6. Κατακερματισμός

6.1 Τεχνικές Κατακερματισμού

Στην γραμμική αναζήτηση και στην δυαδική αναζήτηση, για να εντοπιστεί ένα στοιχείο απαιτείται μια σειρά από συγκρίσεις, όπου το ζητούμενο στοιχείο συγκρίνεται επανειλημμένα με τα στοιχεία της λίστας. Αν τα στοιχεία είναι *n*, τότε κατά μέσο όρο απαιτούνται *n* συγκρίσεις στην γραμμική αναζήτηση και $\log_2 n$ στη δυαδική. Οι αλγόριθμοι γραμμικής και δυαδικής αναζήτησης μπορεί να είναι αργοί σε ορισμένες περιπτώσεις. Για παράδειγμα, ένας πίνακας συμβόλων, που κατασκευάζεται από έναν compiler, χρησιμοποιείται για την αποθήκευση αναγνωριστικών και πληροφοριών σχετικών μ' αυτά. Η ταχύτητα κατασκευής και αναζήτησης ενός τέτοιου πίνακα εξαρτάται από την ταχύτητα μεταγλώττισης. Για ταχύτερη αναζήτηση, χρησιμοποιείται συνήθως μια δομή δεδομένων που ονομάζεται **πίνακας κατακερματισμού** (**hash table**). Η θέση ενός στοιχείου μέσα σε έναν τέτοιο πίνακα ορίζεται απευθείας ως συνάρτηση του ίδιου του στοιχείου και όχι ύστερα από μια σειρά συγκρίσεων. Σε ιδανικές συνθήκες, για τον εντοπισμό ενός στοιχείου σε ένα πίνακα κατακερματισμού απαιτείται μία μόνο σύγκριση κι επομένως ο χρόνος αναζήτησης είναι σταθερός και δεν εξαρτάται από τον αριθμό των αποθηκευμένων στοιχείων.

Έστω, για παράδειγμα, ότι σε ένα πίνακα κατακερματισμού πρόκειται να αποθηκευτούν 10 ακέραιοι αριθμοί με τιμές μεταξύ 0 και 99. Ο πίνακας αυτός μπορεί να υλοποιηθεί με έναν πίνακα ακεραίων Table με δείκτες 0..99, όπου κάθε στοιχείο του πίνακα έχει αρχικά μια εικονική τιμή, π.χ. -1. Αν αποθηκεύουμε κάθε ακέραιο i στη θέση Table[i], τότε για να καθορίσουμε αν ένας συγκεκριμένος ακέραιος N είναι αποθηκευμένος στον πίνακα, αρκεί να εξετάσουμε αν Table[N]=N. Μια συνάρτηση h, όπως η h(i)=i, που καθορίζει τη θέση ενός στοιχείου i στον πίνακα κατακερματισμού, ονομάζεται **συνάρτηση κατακερματισμού** (**hash function**).

Η συνάρτηση κατακερματισμού του παραπάνω παραδείγματος δουλεύει τέλεια, αφού ο χρόνος αναζήτησης του πίνακα για μια συγκεκριμένη τιμή είναι σταθερός και χρειάζεται να εξετάσουμε μόνο μία θέση του πίνακα. Από άποψη χρόνου, δηλαδή, η αποδοτικότητα είναι μεγάλη, όχι όμως και από άποψη χώρου. Από τις 100 διαθέσιμες θέσεις του πίνακα μόνο οι 10 χρησιμοποιούνται για τη αποθήκευση των στοιχείων, ενώ οι υπόλοιπες 90 παραμένουν αχρησιμοποίητες. Αυτό σημαίνει ότι χρησιμοποιείται μόνο το 10% του διαθέσιμου χώρου και έχουμε σπατάλη 90%.

