Δομές Δεδομένων

Κατακερματισμός

Δημήτρης Μιχαήλ

Τμήμα Πληροφορικής και Τηλεματικής
Χαροκόπειο Πανεπιστήμιο

Λεξικό
Dictionary

Ένα **λεξικό** (dictionary) είναι ένας αφηρημένος τύπος δεδομένων (ΑΤΔ) που διατηρεί ένα σύνολο στοιχείων με κλειδιά και υποστηρίζει τις εξής λειτουργίες:

- insert(item): προσθήκη του item στο σύνολο
- delete(item): διαγραφή του item από το σύνολο
- search(key): επιστροφή του αντικειμένου με κλειδί key εαν υπάρχει

Υποθέτουμε πως τα κλειδιά έχουν μοναδικά κλειδιά ή πως αντικαθιστούμε αντικείμενα με ίδια κλειδιά.

Χρησιμότητα

Το λεξικό είναι μια από τις σημαντικότερες δομές δεδομένων στην επιστήμη των υπολογιστών.

- Υπάρχουν υλοποιημένα στις περισσότερες γλώσσες προγραμματισμού: C++, C#, Java, Python, Perl, Ruby, ...
- Σε μερικές από αυτές είναι ενσωματωμένα μέσα στην ίδια την γλώσσα.

Χρησιμότητα

Παραδείγματα

- μηχανές αναζήτησης web: λέξη $\rightarrow$ σελίδες που περιέχουν την λέξη
- μεταγλωττιστές: όνομα $\rightarrow$ μεταβλητή
- network routers: διεύθυνση IP $\rightarrow$ καλώδιο
- network server: port $\rightarrow$ εφαρμογή
- εικονική μνήμη: virtual address $\rightarrow$ physical address

Ισοζυγισμένα ΔΔΑ

Balanced BSTs

- Λεξικό με $\mathcal{O}(\log n)$ χρόνο για εισαγωγή, διαγραφή, αναζήτηση.
- Προυποθέτει σύνολο που είναι διατεταγμένο από μία σχέση $\leq$.
- Υποστηρίζουν όμως και πολλές άλλες λειτουργίες.

Ισοζυγισμένα ΔΔΑ

Balanced BSTs

- Λεξικό με $\mathcal{O}(\log n)$ χρόνο για εισαγωγή, διαγραφή, αναζήτηση.
- Προυποθέτει σύνολο που είναι διατεταγμένο από μία σχέση $\leq$.
- Υποστηρίζουν όμως και πολλές άλλες λειτουργίες.

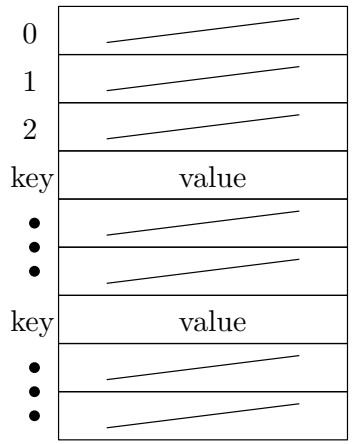
Στόχος

Θέλουμε $\mathcal{O}(1)$ χρόνο για κάθε λειτουργία.

Λεξικά

Η πιο απλή λύση, απευθείας πρόσβαση σε πίνακα:

- αποθηκεύουμε τα αντικείμενα σε ένα πίνακα στην διεύθυνση του κλειδιού (τυχαία προσπέλαση)
- προβλήματα:
  1. τα κλειδιά πρέπει να είναι μη αρνητικά (αλλιώς δύο πίνακες)
  2. μεγάλο εύρος κλειδιών $\rightarrow$ πολύ χώρος
     π.χ ένα κλειδί 10^{15} είναι ένα Petabyte



Κατακερματισμός

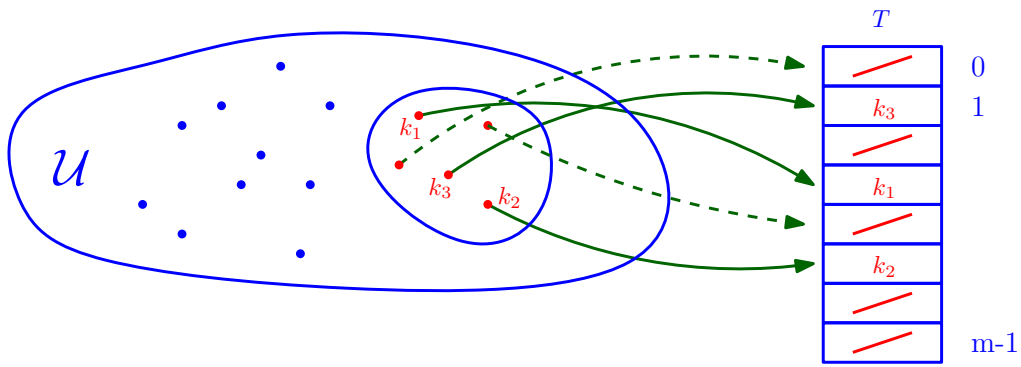
Μειώνουμε το σύνολο $\mathcal{U}$ όλων των κλειδιών (π.χ ακέραιοι) σε ένα μικρότερο πιο ρεαλιστικό μέγεθος m ενός πίνακα.

Ιδέα

$m \approx n =$ αριθμός κλειδιών αποθηκευμένα στο λεξικό

Συνάρτηση Κατακερματισμού (hash function)

$h : \mathcal{U} \mapsto \{0, 1, \ldots, m-1\}$



Δύο κλειδιά x και y *συγκρούονται (collide)* εάν $h(x) = h(y)$.

Σύγκρουση

Όταν για δύο κλειδιά k_1 και k_2 ισχύει $h(k_1) = h(k_2)$.

Επίλυση Συγκρούσεων

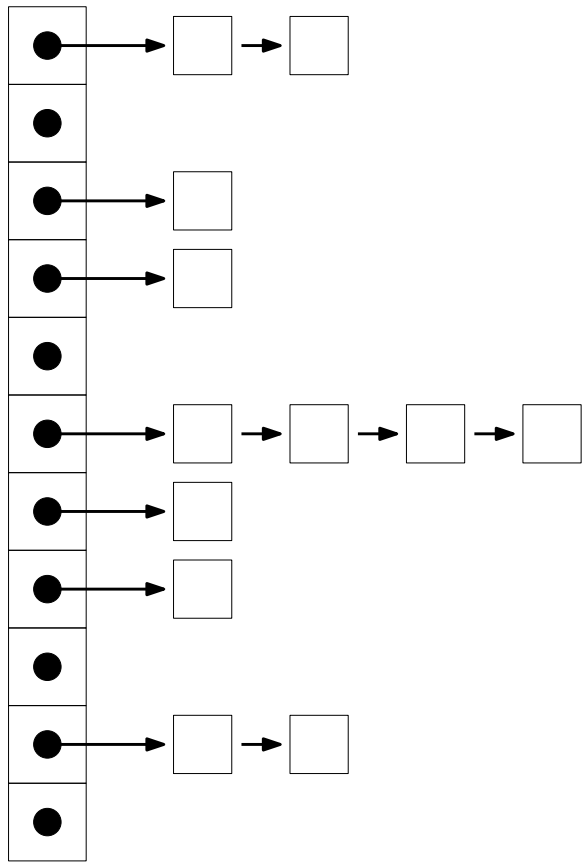
Θα δούμε δύο τρόπους επίλυσης:

1. Αλυσιδωτός κατακερματισμός (chaining)
2. Ανοιχτή διευθυνσιοδότηση (open addressing)

Αλυσιδωτός Κατακερματισμός

Hashing with Chaining

Λύνουμε το πρόβλημα της σύγκρουσης κρατώντας μια λίστα στοιχείων σε κάθε διεύθυνση του πίνακα κατακερματισμού.



