

Δομές Δεδομένων και Αλγόριθμοι

Χρήστος Γκόγκος

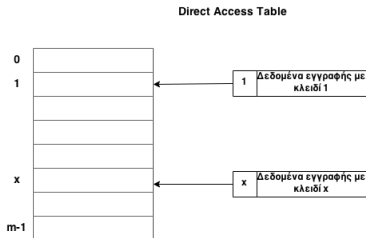
ΤΕΙ Ηπείρου

Χειμερινό Εξάμηνο 2014-2015
Παρουσίαση 19. Hashing - Κατακερματισμός

Πίνακες απευθείας πρόσβασης (Direct Access Tables)

Οι πίνακες απευθείας πρόσβασης είναι σε θέση να αποθηκεύσουν ένα σύνολο εγγραφών με κλειδιά στο διάστημα 0 έως $m - 1$ απευθείας.

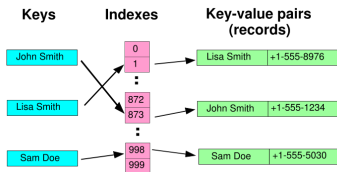
- Η λειτουργία αναζήτησης με βάση το κλειδί μιας εγγραφής γίνεται σε σταθερό χρόνο $O(1)$. Αν η εγγραφή δεν υπάρχει τότε στην αντίστοιχη θέση του πίνακα υπάρχει η τιμή NULL.
- Οι πίνακες απευθείας πρόσβασης μπορούν να χρησιμοποιηθούν μόνο σε ειδικές περιπτώσεις προβλημάτων που τα κλειδιά κυμαίνονται σε ένα σχετικά μικρό εύρος τιμών που ξεκινά από το μηδέν. Μια λύση για τις υπόλοιπες περιπτώσεις είναι το hashing (κατακερματισμός).



Τι είναι το Hashing;

Hashing

Hashing είναι ο μετασχηματισμός ενός κλειδιού σε ένα μικρότερο κλειδί που αναπαριστά το αρχικό κλειδί. Το μικρότερο κλειδί χρησιμοποιείται για να διευθυνσιοδοτήσει τα δεδομένα που συσχετίζονται με το αρχικό κλειδί και τα οποία αποτελούν μαζί με το αρχικό κλειδί μια εγγραφή.



http://en.wikibooks.org/wiki/Data_

[Structures/Hash_Tables](http://en.wikibooks.org/wiki/Data_Structures/Hash_Tables)

Λεξικά (dictionaries) - Πίνακες συσχετίσεων (associative arrays)

Λεξικό

Λεξικό (dictionary) είναι μια δομή δεδομένων η οποία υποστηρίζει με αποδοτικό τρόπο τις λειτουργίες: αναζήτηση, εισαγωγή και διαγραφή. Ο κατακερματισμός αποτελεί ένα αποδοτικό τρόπο δημιουργίας λεξικών.

- Τα στοιχεία του λεξικού μπορεί να είναι οποιοδήποτε τύπου: αριθμοί, χαρακτήρες, λεκτικά. Ωστόσο η σημαντικότερη χρήση τους είναι η περίπτωση των εγγραφών.
- Ανάμεσα στα πεδία της εγγραφής υπάρχει συνήθως τουλάχιστον ένα που αποκαλείται κλειδί και χρησιμοποιείται για να προσδιορίζει μοναδικά τις εγγραφές.

Πίνακας κατακερματισμού - Hash Table

Ο κατακερματισμός βασίζεται στην κατανομή των κλειδιών σε ένα μονοδιάστατο πίνακα m στοιχείων που ονομάζεται hashtable.

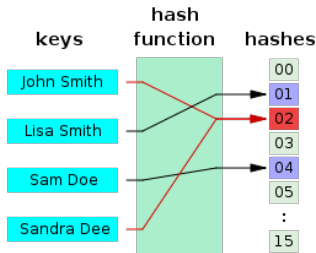
- Για κάθε κλειδί k μια συνάρτηση κατακερματισμού (hash function) h υπολογίζει ένα ακέραιο αριθμό ανάμεσα στο 0 και στο $m-1$. Για παράδειγμα αν τα κλειδιά είναι θετικοί αριθμοί τότε μια απλή συνάρτηση κατακερματισμού είναι η $h(k) = k \bmod m$.
- Η τιμή του m συνήθως επιλέγεται έτσι ώστε να είναι πρώτος αριθμός.