Προκειμένου, λοιπόν, να αποφύγουμε τη σπατάλη χώρου, αφού για την αποθήκευση 10 τιμών απαιτούνται 10 θέσεις, μπορούμε να χρησιμοποιήσουμε έναν πίνακα HashTable 10 θέσεων με δείκτες 0..9. Προφανώς, η αρχική συνάρτηση κατακερματισμού h(i)=i δεν μπορεί πλέον να χρησιμοποιηθεί, γι' αυτό καταφεύγουμε στη μέθοδο της **διαίρεσης** (**division**).

Διαιρούμε, δηλαδή την τιμή του στοιχείου με τον αριθμό των θέσεων του πίνακα και το υπόλοιπο αυτής της διαίρεσης μας δίνει τη θέση στην οποία θα αποθηκευτεί το στοιχείο. Με λίγα λόγια, η συνάρτηση κατακερματισμού που χρησιμοποιείται είναι η:

$$h(i)=i \% 10$$

Επειδή, όμως, είναι συνήθως πιο βολικό οι τιμές κατακερματισμού να ξεκινούν από 1, αυξάνουμε το υπόλοιπο της διαίρεσης κατά 1, ώστε να προκύπτουν αριθμοί μεταξύ 1 και 10. Επομένως, για να βρούμε την τιμή κατακερματισμού για μια τιμή ακεραίου χρησιμοποιούμε τη συνάρτηση:

$$h(i)=(i \% 10) + 1$$

Η συνάρτηση αυτή δίνει πάντα έναν ακέραιο μεταξύ 1 και 10. Έτσι, για παράδειγμα, ο ακέραιος 34 θα αποθηκευτεί στη θέση HashTable[5], αφού h(34)=34 **%** 10 + 1=5, ο ακέραιος 21 θα αποθηκευτεί στη θέση HashTable[2], επειδή h(21)=21 **%** 10 + 1=2, κ.ο.κ. Παρακάτω φαίνεται ο πίνακας HashTable, ύστερα από την αποθήκευση των 34, 21, 79, 18, 5 και 56 στις θέσεις 5, 2, 10, 9, 6 και 7 αντίστοιχα.

HashTable

Θέση	**Αριθμός**
1	-1
2	21
3	-1
4	-1
5	34
6	5
7	56
8	-1
9	18
10	79

Συνήθως, όταν χρησιμοποιείται η τεχνική του κατακερματισμού, τα στοιχεία που αποθηκεύονται είναι εγγραφές. Ένα από τα πεδία κάθε τέτοιας εγγραφής χρησιμοποιείται για τον κατακερματισμό και γι' αυτό ονομάζεται **πεδίο κατακερματισμού** (**hash field**). Κι, επειδή, συνήθως, το πεδίο αυτό είναι το πεδίο-κλειδί της εγγραφής, ονομάζεται **κλειδί κατακερματισμού** (**hash key**).

Όταν οι τιμές των κλειδιών δεν είναι ακεραίου τύπου, όπως υποθέσαμε παραπάνω, τότε μπορούν να μετασχηματιστούν σε ακεραίους και μετά να εκτελεστεί η διαίρεση. Μια συμβολοσειρά μπορεί να μετασχηματιστεί σε ακέραιο, αν πάρουμε τον αριθμητικό κωδικό

που της αντιστοιχεί. Αν, δηλαδή, το κλειδί είναι τύπος αλφαριθμητικού, μπορούμε να το μετατρέψουμε σε έναν ακέραιο χρησιμοποιώντας κάποια αριθμητική αναπαράσταση των χαρακτήρων του κλειδιού, π. χ τους κώδικες ASCII ή EBCDIC. Για παράδειγμα, αν η τιμή ενός κλειδιού είναι C1, τότε μπορούμε να το μετατρέψουμε σε 6749, γιατί 67 είναι ο ASCII κωδικός του C και 49 ο αντίστοιχος του 1.