Αλυσιδωτός Κατακερματισμός

Hashing with Chaining

- εισαγωγή στοιχείου e με κλειδί key
  - εισαγωγή στην λίστα $T[h(key)]$
  - σταθερός χρόνος (θεωρώντας πως η συνάρτηση κατακερματισμού πέρνει επίσης σταθερό χρόνο)

Αλυσιδωτός Κατακερματισμός

Hashing with Chaining

- εισαγωγή στοιχείου e με κλειδί key
  - εισαγωγή στην λίστα $T[h(key)]$
  - σταθερός χρόνος (θεωρώντας πως η συνάρτηση κατακερματισμού πέρνει επίσης σταθερό χρόνο)
- διαγραφή στοιχείου με κλειδί key
  - αναζήτηση στη λίστα $T[h(key)]$ και διαγραφή
  - εάν δεν υπάρχει και όλα τα στοιχεία είναι στην θέση $h(key)$ τότε στην χειρότερη περίπτωση $\Theta(n)$

Αλυσιδωτός Κατακερματισμός

Hashing with Chaining

- εισαγωγή στοιχείου e με κλειδί key
  - εισαγωγή στην λίστα $T[h(key)]$
  - σταθερός χρόνος (θεωρώντας πως η συνάρτηση κατακερματισμού πέρνει επίσης σταθερό χρόνο)
- διαγραφή στοιχείου με κλειδί key
  - αναζήτηση στη λίστα $T[h(key)]$ και διαγραφή
  - εάν δεν υπάρχει και όλα τα στοιχεία είναι στην θέση $h(key)$ τότε στην χειρότερη περίπτωση $\Theta(n)$
- αναζήτηση στοιχείου με κλειδί key
  - αναζήτηση στη λίστα $T[h(key)]$
  - εάν δεν υπάρχει και όλα τα στοιχεία είναι στην θέση $h(key)$ τότε στην χειρότερη περίπτωση $\Theta(n)$

Αλυσιδωτός Κατακερματισμός

Hashing with Chaining

- εισαγωγή στοιχείου e με κλειδί key
  - εισαγωγή στην λίστα $T[h(key)]$
  - σταθερός χρόνος (θεωρώντας πως η συνάρτηση κατακερματισμού πέρνει επίσης σταθερό χρόνο)
- διαγραφή στοιχείου με κλειδί key
  - αναζήτηση στη λίστα $T[h(key)]$ και διαγραφή
  - εάν δεν υπάρχει και όλα τα στοιχεία είναι στην θέση $h(key)$ τότε στην χειρότερη περίπτωση $\Theta(n)$
- αναζήτηση στοιχείου με κλειδί key
  - αναζήτηση στη λίστα $T[h(key)]$
  - εάν δεν υπάρχει και όλα τα στοιχεία είναι στην θέση $h(key)$ τότε στην χειρότερη περίπτωση $\Theta(n)$

Χώρος $\mathcal{O}(n+m)$

Συναρτήσεις Κατακερματισμού

Η συνάρτηση κατακερματισμού παίζει πάρα πολύ σημαντικό ρόλο στην απόδοση.

Συναρτήσεις Κατακερματισμού

Η συνάρτηση κατακερματισμού παίζει πάρα πολύ σημαντικό ρόλο στην απόδοση.

Επιθυμητές Ιδιότητες Κατακερματισμού

1. τα κλειδιά πρέπει να διαμοιράζονται ομοιόμορφα ώστε να μην έχουμε πολλές συγκρούσεις
   - ο αριθμός των συγκρούσεων επηρεάζει τον χρόνο αναζήτησης και διαγραφής

Συναρτήσεις Κατακερματισμού

Η συνάρτηση κατακερματισμού παίζει πάρα πολύ σημαντικό ρόλο στην απόδοση.

Επιθυμητές Ιδιότητες Κατακερματισμού

1. τα κλειδιά πρέπει να διαμοιράζονται ομοιόμορφα ώστε να μην έχουμε πολλές συγκρούσεις
   - ο αριθμός των συγκρούσεων επηρεάζει τον χρόνο αναζήτησης και διαγραφής
2. $m = \mathcal{O}(n)$
   - θέλουμε να ισχύει η ιδιότητα (1) χωρίς το μέγεθος του πίνακα m να είναι πολύ μεγαλύτερο από τον αριθμό των στοιχείων n.

Συναρτήσεις Κατακερματισμού

Η συνάρτηση κατακερματισμού παίζει πάρα πολύ σημαντικό ρόλο στην απόδοση.

Επιθυμητές Ιδιότητες Κατακερματισμού

1. τα κλειδιά πρέπει να διαμοιράζονται ομοιόμορφα ώστε να μην έχουμε πολλές συγκρούσεις
   - ο αριθμός των συγκρούσεων επηρεάζει τον χρόνο αναζήτησης και διαγραφής
2. $m = \mathcal{O}(n)$
   - θέλουμε να ισχύει η ιδιότητα (1) χωρίς το μέγεθος του πίνακα m να είναι πολύ μεγαλύτερο από τον αριθμό των στοιχείων n.
3. η συνάρτηση h πρέπει να είναι πολύ γρήγορη, θέλουμε η συνάρτηση να υπολογίζεται σε σταθερό χρόνο

Θεώρημα

Για κάθε συνάρτηση κατακερματισμού h, εάν $|\mathcal{U}| > (n-1) \cdot m + 1$, υπάρχει ένα σύνολο S μεγέθους n που όλα τα στοιχεία του κατακερματίζονται στην ίδια θέση.

Απόδειξη

Απευθείας από την αρχή του περιστερώνα. □

Θεώρημα

Για κάθε συνάρτηση κατακερματισμού h, εάν $|\mathcal{U}| > (n-1) \cdot m + 1$, υπάρχει ένα σύνολο S μεγέθους n που όλα τα στοιχεία του κατακερματίζονται στην ίδια θέση.

Απόδειξη

Απευθείας από την αρχή του περιστερώνα. □

- Εμείς θέλουμε μια συνάρτηση καλή για κάθε σύνολο στοιχείων, το οποίο δεν γίνεται σύμφωνα με το παραπάνω θεώρημα.
- Στην πράξη για μερικά τυπικά σύνολα S υπάρχουν απλές συναρτήσεις κατακερματισμού που λειτουργούν καλά.
- θα δούμε μερικές στο τέλος του κεφαλαίου.