Χαρακτηριστικά της συνάρτησης κατακερματισμού

Μια καλή συνάρτηση κατακερματισμού θα πρέπει:

- να κατανέμει τα κλειδιά στα κελιά του hashtable όσο πιο ομοιόμορφα γίνεται.
- να είναι εύκολο να υπολογιστεί.

Είναι επιθυμητό το παραγόμενο αποτέλεσμα από τη συνάρτηση κατακερματισμού να εξαρτάται από όλα τα δυαδικά ψηφία του κλειδιού.



http://en.wikipedia.org/wiki/Hash_function

Παραδείγματα hash functions

- Μέθοδος υπολοίπου ακέραιας διαίρεσης. Επιστρέφει το υπόλοιπο της ακέραιας διαίρεσης με έναν αριθμό.
- Μέθοδος αναδίπλωσης (folding). Τα ψηφία του κλειδιού ομαδοποιούνται με κάποιο τρόπο (π.χ. ανά τριάδες), αθροίζονται και στη συνέχεια χρησιμοποιείται η μέθοδος υπολοίπου ακέραιας διαίρεσης.
- Μέθοδος μετασχηματισμού της αριθμητικής βάσης (radix transformation). Γίνεται αλλαγή της αριθμητικής βάσης του κλειδιού (για παράδειγμα από το δεκαδικό σύστημα στο επταδικό σύστημα) και στη συνέχεια χρησιμοποιείται η μέθοδος υπολοίπου ακέραιας διαίρεσης.
- Μέθοδος μέσου του τετραγώνου (mid square). Το κλειδί διπλασιάζεται και στη συνέχεια λαμβάνονται κάποια από τα μεσαία ψηφία του αριθμού.
- Μέθοδος εξάρτησης από το μήκος (length-dependent). Το κλειδί και το μήκος του κλειδιού συνδυάζονται με κάποιο τρόπο (π.χ. πολλαπλασιασμός του κλειδιού με το μήκος του) έτσι ώστε να δημιουργήσουν μια νέα τιμή, η οποία στη συνέχεια με κάποια άλλη μέθοδο όπως οι παραπάνω μετασχηματίζεται σε μια αριθμητική τιμή.

Σε περίπτωση που το κλειδί είναι κείμενο τότε μπορούν να μετασχηματιστούν τα σύμβολα από τα οποία αποτελείται στην ASCII αριθμητική τους τιμή και με αυτό τον τρόπο να είναι δυνατό να γίνουν αριθμητικές πράξεις στο κλειδί. Εναλλακτικά μπορεί να χρησιμοποιηθεί η δυαδική τους αναπαράσταση και ο τελεστής XOR.

Σύγκρουση

Σύγκρουση συμβαίνει όταν 2 ή περισσότερα κλειδιά γίνονται hash στο ίδιο κελί του hashtable.

Βασικοί μηχανισμοί επίλυσης συγκρούσεων

- Κατακερματισμός με αλυσίδες.
- Ανοικτή διευθυνσιοδότηση (γραμμική αναζήτηση, τετραγωνική αναζήτηση, διπλός κατακερματισμός).

Κατακερματισμός με αλυσίδες - Separate Chaining

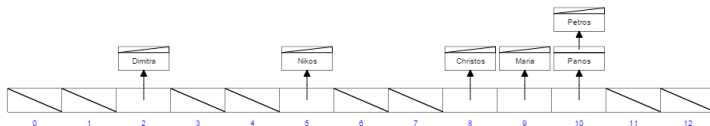
Τα κλειδιά αποθηκεύονται σε συνδεδεμένες λίστες κάθε μια από τις οποίες είναι προσαρτημένες στα κελιά ενός hashtable.

