SELECT CASE Green                   'and so forth
                          CASE 0 TO 63
                               pal = pal
                          CASE 64 TO 127
                               pal = pal + &H10
                          CASE 128 TO 191
```

```
                    pal = pal + &H2
                CASE 192 TO 255
                    pal = pal + &H12
            END SELECT
            SELECT CASE Blue
                CASE 0 TO 63
                    pal = pal
                CASE 64 TO 127
                    pal = pal + &H8
                CASE 128 TO 191
                    pal = pal + &H1
                CASE 192 TO 255
                    pal = pal + &H9
            END SELECT
            PALETTE i - 1, pal
        CASE 8
            pal = 0
            SELECT CASE Red
                CASE 0 TO 127          'red bit off
                    pal = pal
                CASE 128 TO 255        'red bit on
                    pal = pal + 4
                    intense = 8     'set intensity bit (optional)
            END SELECT
            SELECT CASE Green              ' and so forth
                CASE 0 TO 127
                    pal = pal
                CASE 128 TO 255
                    pal = pal + 2
                    intense = 8
            END SELECT
            SELECT CASE Blue
                CASE 0 TO 127
                    pal = pal
                CASE 128 TO 255
                    pal = pal + 1
                    intense = 8
            END SELECT
        PALETTE i - 1, pal + intense
    END SELECT
NEXT i
'NewOffset = 2 ^ PixelSize   'For future palette fragmentation (not
presently used)
RETURN  '********************* Get Color Map Ends *****************

ImageProcessing:    '****** Image processing block
DO                      'Look for "," denoting the start of the block
    IF EOF(1) THEN RETURN
    Image$ = INPUT$(1, #1)
LOOP UNTIL Image$ = ","         'get image block information as before
```

```
        ImageLeft = CVI(INPUT$(2, #1))
        ImageTop = CVI(INPUT$(2, #1))
        ImageWidth = CVI(INPUT$(2, #1))
        ImageHeight = CVI(INPUT$(2, #1))
        'PRINT Image$, ImageLeft, ImageTop, ImageWidth, ImageHeight
        LocalMapDat = ASC(INPUT$(1, #1))
        LocalMap = (LocalMapDat AND &H80) / &H80
        Interlace = (LocalMapDat AND &H40) / &H40
        LSrt = (LocalMapDat AND &H20) / &H20
        LR1 = (LocalMapDat AND &H10) / &H10
        LR2 = (LocalMapDat AND &H8) / &H8
        LPixelSize = (LocalMapDat AND &H7) + 1
        IF LocalMap THEN
            GOSUB GetColorMap          'Use local Color map.
            ColorOffset = ColorOffset  ' + NewOffset (future)
        END IF
        GOSUB Decode                   'Display the GIF image
        IF NOT EOF(1) THEN GOTO ImageProcessing    'Get more images
        RETURN  '****************** ImageProcessing ENDS ***************

Decode:  '****************** GIF Decompression and Display *********
        x = ImageLeft
        y = ImageTop
        Pass = 1          'used to control interlace display
        YInterlace = 8
        FOR i = 1 TO 2 ^ PixelSize
            StringTable%(i, Suffixes) = i    'initialize the string table
        NEXT i

                          '*** read root code size
        FI$ = INPUT$(1, #1)
        PixelCodeSize = ASC(FI$ + ze$)
                          '*** initialize variables
        CodeSize = PixelCodeSize + 1
        ClearCode = 2 ^ PixelCodeSize
        EndOfImage = ClearCode + 1
        NextCode = EndOfImage + 1
        MaxCode = 2 ^ CodeSize
                          '*** read first block length
        FI$ = INPUT$(1, #1)
        BlockLength = ASC(FI$ + ze$)
        ByteCount = 1
                          '*** read first data byte
        FI$ = INPUT$(1, #1)
        Byte = ASC(FI$ + ze$)
        RightShiftCount = 0
                          '*** read code
ReadCode: '************  Image Data Read Loop Begins ********************
        Code = 0
        FOR i = 1 TO CodeSize              'peel out CodeSize bits
```

```
            IF INKEY$ <> "" THEN          'check for key press abort.
                CLOSE
                END
            END IF
            Code = Code + (Byte AND 1) * 2 ^ (i - 1)       ' reconstruct code -
peel off the right bit
            Byte = INT(Byte / 2)                          ' shift right one bit
            RightShiftCount = RightShiftCount + 1          ' count the bits
shifted out of the byte
            IF RightShiftCount < 8 THEN GOTO NextBit       ' Process all 8
bits of the byte before reading another
            ByteCount = ByteCount + 1                      ' count the bytes
processed and compare
            IF ByteCount <= BlockLength THEN GOTO NextByte  ' with the
block length read.
            FI$ = INPUT$(1, #1)                            ' read the next block
length
            BlockLength = ASC(FI$ + ze$)                   ' convert to an integer
            ByteCount = 1                                 ' reset the byte counter
            IF BlockLength = 0 THEN RETURN                 ' zero means we
are done with this image
NextByte:     FI$ = INPUT$(1, #1)                          ' get another byte
            Byte = ASC(FI$ + ze$)                         ' convert to an integer
            RightShiftCount = 0                           ' reset the bit shift counter
NextBit:
      NEXT i


 'Now that we have the next compressed code, go expland it into a string if
required.