Θεώρημα

Για κάθε συνάρτηση κατακερματισμού h, εάν $|\mathcal{U}| > (n-1) \cdot m + 1$, υπάρχει ένα σύνολο S μεγέθους n που όλα τα στοιχεία του κατακερματίζονται στην ίδια θέση.

Απόδειξη

Απευθείας από την αρχή του περιστερώνα. □

- Εμείς θέλουμε μια συνάρτηση καλή για κάθε σύνολο στοιχείων, το οποίο δεν γίνεται σύμφωνα με το παραπάνω θεώρημα.
- Για να λύσουμε το πρόβλημα για οποιοδήποτε σύνολο προσθέτουμε **τυχαίοτητα**.
- Από ένα σύνολο $\mathcal{H}$ από συναρτήσεις επιλέγουμε μία συνάρτηση h τυχαία.

Συναρτήσεις Κατακερματισμού

Η συνάρτηση κατακερματισμού παίζει πάρα πολύ σημαντικό ρόλο στην απόδοση.

- Έστω $\mathcal{H}$ το σύνολο *όλων* των συναρτήσεων από το $\mathcal{U}$ στο $0 \ldots m-1$.
- Ας υποθέσουμε πως η συνάρτηση h διαλέγεται τυχαία από το H.

Θεώρημα

Εάν n στοιχεία είναι αποθηκευμένα σε ένα πίνακα κατακερματισμού με m θέσεις με την χρήση μιας τυχαίας συνάρτησης κατακερματισμού, τότε ο αναμενόμενος χρόνος εκτέλεσης της *διαγραφής* ή *αναζήτησης* είναι $\mathcal{O}(1+\frac{n}{m})$.

Τυχαία Συνάρτηση Κατακερματισμού

Θεώρημα

Εάν n στοιχεία είναι αποθηκευμένα σε ένα πίνακα κατακερματισμού με m θέσεις με την χρήση μιας τυχαίας συνάρτησης κατακερματισμού, τότε ο αναμενόμενος χρόνος εκτέλεσης της *διαγραφής* ή *αναζήτησης* είναι $\mathcal{O}(1 + \frac{n}{m})$.

Απόδειξη

Για ένα συγκεκριμένο κλειδί k οι συναρτήσεις αναζήτησης και διαγραφής χρειάζονται χρόνο σταθερό συν τον χρόνο να διαβάσουν την λίστα $T[h(k)]$.

Τυχαία Συνάρτηση Κατακερματισμού

Θεώρημα

Εάν n στοιχεία είναι αποθηκευμένα σε ένα πίνακα κατακερματισμού με m θέσεις με την χρήση μιας τυχαίας συνάρτησης κατακερματισμού, τότε ο αναμενόμενος χρόνος εκτέλεσης της *διαγραφής* ή *αναζήτησης* είναι $\mathcal{O}(1 + \frac{n}{m})$.

Απόδειξη

Για ένα συγκεκριμένο κλειδί k οι συναρτήσεις αναζήτησης και διαγραφής χρειάζονται χρόνο σταθερό συν τον χρόνο να διαβάσουν την λίστα $T[h(k)]$.
Ο αναμενόμενος χρόνος είναι $\mathcal{O}(1 + E[X])$ όπου X είναι η τυχαία μεταβλητή που υποδηλώνει το μέγεθος της λίστας $T[h(k)]$.

Τυχαία Συνάρτηση Κατακερματισμού

Θεώρημα

Εάν n στοιχεία είναι αποθηκευμένα σε ένα πίνακα κατακερματισμού με m θέσεις με την χρήση μιας τυχαίας συνάρτησης κατακερματισμού, τότε ο αναμενόμενος χρόνος εκτέλεσης της *διαγραφής* ή *αναζήτησης* είναι $\mathcal{O}(1 + \frac{n}{m})$.

Απόδειξη

Για ένα συγκεκριμένο κλειδί k οι συναρτήσεις αναζήτησης και διαγραφής χρειάζονται χρόνο σταθερό συν τον χρόνο να διαβάσουν την λίστα $T[h(k)]$.
Ο αναμενόμενος χρόνος είναι $\mathcal{O}(1 + E[X])$ όπου X είναι η τυχαία μεταβλητή που υποδηλώνει το μέγεθος της λίστας $T[h(k)]$.

Έστω S το σύνολο των n στοιχείων που είναι αποθηκευμένα. Για κάθε $e \in S$, έστω X_e η τυχαία μεταβλητή (indicator) όπου $X_e = 1$ εάν $h(e) = h(k)$ και $X_e = 0$ αλλιώς.

Τυχαία Συνάρτηση Κατακερματισμού

Θεώρημα

Εάν n στοιχεία είναι αποθηκευμένα σε ένα πίνακα κατακερματισμού με m θέσεις με την χρήση μιας τυχαίας συνάρτησης κατακερματισμού, τότε ο αναμενόμενος χρόνος εκτέλεσης της *διαγραφής* ή *αναζήτησης* είναι $\mathcal{O}(1 + \frac{n}{m})$.

Απόδειξη

Για ένα συγκεκριμένο κλειδί k οι συναρτήσεις αναζήτησης και διαγραφής χρειάζονται χρόνο σταθερό συν τον χρόνο να διαβάσουν την λίστα $T[h(k)]$.
Ο αναμενόμενος χρόνος είναι $\mathcal{O}(1 + E[X])$ όπου X είναι η τυχαία μεταβλητή που υποδηλώνει το μέγεθος της λίστας $T[h(k)]$.

Έστω S το σύνολο των n στοιχείων που είναι αποθηκευμένα. Για κάθε $e \in S$, έστω X_e η τυχαία μεταβλητή (indicator) όπου $X_e = 1$ εάν $h(e) = h(k)$ και $X_e = 0$ αλλιώς.

Έχουμε $X = \sum_{e \in S} X_e$.

Τυχαία Συνάρτηση Κατακερματισμού

Θεώρημα

Εάν n στοιχεία είναι αποθηκευμένα σε ένα πίνακα κατακερματισμού με m θέσεις με την χρήση μιας τυχαίας συνάρτησης κατακερματισμού, τότε ο αναμενόμενος χρόνος εκτέλεσης της *διαγραφής* ή *αναζήτησης* είναι $\mathcal{O}(1 + \frac{n}{m})$.

Απόδειξη

Για ένα συγκεκριμένο κλειδί k οι συναρτήσεις αναζήτησης και διαγραφής χρειάζονται χρόνο σταθερό συν τον χρόνο να διαβάσουν την λίστα $T[h(k)]$.
Ο αναμενόμενος χρόνος είναι $\mathcal{O}(1 + E[X])$ όπου X είναι η τυχαία μεταβλητή που υποδηλώνει το μέγεθος της λίστας $T[h(k)]$.

