ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

Δομές Δεδομένων (Εργ.)
Ακ. Έτος 2018-19
Διδάσκων: Ευάγγελος Σπύρου

Εργαστήριο 8 – Κατακερματισμός με χρήση αλυσιδωτής σύνδεσης

1. Στόχος του εργαστηρίου

Στόχος του όγδοου εργαστηρίου είναι η κατανόηση του κατακερματισμού με χρήση αλυσιδωτής σύνδεσης (chained hashing), καθώς και η υλοποίηση του σε γλώσσα C.

2. Κατακερματισμός

Οι συνήθεις μέθοδοι αναζήτησης όπως π.χ., η *σειριακή* αναζήτηση, η *δυαδική* αναζήτηση, η αναζήτηση *Fibonacci*, η αναζήτηση *παρεμβολής* και η αναζήτηση *άλματος* βρίσκουν το ζητούμενο κλειδί σε χρόνους $O(n)$, $O(\lg n)$, $O(\lg n)$, $O(\lg \lg n)$ και $O(n)$, αντίστοιχα. Ωστόσο, είναι δυνατόν να γίνει η αναζήτηση ενός κλειδιού με ακόμη λιγότερες συγκρίσεις, εφόσον δαπανηθεί περισσότερος χώρος για την αποθήκευση του πίνακα.

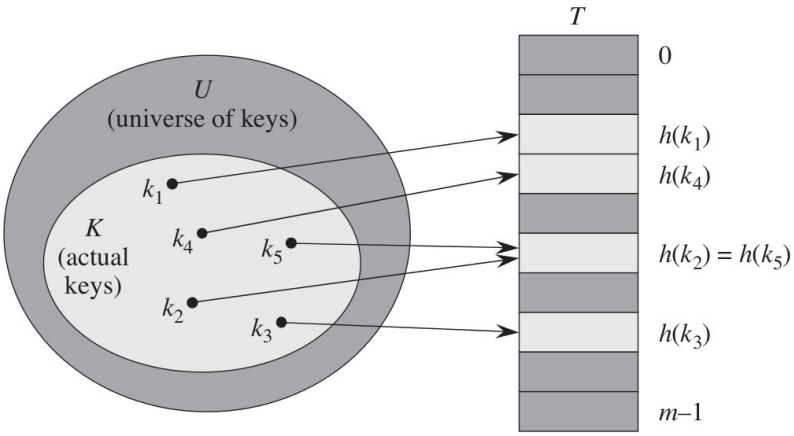
Πολλές εφαρμογές απαιτούν κάποιο δυναμικό σύνολο που να υποστηρίζει μόνο τις πράξεις ευρετηρίασης, δηλαδή τις εισαγωγή, διαγραφή και αναζήτηση. Ο κατακερματισμός (hashing) είναι μια μέθοδος για τη φύλαξη στοιχείων με βάση ένα κλειδί σε γραμμικές δομές δεδομένων (πίνακες, αρχεία) με στόχο τη γρήγορη ανεύρεσή τους. Σε αυτές τις περιπτώσεις μπορεί να χρησιμοποιηθεί ένας πίνακας διασποράς, ο οποίος αποτελεί μια δομή δεδομένων κατάλληλη για την υλοποίηση ευρετηρίων. Ένα ευρετήριο αποτελείται από θυρίδες και κάθε κλειδί αποθηκεύεται σε μία θυρίδα. Παρόλο που η αναζήτηση ενός στοιχείου σε έναν πίνακα διασποράς μπορεί να απαιτεί χρόνο $\Theta(n)$ (όπως η αλυσίδα), στην πράξη η διασπορά έχει εξαιρετικές επιδόσεις. Υπό κάποιες παραδοχές, ο χρόνος αναζήτησης γίνεται ακόμη και $O(1)$. Ο πίνακας διασποράς αποτελεί γενίκευση της συνήθους συστοιχίας.

3. Πίνακες Διασποράς

Με τη μέθοδο της διασποράς, το κλειδί με τιμή k αποθηκεύεται στη θυρίδα $h(k)$. Χρησιμοποιούμε μια συνάρτηση διασποράς h για να υπολογίσουμε τη θυρίδα από το κλειδί k. Η h αποτελεί μια απεικόνιση του χώρου αναφοράς U των κλειδιών στις θυρίδες ενός πίνακα διασποράς $T[0, \ldots, m-1]$. Η τιμή $h(k)$ ονομάζεται τιμή διασποράς του κλειδιού k, $h : U \rightarrow \{0, 1, \ldots, m-1\}$.