Ένας άλλος τρόπος είναι να επιλέξουμε ένα σταθερό τμήμα του κλειδιού και να το διαιρέσουμε, όμως, σ' αυτήν την περίπτωση πρέπει να είμαστε πολύ προσεκτικοί το τμήμα του κλειδιού που θα διαλέξουμε να περιέχει όσο γίνεται λιγότερες επαναλαμβανόμενες τιμές για την πλειοψηφία των κλειδιών.

Έχουν αναπτυχθεί αρκετές τεχνικές για τον προσδιορισμό ικανοποιητικών συναρτήσεων κατακερματισμού, όπως η **διαίρεση**, που περιγράψαμε παραπάνω και η οποία θεωρείται η καλύτερη γενικού σκοπού μέθοδος μέχρι στιγμής, η **μετατροπή της ρίζας** (**radix conversion**), η **μέση του τετραγώνου** (**mid square**), η **αναδίπλωση**(**folding**), κ.α.

Η τεχνική κατακερματισμού που θα επιλεγεί καθορίζει τις τιμές κατακερματισμού που παράγονται από τις τιμές των κλειδιών των εγγραφών. Υπάρχει περίπτωση, όμως, διαφορετικά κλειδιά να κατακερματίζονται στην ίδια τιμή κατακερματισμού. Τέτοια κλειδιά ονομάζονται **συνώνυμα** (**synonyms**) και πολλές φορές οι εγγραφές των οποίων οι τιμές κλειδιών είναι συνώνυμα ονομάζονται κι αυτές συνώνυμα. Το σύνολο των συνωνύμων εξαρτάται από την τεχνική κατακερματισμού, δηλαδή μια αλλαγή στην τεχνική κατακερματισμού θα αλλάξει και το σύνολο των συνωνύμων.

Όταν ένα κλειδί κατακερματίζεται σε μια τιμή η οποία έχει ήδη παραχθεί από ένα άλλο κλειδί, δηλαδή όταν κατακερματίζονται συνώνυμα, εμφανίζεται το φαινόμενο της **σύγκρουσης** (**collision**). Στο παραπάνω παράδειγμα με τον πίνακα HashTable, αν θελήσουμε να αποθηκεύσουμε τον ακέραιο 26, θα δημιουργηθεί σύγκρουση, γιατί ο αριθμός αυτός πρέπει να μπει στη θέση h(26)=26 **%** 10 + 1=7, όπου όμως είναι αποθηκευμένος ήδη ο αριθμός 56.

Συνήθως, δεν είναι πρακτικό να χρησιμοποιούμε έναν πίνακα με μέγεθος ίσο με το πλήθος των στοιχείων που πρόκειται να αποθηκευτούν. Ακόμα κι όταν το μέγεθος του πίνακα είναι αρκετά μεγάλο και μπορεί να χωρέσει περισσότερα στοιχεία από όσα πρόκειται να αποθηκευτούν σ' αυτόν, οι συγκρούσεις είναι πολύ πιθανές. Για παράδειγμα, για έναν πίνακα κατακερματισμού με 365 θέσεις, στον οποίο πρόκειται να αποθηκευτούν 23 τυχαία επιλεγμένα στοιχεία, η πιθανότητα εμφάνισης σύγκρουσης είναι 0.5! Κατά συνέπεια, είναι παράλογο να περιμένουμε από ένα σχήμα κατακερματισμού να αποτρέπει εντελώς τις συγκρούσεις. Αντιθέτως, θα πρέπει να είμαστε ικανοποιημένοι με πίνακες κατακερματισμού στους οποίους εμφανίζονται λίγες συγκρούσεις. Εμπειρικές μελέτες προτείνουν τη χρήση

πινάκων που έχουν μέγεθος περίπου 1,5 με 2 φορές το πλήθος των στοιχείων που θα αποθηκευτούν.

Οι πιο συνηθισμένες τεχνικές για το χειρισμό των συγκρούσεων είναι η **αλυσιδωτή σύνδεση** (**chaining**) και η **ανοιχτή διευθυνσιοδότηση** (**open addressing**).