Έστω S το σύνολο των n στοιχείων που είναι αποθηκευμένα. Για κάθε $e \in S$, έστω X_e η τυχαία μεταβλητή (indicator) όπου $X_e = 1$ εάν $h(e) = h(k)$ και $X_e = 0$ αλλιώς.

Έχουμε $X = \sum_{e \in S} X_e$. Χρησιμοποιώντας την γραμμικότητα της μέσης τιμής πέρνουμε

$$E[X] = E[\sum_{e \in S} X_e] = \sum_{e \in S} E[X_e] = \sum_{e \in S} prob(X_e = 1).$$

Τυχαία Συνάρτηση Κατακερματισμού

Θεώρημα

Εάν n στοιχεία είναι αποθηκευμένα σε ένα πίνακα κατακερματισμού με m θέσεις με την χρήση μιας τυχαίας συνάρτησης κατακερματισμού, τότε ο αναμενόμενος χρόνος εκτέλεσης της *διαγραφής* ή *αναζήτησης* είναι $\mathcal{O}(1 + \frac{n}{m})$.

Απόδειξη

Για ένα συγκεκριμένο κλειδί k οι συναρτήσεις αναζήτησης και διαγραφής χρειάζονται χρόνο σταθερό συν τον χρόνο να διαβάσουν την λίστα $T[h(k)]$.
Ο αναμενόμενος χρόνος είναι $\mathcal{O}(1 + E[X])$ όπου X είναι η τυχαία μεταβλητή που υποδηλώνει το μέγεθος της λίστας $T[h(k)]$.

Έστω S το σύνολο των n στοιχείων που είναι αποθηκευμένα. Για κάθε $e \in S$, έστω X_e η τυχαία μεταβλητή (indicator) όπου $X_e = 1$ εάν $h(e) = h(k)$ και $X_e = 0$ αλλιώς.

Έχουμε $X = \sum_{e \in S} X_e$. Χρησιμοποιώντας την γραμμικότητα της μέσης τιμής πέρνουμε

$$E[X] = E[\sum_{e \in S} X_e] = \sum_{e \in S} E[X_e] = \sum_{e \in S} prob(X_e = 1).$$

Μια τυχαία συνάρτηση αντιστοιχεί ένα στοιχείο e σε όλες τις m θέσεις με ίδια πιθανότητα $1/m$ και άρα $E[X] = n/m$. □

Τυχαία Συνάρτηση Κατακερματισμού

Δυστυχώς η υπόθεση της τυχαίας συνάρτησης κατακερματισμού είναι τελείως αβάσιμη.

Έστω $\mathcal{H}$ όλες οι συναρτήσεις από το $\mathcal{U}$ στο $0 \ldots m-1$.

- Υπάρχουν $m^{|\mathcal{U}|}$ συναρτήσεις στο σύνολο $\mathcal{H}$.
- Για να διαλέξουμε μια συνάρτηση από το $\mathcal{H}$ χρειαζόμαστε

$$\log_2 m^{|\mathcal{U}|} = |\mathcal{U}| \cdot \log_2 m$$

  bits.

Έτσι καταργούμε τον στόχο μας να μειώσουμε τον χώρο από $|\mathcal{U}|$ σε n.

Στόχοι

- θέλουμε να διαλέγουμε τυχαία μία συνάρτηση από ένα μικρότερο σύνολο από συναρτήσεις,
- η επιλογή μιας συνάρτησης πρέπει να μπορεί να γίνει με σταθερό χώρο,
- πρέπει οι συναρτήσεις να υπολογίζονται εύκολα.

Ορισμός

Έστω c μια θετική σταθερά. Μια οικογένεια συναρτήσεων από το $\mathcal{U}$ στο $0 \ldots m-1$ λέγεται *c-καθολική* (c-universal) εάν δυο διαφορετικά κλειδιά συγκρούονται με πιθανότητα το πολύ $\frac{c}{m}$.

Ορισμός

Έστω c μια θετική σταθερά. Μια οικογένεια συναρτήσεων από το $\mathcal{U}$ στο $0 \ldots m-1$ λέγεται *c-καθολική* (c-universal) εάν δυο διαφορετικά κλειδιά συγκρούονται με πιθανότητα το πολύ $\frac{c}{m}$.

Για κάθε $x, y \in \mathcal{U}$ με $x \neq y$ έχουμε:

$$|h \in \mathcal{H} : h(x) = h(y)| \leq \frac{c}{m}|\mathcal{H}|.$$

Ορισμός

Έστω c μια θετική σταθερά. Μια οικογένεια συναρτήσεων από το $\mathcal{U}$ στο $0 \ldots m-1$ λέγεται *c-καθολική* (c-universal) εάν δυο διαφορετικά κλειδιά συγκρούονται με πιθανότητα το πολύ $\frac{c}{m}$.

Για κάθε $x, y \in \mathcal{U}$ με $x \neq y$ έχουμε:

$$|h \in \mathcal{H} : h(x) = h(y)| \leq \frac{c}{m}|\mathcal{H}|.$$

Με άλλα λόγια, για τυχαία $h \in \mathcal{H}$,

$$prob(h(x) = h(y)) \leq \frac{c}{m}$$

Ορισμός

Έστω c μια θετική σταθερά. Μια οικογένεια συναρτήσεων από το $\mathcal{U}$ στο $0 \ldots m-1$ λέγεται *c-καθολική* (c-universal) εάν δυο διαφορετικά κλειδιά συγκρούονται με πιθανότητα το πολύ $\frac{c}{m}$.

Για κάθε $x, y \in \mathcal{U}$ με $x \neq y$ έχουμε:

$$|h \in \mathcal{H} : h(x) = h(y)| \leq \frac{c}{m}|\mathcal{H}|.$$

Με άλλα λόγια, για τυχαία $h \in \mathcal{H}$,

$$prob(h(x) = h(y)) \leq \frac{c}{m}$$

Μία *c-καθολική* συνάρτηση λειτουργεί μια χαρά στην απόδειξη που κάναμε με τις τυχαίες συναρτήσεις.

Καθολική Συνάρτηση Κατακερματισμού

Ερώτηση

Μπορούμε να κατασκευάσουμε μία c-καθολική οικογένεια συναρτήσεων;

Καθολική Συνάρτηση Κατακερματισμού

Ερώτηση

Μπορούμε να κατασκευάσουμε μία c-καθολική οικογένεια συναρτήσεων;

Η απάντηση είναι ΝΑΙ!

Έστω:

- Τα κλειδιά έχουν μήκος u bits.
- Ο πίνακας T έχει μέγεθος 2^b όπου b είναι ο αριθμός των bits για την διευθυνσιοδότηση του.

Έστω:

- Τα κλειδιά έχουν μήκος u bits.
- Ο πίνακας T έχει μέγεθος 2^b όπου b είναι ο αριθμός των bits για την διευθυνσιοδότηση του.

Τότε:

- Επιλέγουμε έναν τυχαίο 0/1 πίνακα h διαστάσεων $b \times u$
- Ορίζουμε $h(x) = h \cdot x$ όπου κάνουμε πρόσθεση με υπόλοιπο 2 (mod 2).