      IF Code = ClearCode THEN
            CodeSize = PixelCodeSize + 1
            NextCode = ClearCode + 2
            MaxCode = 2 ^ CodeSize
            Flag = 0
            GOTO ReadCode
      END IF
      IF Code = EndOfImage THEN RETURN
      IF Flag <> 1 THEN
            Flag = 1
            Value = Code
            GOSUB Display
            Prefix = Code
            GOTO ReadCode
      END IF
      IF Code < NextCode THEN     '*** code exists in string table
            Value = Code                  'set the value to the input code
            DO WHILE Value > ClearCode          'decompose the code
until it is a pixel value
```

```
                Value = StringTable%(Value, Prefixes)    'and not another
string code
            LOOP
            StringTable%(NextCode, Suffixes) = Value 'store the pixel value
at the end of the table
            Value = Code                        'restore the input value


        ELSE                    '*** code doesn't exist in string table
            Value = Prefix                  'set the value to the last prefix
            DO WHILE Value > ClearCode          'decompose the code
until it is a pixel value
                Value = StringTable%(Value, Prefixes)    'and not another
string code
            LOOP
            StringTable%(NextCode, Suffixes) = Value 'store the pixel value
at the end of the table
            Value = NextCode                    'set value to next table index
        END IF
        StringTable%(NextCode, Prefixes) = Prefix       'store the current prefix
in the table
        GOSUB Display                       'go display pixel(s)
        Prefix = Code                       'Code becomes next prefix
        NextCode = NextCode + 1                 'next table index

        IF NextCode = MaxCode THEN              'increment code size if
needed
            CodeSize = CodeSize + 1
            MaxCode = MaxCode * 2
            IF CodeSize = 13 THEN CodeSize = 12  'codes are never larger
than 12
        END IF
        GOTO ReadCode    '****************** Bottom of Image Read Loop
******

Display:    'Display pixels by decoding from the string table while pushing
them onto
            'a last in, first out stack
        StackPointer = 0
        Stack(StackPointer) = Value      'Push value onto the stack
        StackPointer = StackPointer + 1
        DO WHILE Value > ClearCode              'if value is a string code
and not
            Value = StringTable%(Value, Prefixes)  'a pixel value then
            Stack(StackPointer) = Value         'decompose and push
prefixes
            StackPointer = StackPointer + 1     'on the stack
        LOOP
        DO UNTIL StackPointer = 0               'Now, Pop the stack and
use the
```

```
        StackPointer = StackPointer - 1        'prefixes to point to the
suffixes
        Value = StringTable%(Stack(StackPointer), Suffixes)  'which are
the actual
                                               'output pixel values
        IF Pixels THEN                 'Paint pixels per mode selection
            IF Interlace = 0 THEN    'Non interlaced image
                IF x > (ImageLeft + ImageWidth - 1) THEN
                    x = ImageLeft
                    y = y + 1
                END IF
                x = x + 1
                PSET (x, y), Value
            ELSE                        'Interlaced image
                IF x > (ImageLeft + ImageWidth - 1) THEN
                    x = ImageLeft
                    y = y + YInterlace
                END IF
                x = x + 1
                IF y >= ImageTop + ImageHeight THEN
                    SELECT CASE Pass    'y increment depends on
interlace pass
                        CASE 1
                            y = ImageTop + 4
                        CASE 2
                            y = ImageTop + 2
                            YInterlace = 4
                        CASE 3
                            y = ImageTop + 1
                            YInterlace = 2
                        CASE ELSE
                    END SELECT
                    Pass = Pass + 1
                END IF
                PSET (x, y), Value
            END IF
        ELSE                            'Paint Boxes instead of Pixels.
            IF Interlace = 0 THEN                       'and so forth.
                IF x > (ImageLeft + ImageWidth - 1) THEN
                    x = ImageLeft
                    y = y + 1
                END IF
                x = x + 1
                LINE (x, y)-(x + 1, y + 1), Value, BF
            ELSE
                IF x > (ImageLeft + ImageWidth - 1) THEN
                    x = ImageLeft
                    y = y + YInterlace
                END IF
                x = x + 1
```

```
                    IF y >= ImageTop + ImageHeight THEN
                        SELECT CASE Pass
                            CASE 1
                                y = ImageTop + 4
                            CASE 2
                                y = ImageTop + 2
                                YInterlace = 4
                            CASE 3
                                y = ImageTop + 1
                                YInterlace = 2
                            CASE ELSE
                        END SELECT
                        Pass = Pass + 1
                    END IF
                    LINE (x, y)-(x + 1, y + 1), Value, BF
                END IF
            END IF
        LOOP
RETURN '********************* Decode ENDS ****************
```

Σχήμα 3.9.

Βιβλιογραφία

- Gilbert Held, Data and Image Compression, J.Wiley, 1996.
- A.K. Jain, Fundamentals of Digital Image Processing, Prentice Hall, 1989.
- E. Horowitz, S. Sahni, Fundamentals of computer algorithms, Computer Science Press, 1984.
- Ιωάννης Πίτας, Ψηφιακή επεξεργασία εικόνας, 1999.