- Η απόδοση της λειτουργίας αναζήτησης εξαρτάται από τα μήκη των συνδεδεμένων λιστών. Αν η συνάρτηση κατακερματισμού κατανέμει τα n κλειδιά ανάμεσα στα m κελιά ομοιόμορφα τότε κάθε λίστα θα έχει μήκος n/m .
- Η παράσταση n/m ονομάζεται παράγοντας φόρτωσης (load factor) και δεν θα πρέπει να απέχει πολύ από την μονάδα.
- Πολύ μικρό load factor σημαίνει ότι υπάρχουν πολλές κενές λίστες και συνεπώς δεν γίνεται αποδοτική χρήση του χώρου.
- Μεγάλο load factor σημαίνει μακριές συνδεδεμένες λίστες και μεγαλύτερους χρόνους αναζήτησης.

Κατακερματισμός με αλυσίδες

Open Hashing

☐ Hash Integer
☒ Hash Strings



Animation Completed

Animation Speed

Algorithm Visualizations

<https://www.cs.usfca.edu/~galles/visualization/OpenHash.html>

Κατακερματισμός με αλυσίδες

Hash Table

Execution mode: Store

Status: Found


0		25	
1		26	776 → 626
2	302	27	
3		28	378
4	604	29	
5		30	
6	206	31	
7		32	
8	908	33	683
9		34	
10		35	
11		36	
12		37	
13		38	388
14	564	39	
15		40	
16	916	41	
17		42	
18	468 → 468 → 168	43	93
19		44	
20	320	45	
21	71	46	546
22		47	
23		48	
24	74 → 424	49	

HASH FUNCTION

Number Stored	$h(x)$	Home Address
424	mod 50	24

Numbers Remaining
5

100%



STATISTICS TABLE

TIME	00:11:68
AVERAGE PROBES	1.0
LOAD FACTOR	40.0%
SUCCESSFUL SEARCHES	20
UNSUCCESSFUL SEARCHES	0

Source Code

Hashing algorithm

linked list chaining

Display

Setup

Execute

Results

Plot

Execution

Run

Step

Pause

Abort

Options

Help

<http://groups.engin.umd.umich.edu/CIS/course.des/cis350/hashing/WEB/HashApplet.htm>

Αποτελέσματα θεωρητικής μελέτης κατακερματισμού με αλυσίδες

- Ο μέσος αριθμός στοιχείων που εξετάζονται στις επιτυχείς αναζητήσεις είναι $1 + \frac{\alpha}{2}$ όπου α είναι ο παράγοντας φόρτωσης (load factor).
- Ο μέσος αριθμός στοιχείων που εξετάζονται στις ανεπιτυχείς αναζητήσεις είναι α .

Αν το load factor α διατηρηθεί σε τιμή κοντά στη μονάδα τότε κατά μέσο όρο απαιτούνται 1 ή 2 συγκρίσεις ανά αναζήτηση συν το χρόνο που απαιτείται για τον υπολογισμό της hash function ο οποίος όμως είναι σταθερός και δεν εξαρτάται από τα μεγέθη n και m .

Όλα τα κλειδιά αποθηκεύονται στο hashtable χωρίς τη χρήση συνδεδεμένων λιστών (θα πρέπει το μέγεθος του hashtable να είναι μεγαλύτερο ή ίσο από το πλήθος των στοιχείων η που πρόκειται να αποθηκευτούν σε αυτό). Η βασική ιδέα είναι ότι αν συμβεί σύγκρουση τότε ελέγχεται αν κάποιο από τα επόμενα στη σειρά κελιά είναι διαθέσιμο και το κλειδί τοποθετείται εκεί. Αν φτάσει στο τέλος του πίνακα ο έλεγχος συνεχίζεται από την αρχή και μέχρι να εξαντληθούν όλα τα στοιχεία (γραμμική αναζήτηση).

- Η απόδοση της ανοικτής διευθυνσιοδότησης μειώνεται κατακόρυφα σε περίπτωση που το hash table είναι σχεδόν γεμάτο.
- Εκτός από τη γραμμική αναζήτηση (linear probing) υπάρχει και η τετραγωνική αναζήτηση καθώς και ο διπλός κατακερματισμός.