Έστω πως:

- τα κλειδιά μας έχουν μήκος $u = 8$ bits (π.χ 256 χαρακτήρες του πίνακα ASCII).
- ο πίνακας T έχει μέγεθος $2^4 = 16$ και άρα χρειαζόμαστε $b = 4$ bits για την διευθυνσιοδότηση του.

$$h(x) = h \cdot x = \begin{pmatrix} 0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} 0 \\ 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \end{pmatrix}$$

- Επιλέγουμε έναν τυχαίο 0/1 πίνακα h διαστάσεων 4×8
- Ορίζουμε $h(x) = h \cdot x$ όπου κάνουμε πρόσθεση με υπόλοιπο 2 (mod 2).

Θεώρημα

Για $x \neq y$, $Pr_h[h(x) = h(y)] = \frac{1}{2^b}$.

Απόδειξη

Καταρχήν, τι σημαίνει ο πολλαπλασιασμός h επί x; Μπορούμε να σκεφτούμε πως προσθέτουμε κάποιες στήλες του h (πρόσθεση mod 2) και τα 1 του x μας λένε ποιες στήλες να προσθέσουμε.

Θεώρημα

Για $x \neq y$, $Pr_h[h(x) = h(y)] = \frac{1}{2^b}$.

Απόδειξη

Καταρχήν, τι σημαίνει ο πολλαπλασιασμός h επί x; Μπορούμε να σκεφτούμε πως προσθέτουμε κάποιες στήλες του h (πρόσθεση mod 2) και τα 1 του x μας λένε ποιες στήλες να προσθέσουμε.

Έστω οποιοδήποτε ζευγάρι κλειδιών x, y ώστε $x \neq y$. Πρέπει να διαφέρουν κάπου, ας υποθέσουμε λοιπόν πως $x_i = 0$ και $y_i = 1$ για κάποια θέση i.

Θεώρημα

Για $x \neq y$, $Pr_h[h(x) = h(y)] = \frac{1}{2^b}$.

Απόδειξη

Καταρχήν, τι σημαίνει ο πολλαπλασιασμός h επί x; Μπορούμε να σκεφτούμε πως προσθέτουμε κάποιες στήλες του h (πρόσθεση mod 2) και τα 1 του x μας λένε ποιες στήλες να προσθέσουμε.

Έστω οποιοδήποτε ζευγάρι κλειδιών x, y ώστε $x \neq y$. Πρέπει να διαφέρουν κάπου, ας υποθέσουμε λοιπόν πως $x_i = 0$ και $y_i = 1$ για κάποια θέση i.

Φανταστείτε πως διαλέγουμε πρώτα όλο τον πίνακα h εκτός από την i στήλη. Σε σχέση με τις αναπομείναντες επιλογές η $h(x)$ είναι σταθερή μιας και ισχύει $x_i = 0$.

Θεώρημα

Για $x \neq y$, $Pr_h[h(x) = h(y)] = \frac{1}{2^b}$.

Απόδειξη

Καταρχήν, τι σημαίνει ο πολλαπλασιασμός h επί x; Μπορούμε να σκεφτούμε πως προσθέτουμε κάποιες στήλες του h (πρόσθεση mod 2) και τα 1 του x μας λένε ποιες στήλες να προσθέσουμε.

Έστω οποιοδήποτε ζευγάρι κλειδιών x, y ώστε $x \neq y$. Πρέπει να διαφέρουν κάπου, ας υποθέσουμε λοιπόν πως $x_i = 0$ και $y_i = 1$ για κάποια θέση i.

Φανταστείτε πως διαλέγουμε πρώτα όλο τον πίνακα h εκτός από την i στήλη. Σε σχέση με τις αναπομείναντες επιλογές η $h(x)$ είναι σταθερή μιας και ισχύει $x_i = 0$.

Τώρα, κάθε μία από τις 2^b διαφορετικές επιλογές για την i στήλη, δίνει διαφορετική τιμή στην $h(y)$ (κάθε φορά που αλλάζει ένα bit στην στήλη, αλλάζει και το αντίστοιχο bit στην $h(y)$).

Θεώρημα

Για $x \neq y$, $Pr_h[h(x) = h(y)] = \frac{1}{2^b}$.

Απόδειξη

Καταρχήν, τι σημαίνει ο πολλαπλασιασμός h επί x; Μπορούμε να σκεφτούμε πως προσθέτουμε κάποιες στήλες του h (πρόσθεση mod 2) και τα 1 του x μας λένε ποιες στήλες να προσθέσουμε.

Έστω οποιοδήποτε ζευγάρι κλειδιών x, y ώστε $x \neq y$. Πρέπει να διαφέρουν κάπου, ας υποθέσουμε λοιπόν πως $x_i = 0$ και $y_i = 1$ για κάποια θέση i.

Φανταστείτε πως διαλέγουμε πρώτα όλο τον πίνακα h εκτός από την i στήλη. Σε σχέση με τις αναπομείναντες επιλογές η $h(x)$ είναι σταθερή μιας και ισχύει $x_i = 0$.

Τώρα, κάθε μία από τις 2^b διαφορετικές επιλογές για την i στήλη, δίνει διαφορετική τιμή στην $h(y)$ (κάθε φορά που αλλάζει ένα bit στην στήλη, αλλάζει και το αντίστοιχο bit στην $h(y)$).

Άρα υπάρχει ακριβώς $\frac{1}{2^b}$ πιθανότητα να ισχύει $h(x) = h(y)$.

□

Μέγεθος Πίνακα

Αλυσιδωτός Κατακερματισμός

Εαν υποθέσουμε καθολική συνάρτηση κατακερματισμού, τότε έχουμε μέσο χρόνο αναζήτησης/διαγραφής: $\mathcal{O}(1 + \frac{n}{m})$.

Μέγεθος Πίνακα

Αλυσιδωτός Κατακερματισμός

Εαν υποθέσουμε καθολική συνάρτηση κατακερματισμού, τότε έχουμε μέσο χρόνο αναζήτησης/διαγραφής: $\mathcal{O}(1 + \frac{n}{m})$.

Τι μέγεθος πίνακα να διαλέξουμε

- θέλουμε $m = \mathcal{O}(n)$, δηλαδή μέσο χρόνο αναζήτησης/διαγραφής $\mathcal{O}(1)$
- δεν ξέρουμε το n κατά την δημιουργία του πίνακα
- πολύ μικρό $m \Rightarrow$ αργή αναζήτηση/διαγραφή
- πολύ μεγάλο $m \Rightarrow$ χάσιμο χώρου

Επανακατακερματισμός
Rehashing

Ιδέα

- αρχικά μικρό m (σταθερά)
- αύξηση (& μείωση) ανάλογα με την χρήση

Επανακατακερματισμός
Rehashing

Ιδέα

- αρχικά μικρό m (σταθερά)
- αύξηση (& μείωση) ανάλογα με την χρήση

Επανακατακερματισμός

Για να μεγαλώσει ή να μικρύνει ο πίνακας κατακερματισμού πρέπει να αλλάξει η συνάρτηση κατακερματισμού.