Το ζητούμενο με τη συνάρτηση διασποράς είναι να μειωθεί το πλήθος των θυρίδων που πρέπει να διαχειριστούμε. Αντί για $|U|$ τιμές, αρκεί να διαχειριστούμε m τιμές. Ανάλογα μειώνονται και οι απαιτήσεις σε αποθηκευτικό χώρο. Το μόνο κώλυμα είναι ότι στην ίδια θυρίδα ενδέχεται να γίνει διασπορά περισσότερων από ένα κλειδιών. Ένα τέτοιο συμβάν λέγεται σύμπτωση. Υπάρχουν αποτελεσματικές τεχνικές που αίρουν τις συγκρούσεις που προκαλούν οι συμπτώσεις.

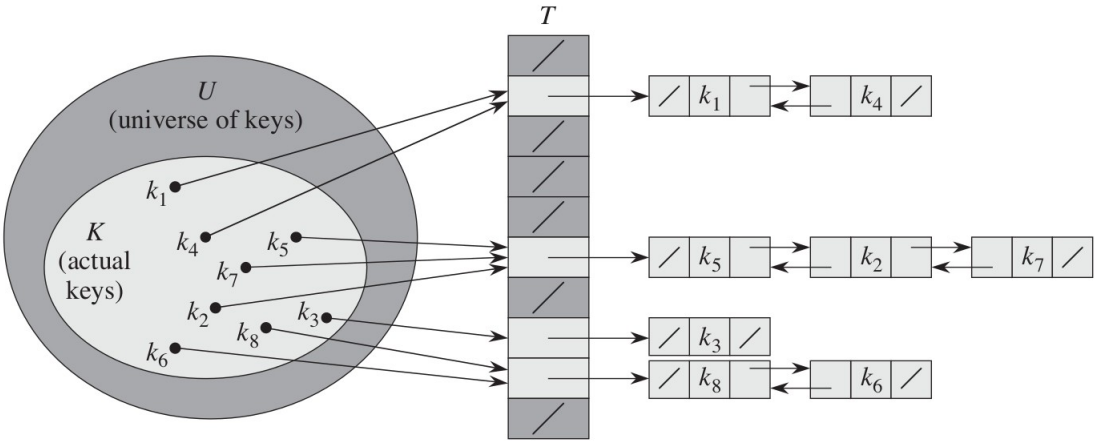
Δείτε π.χ. στο ακόλουθο σχήμα πώς υπάρχει σύγκρουση των κλειδιών k_2 και k_5.



Η ιδανική λύση θα ήταν η πλήρης αποφυγή των συμπτώσεων. Θα μπορούσε αυτό να επιτευχθεί με τη χρήση μιας κατάλληλης συνάρτησης *h*, π.χ. θα μπορούσε κάποιος να επιλέξει μια συνάρτηση *h*, η οποία να φαίνεται "τυχαία." Έτσι ίσως θα κατάφερνε να αποφύγει ή να ελαχιστοποιήσει τις συμπτώσεις. Ωστόσο, αφού $|U| > m$, σε κάθε περίπτωση θα υπάρχουν δύο κλειδιά που θα έχουν την ίδια τιμή διασποράς. Άρα η πλήρης αποφυγή των συμπτώσεων είναι αδύνατη. Χρειαζόμαστε μια μέθοδο για την άρση των συμπτώσεων που θα συμβούν τελικά.

4. Άρση των συμπτώσεων μέσω αλυσιδωτής σύνδεσης

Σύμφωνα με τη μεθοδολογία αυτή, τοποθετούμε όλα τα στοιχεία που διασπείρονται στην ίδια θυρίδα σε μια αλυσίδα. Η θυρίδα *j* περιέχει έναν δείκτη προς την κεφαλή της αλυσίδας όλων των αποθηκευμένων στοιχείων που έχουν διασπαρεί στην *j*. Αν δεν υπάρχει κανένα τέτοιο στοιχείο, η *j* περιέχει NULL. Δείτε π.χ. στο παρακάτω σχήμα πώς γίνεται η άρση των συγκρούσεων $h(k_1) = h(k_4)$ και $h(k_5) = h(k_2) = h(k_7)$.



Σημειώνεται ότι στην περίπτωση αυτή, ο χρόνος χειρότερης περίπτωσης για την εισαγωγή θα είναι O(*1*), για την αναζήτηση ανάλογος προς το μήκος της αλυσίδας, ενώ η διαγραφή θα μπορεί να πραγματοποιηθεί σε O(*1*).

5. Υλοποίηση κατακερματισμού με χρήση αλυσιδωτής σύνδεσης

Να υλοποιηθεί σε γλώσσα C μια δομή κατακερματισμού με χρήση αλυσιδωτής σύνδεσης, που να υποστηρίζει τις πράξεις της εισαγωγής, της αναζήτησης και της διαγραφής