Closed Hashing

- ☒ Hash Integer
- ☒ Linear Probing: $f(i) = i$
- ☒ Hash Strings
- ☐ Quadratic Probing: $f(i) = i * i$
- ☐ Double Hashing: $f(i) = i * \text{hash2}(\text{elem})$

			Nikos	Dimitra			Panos			
0	1	2	3	4	5	6	7	8	9	10

			Maria				Christina			
11	12	13	14	15	16	17	18	19	20	21

		Katerina		Petros	Christos	Giorgos
22	23	24	25	26	27	28

Animation Completed

[Skip Back](#)
[Step Back](#)
[Pause](#)
[Step Forward](#)
[Skip Forward](#)
v: 1000 h: 500
[Change Canvas Size](#)
[Move Controls](#)

<https://www.cs.usfca.edu/~galles/visualization/ClosedHash.html>

Ανοικτή διευθυνσιοδότηση

Hash Table

Execution mode: Store

Status: Searching...


0	25	50	650	75
1	26	51		76
2	27	52		77
3	28	53		78
4	29	54		79
5	30	55		80
6	31	531		81
7	32	57		82
8	33	58		83
9	34	134		84
10	35	60		85
11	36	61		86
12	912	37		87
13		63		88
14	614	39		89
15		40		90
16		41		91
17		42	442	92
18		43	742	93
19	19	44		94
20		45		95
21		46		96
22		47		97
23		48		98
24		49		99

HASH FUNCTION

Number Stored	$h(x)$	Home Address
750	mod 100	50

Numbers Remaining
39

100%



STATISTICS TABLE

TIME	06:59:19
AVERAGE PROBES	1.08
LOAD FACTOR	11.0%
SUCCESSFUL SEARCHES	11
UNSUCCESSFUL SEARCHES	0

Source Code

Hashing algorithm

Display

Execution

linear probing

Setup

Execute

Results

Plot

Run

Step

Pause

Abort

Options

Help

<http://groups.engin.umd.umich.edu/CIS/course.des/cis350/hashing/WEB/HashApplet.htm>

Αποτελέσματα θεωρητικής μελέτης ανοικτής διευθυνσιοδότησης (με γραμμική αναζήτηση)

- Ο μέσος αριθμός στοιχείων που εξετάζονται στις επιτυχείς αναζητήσεις είναι $\frac{1}{2} \left(1 + \frac{1}{1-\alpha} \right)$ όπου α είναι ο παράγοντας φόρτωσης (load factor).
- Ο μέσος αριθμός στοιχείων που εξετάζονται στις ανεπιτυχείς αναζητήσεις είναι $\frac{1}{2} \left(1 + \frac{1}{(1-\alpha)^2} \right)$.

Στο linear probing καθώς το hashtable πλησιάζει στο να γεμίσει η απόδοσή του μειώνεται λόγω του φαινομένου clustering (συνεχόμενα κατειλημμένα κελιά). Καθώς τα clusters γίνονται μεγαλύτερα η πιθανότητα να προσαρτηθεί σε αυτά ένα νέο στοιχείο αυξάνεται.

Η εισαγωγή και η αναζήτηση υλοποιούνται εύκολα στον κατακερματισμό με ανοικτή διευθυνσιοδότηση αλλά η διαγραφή όχι. Μια λύση είναι να χρησιμοποιηθεί lazy deletion δηλαδή τα στοιχεία που διαγράφονται να μαρκάρονται ως διαγραμμένα και όχι να διαγράφονται στην πραγματικότητα.

Τετραγωνικός κατακερματισμός - quadratic probing

Στον τετραγωνικό κατακερματισμό αν συμβεί σύγκρουση τότε χρησιμοποιείται ένας τύπος της μορφής $i * i$ (όπου i είναι ο αύξων αριθμός της αναζήτησης) έτσι ώστε να προσδιοριστεί η επόμενη θέση στο hashtable που θα εξεταστεί για το εάν είναι διαθέσιμη.