- επιλέγουμε νέα συνάρτηση κατακερματισμού από το σύνολο $\mathcal{H}$
- ο πίνακας πρέπει να φτιαχτεί από την αρχή
- κάθε αντικείμενο του παλιού πίνακα κατακερματισμού προστίθεται στον καινούριο πίνακα

Επανακατακερματισμός
Rehashing

Έχοντας μια καθολική συνάρτηση κατακερματισμού μπορούμε να έχουμε γραμμικό χώρο και σταθερό αναμενόμενο χρόνο όλων των λειτουργιών του πίνακα κατακερματισμού

1. φροντίζουμε πάντα $m = \Theta(n)$
2. εαν το n γίνει $> m$ τότε διπλασιάζουμε το μέγεθος του πίνακα
3. εαν το n γίνει $< \frac{1}{4}m$ υποδιπλασιάζουμε το μέγεθος του πίνακα

Επανακατακερματισμός
Rehashing

Έχοντας μια καθολική συνάρτηση κατακερματισμού μπορούμε να έχουμε γραμμικό χώρο και σταθερό αναμενόμενο χρόνο όλων των λειτουργιών του πίνακα κατακερματισμού

1. φροντίζουμε πάντα $m = \Theta(n)$
2. εαν το n γίνει $> m$ τότε διπλασιάζουμε το μέγεθος του πίνακα
3. εαν το n γίνει $< \frac{1}{4}m$ υποδιπλασιάζουμε το μέγεθος του πίνακα

Κάθε φορά που αλλάζουμε το μέγεθος του πίνακα, διαλέγουμε μια καινούρια συνάρτηση κατακερματισμού και υπολογίζουμε τις θέσεις όλων των στοιχείων με βάση τη νέα συνάρτηση.

Επανακατακερματισμός
Rehashing

Έχοντας μια καθολική συνάρτηση κατακερματισμού μπορούμε να έχουμε γραμμικό χώρο και σταθερό αναμενόμενο χρόνο όλων των λειτουργιών του πίνακα κατακερματισμού

1. φροντίζουμε πάντα $m = \Theta(n)$
2. εαν το n γίνει $> m$ τότε διπλασιάζουμε το μέγεθος του πίνακα
3. εαν το n γίνει $< \frac{1}{4}m$ υποδιπλασιάζουμε το μέγεθος του πίνακα

Κάθε φορά που αλλάζουμε το μέγεθος του πίνακα, διαλέγουμε μια καινούρια συνάρτηση κατακερματισμού και υπολογίζουμε τις θέσεις όλων των στοιχείων με βάση τη νέα συνάρτηση.

Μερικές λειτουργίες κοστίζουν γραμμικό χρόνο αλλά ο μέσος όρος είναι $\mathcal{O}(1)$

Ανοιχτή Διευθυνσιοδότηση
Open Addressing

1. όλα τα στοιχεία είναι απευθείας στον πίνακα, όχι σύνδεσμοι
2. υποχρεωτικά $n < m$
3. βασική ιδέα: εάν μία θέση είναι γεμάτη, αναζήτησε άλλη

Επίλυση Σύγκρουσης

Σε περίπτωση που η θέση $T[h(key)]$ έχει ήδη κάποιο στοιχείο αναζητάμε κάποια ελεύθερη θέση με βάση κάποια συγκεκριμένη στρατηγική.

Γραμμική Διερεύνηση

Linear Probing

Η απλούστερη μέθοδος ανοιχτής διευθυνσιοδότησης ονομάζεται *γραμμική διερεύνηση*.

Ο πίνακας κατακερματισμού αποθηκεύει όλα τα στοιχεία. Κενές θέσεις συμβολίζονται με κάποιο ειδικό σύμβολο, π.χ $\perp$.

- Ένα στοιχείο e αποθηκεύεται στην θέση $t[h(e)]$ ή στην πρώτη κενή προς τα δεξιά
- εάν το e είναι στην θέση $t[i]$ με $i > h(e)$ πρέπει υποχρεωτικά οι θέσεις $h(e)$ μέχρι και $i - 1$ να μην είναι κενές ($\perp$).

Γραμμική Διερεύνηση

Linear Probing

Εισαγωγή του στοιχείου e:

- ψάχνουμε στον πίνακα ξεκινώντας από την θέση $t[h(e)]$ προς τα δεξιά μέχρι να βρούμε ένα $\perp$. Εκεί αποθηκεύουμε το στοιχείο e.

Αναζήτηση του στοιχείου e:

- ψάχνουμε στον πίνακα ξεκινώντας από την θέση $t[h(e)]$ προς τα δεξιά μέχρι να βρούμε το e ή ένα $\perp$.

Και στις δύο περιπτώσεις χρησιμοποιούμε τον πίνακα ως κυκλικό πίνακα, καθώς κινούμαστε προς τα δεξιά.

Η διαγραφή ενός στοιχείου e είναι η μόνη δύσκολη διαδικασία:

- αφού βρούμε το e δεν μπορούμε απλά να το διαγράψουμε γράφοντας $\perp$ στην θέση του
- μπορεί να δημιουργήσουμε πρόβλημα σε άλλα στοιχεία

π.χ
έστω τα στοιχεία x, y, z τα οποία προστίθενται με αυτή την σειρά και έστω $h(x) = h(z)$ και $h(y) = h(x) + 1$.

Το z αποθηκεύεται στην θέση $h(x) + 2$. Όταν θέλουμε να σβήσουμε το y δεν μπορούμε απλά να γράψουμε στην θέση $h(x) + 1$ το $\perp$ μιας και θα κάνει το στοιχείο z μη προσβάσιμο.

Υπάρχουν 3 λύσεις:

1. να μην επιτρέπουμε διαγραφές
2. μπορούμε να σημαδέψουμε το στοιχείο e αλλά να μην το αφαιρέσουμε. Οι αναζητήσεις μπορούν να σταματάνε σε $\perp$ αλλά όχι σε σημαδεμένο στοιχείο.
3. να διορθώσουμε την κατάσταση

Η δεύτερη περίπτωση παρουσιάζει το πρόβλημα πως ο πίνακας γεμίζει και τελικά γίνονται αργές οι αναζητήσεις.

Έστω πως θέλουμε να διαγράψουμε το στοιχείο στην θέση i.

- γράφουμε στην θέση του το $\perp$ αφήνοντας ένα κενό
- στην συνέχεια ψάχνουμε στα δεξιά για να δούμε εαν χαλάσαμε τίποτα, θέτουμε $j = i + 1$
- εάν $t[j] = \perp$ είμαστε καλά και τερματίζουμε
- αλλιώς έστω f το στοιχείο που είναι αποθηκευμένο στο $t[j]$
- εάν $h(f) > i$ δεν μπορούμε να κάνουμε τίποτα και αυξάνουμε τον δείκτη j
- εάν $h(f) \leq i$ αφήνοντας το κενό θα δημιουργούσε πρόβλημα, και θα χανόταν το f. Για να λύσουμε το πρόβλημα αντιμεταθέτουμε τα $t[i]$ και $t[j]$, δηλαδή αντιμεταθέτουμε τα f και $\perp$. Θέτουμε την θέση του κενού i στην νέα θέση j και συνεχίζουμε με $j = j + 1$.

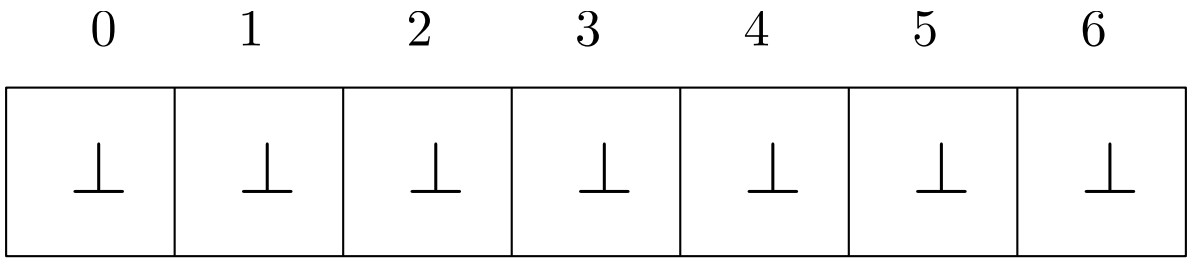
Παράδειγμα

Γραμμική Διερεύνηση

Έστω ένας πίνακας κατακερματισμού με

- μέγεθος $m = 7$
- συνάρτηση κατακερματισμού $h(k) = k \; mod \; 7$.

όπου θέλουμε να εισάγουμε τα στοιχεία 9, 10, 12, 16, 17, 23.



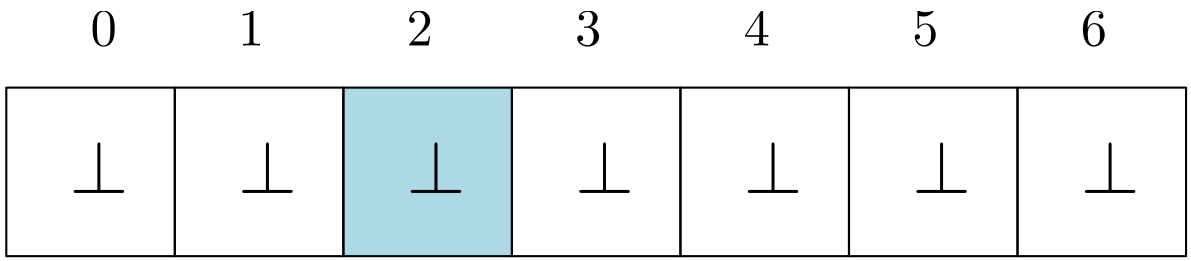
Παράδειγμα

Γραμμική Διερεύνηση

Έστω ένας πίνακας κατακερματισμού με

- μέγεθος $m = 7$
- συνάρτηση κατακερματισμού $h(k) = k \; mod \; 7$.

όπου θέλουμε να εισάγουμε τα στοιχεία 9, 10, 12, 16, 17, 23.



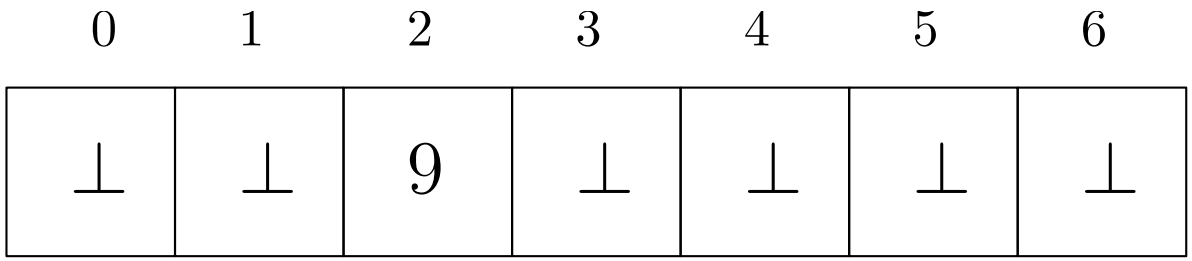
Παράδειγμα

Γραμμική Διερεύνηση

Έστω ένας πίνακας κατακερματισμού με

- μέγεθος $m = 7$
- συνάρτηση κατακερματισμού $h(k) = k \; mod \; 7$.

όπου θέλουμε να εισάγουμε τα στοιχεία 9, 10, 12, 16, 17, 23.



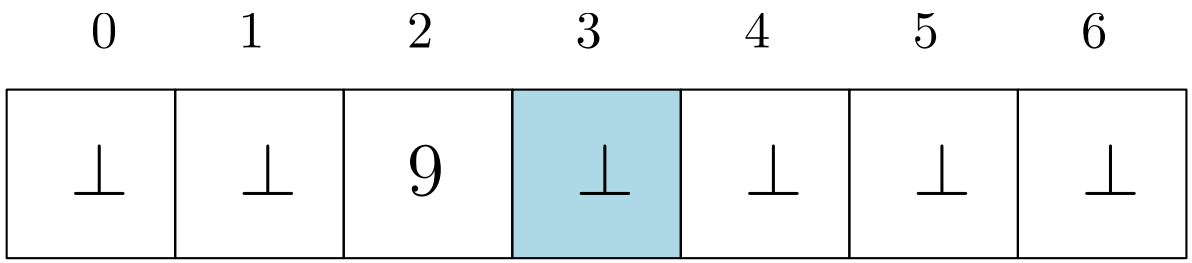
Παράδειγμα

Γραμμική Διερεύνηση

Έστω ένας πίνακας κατακερματισμού με

- μέγεθος $m = 7$
- συνάρτηση κατακερματισμού $h(k) = k \ mod \ 7$.

όπου θέλουμε να εισάγουμε τα στοιχεία 9, 10, 12, 16, 17, 23.



Παράδειγμα

Γραμμική Διερεύνηση

Έστω ένας πίνακας κατακερματισμού με

- μέγεθος $m = 7$
- συνάρτηση κατακερματισμού $h(k) = k \ mod \ 7$.

όπου θέλουμε να εισάγουμε τα στοιχεία 9, 10, 12, 16, 17, 23.

0	1	2	3	4	5	6
$\perp$	$\perp$	9	10	$\perp$	$\perp$	$\perp$

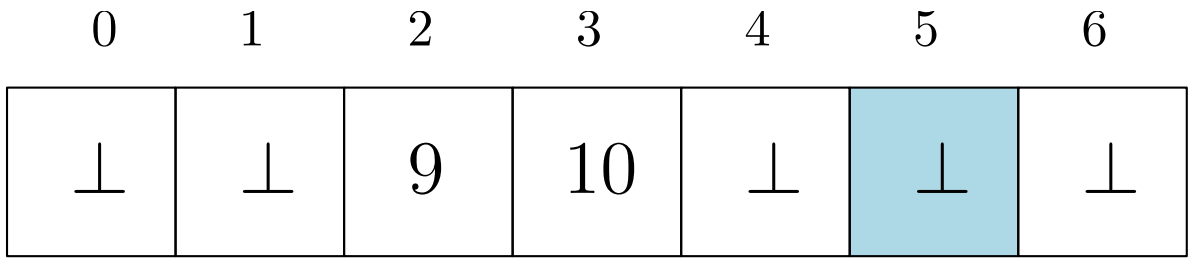
Παράδειγμα

Γραμμική Διερεύνηση

Έστω ένας πίνακας κατακερματισμού με

- μέγεθος $m = 7$
- συνάρτηση κατακερματισμού $h(k) = k \; mod \; 7$.

όπου θέλουμε να εισάγουμε τα στοιχεία 9, 10, 12, 16, 17, 23.



Παράδειγμα

Γραμμική Διερεύνηση

Έστω ένας πίνακας κατακερματισμού με

- μέγεθος $m = 7$
- συνάρτηση κατακερματισμού $h(k) = k \ mod \ 7$.

όπου θέλουμε να εισάγουμε τα στοιχεία 9, 10, 12, 16, 17, 23.

0	1	2	3	4	5	6
$\perp$	$\perp$	9	10	$\perp$	12	$\perp$

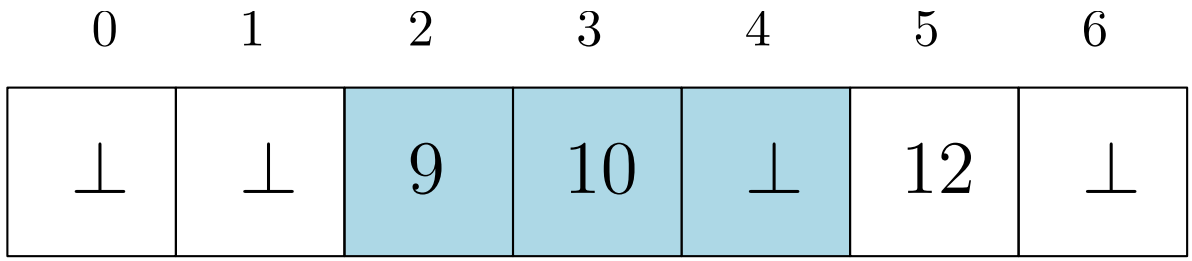
Παράδειγμα

Γραμμική Διερεύνηση

Έστω ένας πίνακας κατακερματισμού με

- μέγεθος $m = 7$
- συνάρτηση κατακερματισμού $h(k) = k \ mod \ 7$.

όπου θέλουμε να εισάγουμε τα στοιχεία 9, 10, 12, 16, 17, 23.



Παράδειγμα

Γραμμική Διερεύνηση

Έστω ένας πίνακας κατακερματισμού με

- μέγεθος $m = 7$
- συνάρτηση κατακερματισμού $h(k) = k \ mod \ 7$.

όπου θέλουμε να εισάγουμε τα στοιχεία 9, 10, 12, 16, 17, 23.

0	1	2	3	4	5	6
$\perp$	$\perp$	9	10	16	12	$\perp$

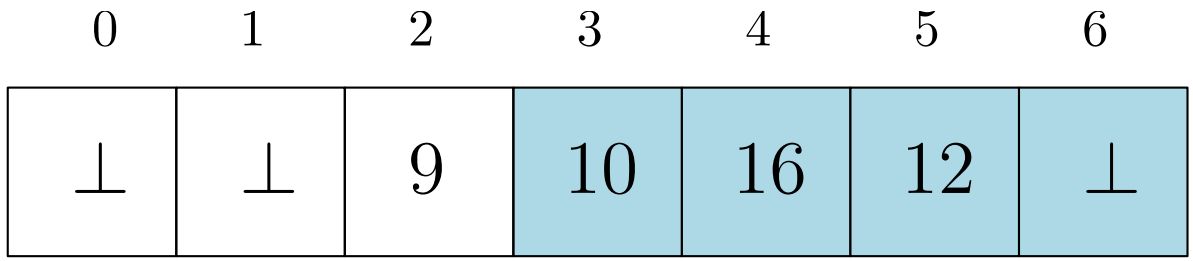
Παράδειγμα

Γραμμική Διερεύνηση

Έστω ένας πίνακας κατακερματισμού με

- μέγεθος $m = 7$
- συνάρτηση κατακερματισμού $h(k) = k \ mod \ 7$.

όπου θέλουμε να εισάγουμε τα στοιχεία 9, 10, 12, 16, 17, 23.



Παράδειγμα

Γραμμική Διερεύνηση

Έστω ένας πίνακας κατακερματισμού με

- μέγεθος $m = 7$
- συνάρτηση κατακερματισμού $h(k) = k \; mod \; 7$.

όπου θέλουμε να εισάγουμε τα στοιχεία 9, 10, 12, 16, 17, 23.

0	1	2	3	4	5	6
$\perp$	$\perp$	9	10	16	12	17

Παράδειγμα

Γραμμική Διερεύνηση

Έστω ένας πίνακας κατακερματισμού με

- μέγεθος $m = 7$
- συνάρτηση κατακερματισμού $h(k) = k \ mod \ 7$.

όπου θέλουμε να εισάγουμε τα στοιχεία 9, 10, 12, 16, 17, 23.

0	1	2	3	4	5	6
$\perp$	$\perp$	9	10	16	12	17

Παράδειγμα

Γραμμική Διερεύνηση

Έστω ένας πίνακας κατακερματισμού με

- μέγεθος $m = 7$
- συνάρτηση κατακερματισμού $h(k) = k \ mod \ 7$.

όπου θέλουμε να εισάγουμε τα στοιχεία 9, 10, 12, 16, 17, 23.

0	1	2	3	4	5	6
23	$\perp$	9	10	16	12	17

Άλλες Στρατηγικές

Στην γενική μορφή η συνάρτηση κατακερματισμού είναι:

$$h(key, i) = (h1(key) + F(i)) \ \ mod \ \ size.$$

όπου F είναι μία συνάρτηση επίλυσης συγκρούσεων.

Linear Probing $F(i) = i$

Quadratic Probing $F(i) = i^2$

Double Hashing $F(i) = i \cdot h2(key)$

Στην πράξη

Οι περισσότερες υλοποιήσεις κατακερματισμού που θα βρείτε διαθέσιμες, π.χ στην Java ή στην STL C++, χρησιμοποιούν:

- αλυσιδωτό κατακερματισμό
- επανακατακερματισμό
- δεν χρησιμοποιούν καθολικό κατακερματισμό
- παρέχουν μία συνάρτηση κατακερματισμού που λειτουργεί σχετικά καλά για τα κλειδιά που εμφανίζονται συνήθως στην πράξη