Άρα ενώ αν συμβεί σύγκρουση στο linear probing οι επόμενες θέσεις που εξετάζονται για διαθεσιμότητα απέχουν 1, 2, 3, 4, 5 ... από τη θέση στην οποία έγινε η σύγκρουση, στο quadratic probing εξετάζονται οι θέσεις 1, 4, 9, 16, 25 ... από τη θέση στην οποία έγινε η σύγκρουση.

Εναλλακτικά μπορούν να χρησιμοποιηθούν και πιο σύνθετοι τύποι σε σχέση με το $i * i$ όπως ο $(-1)^{i-1}(\frac{i+1}{2})^2$ ο οποίος παράγει την ακολουθία τιμών 1, -1, 4, -4, 9, -9, ...

Διπλός κατακερματισμός - double hashing

Στο διπλό κατακερματισμό αν συμβεί σύγκρουση τότε χρησιμοποιείται μια δεύτερη hash function για να καθορίσει το βήμα με το οποίο θα εξεταστούν τα επόμενα κελιά έτσι ώστε να βρεθεί κενή θέση.

Στην πράξη, με καλή επιλογή των δύο hash functions, το double hashing δίνει καλύτερα αποτελέσματα από το linear probing. Για παράδειγμα μπορεί να χρησιμοποιηθούν οι ακόλουθες συναρτήσεις

- $h_2(\text{key}) = m - 2 - \text{key} \bmod (m - 2)$ για μικρούς πίνακες
- $h_2(\text{key}) = \text{key} \bmod 97 + 1$ για μεγάλους πίνακες

Το hashtable έχει αρχικά μέγεθος m και όταν γεμίσει κατά το ήμισυ το μέγεθός του διπλασιάζεται. Για κάθε στοιχείο του αρχικού hashtable επαναυπολογίζεται η νέα του θέση στο νέο hashtable.

Επιλογή ανάμεσα στις εναλλακτικές υλοποιήσεις κατακερματισμού

Ο κατακερματισμός με ανοικτή διευθυνσιοδότηση κάνει αποδοτικότερη χρήση μνήμης καθώς δεν απαιτείται επιπλέον χώρος για τους pointers.

Γενικά μπορεί να χρειαστεί η υλοποίηση εναλλακτικών τρόπων (κατακερματισμού με αλυσίδες και ανοικτή διευθυνσιοδότηση) έτσι ώστε να επιλεγεί κατά περίπτωση η αποδοτικότερη υλοποίηση.

Εφαρμογές κατακερματισμού

- Αφαίρεση διπλοτύπων
- 2-sum problem. Δίνεται μια μη ταξινομημένη σειρά από n ακεραίους και μια τιμή x . Ζητείται να βρεθεί το εάν υπάρχει ζεύγος τιμών που να δίνει ως άθροισμα το x .
- Μεταγλωττιστές. Τοποθέτηση σε hashtable συμβόλων για τα οποία χρειάζεται συχνά αναζήτηση.
- Routers. Μπλοκάρισμα κυκλοφορίας που προέρχεται από συγκεκριμένες διευθύνσεις (blacklists).
- Εξυπηρετούν ανάγκες αναζήτησης σε περιπτώσεις που υπάρχουν πολλές εναλλακτικές λύσεις και θέλουμε να γνωρίζουμε αν έχουμε ξαναπεράσει από μια κατάσταση την οποία την βλέπουμε πάλι μπροστά μας.

- Η C++ υποστηρίζει απευθείας hashtable μέσω του `std::unordered_map` το οποίο έχει υλοποιηθεί χρησιμοποιώντας κατακερματισμό με αλυσίδες.
- Η δομή `std::unordered_map` είναι ταχύτερη από τη δομή `std::map` (που είναι υλοποιημένη με δυαδικό δένδρο αναζήτησης) όταν ζητούνται αναζητήσεις στοιχείων με βάση το κλειδί. Ωστόσο είναι λιγότερο αποδοτική όταν πρόκειται να διανυθεί ένα υποσύνολο των στοιχείων της.

```
...
unordered_map<string,
             string> agenta;
agenta["Christos"] =
    "12345";
agenta["Maria"] = "23456";
agenta["Petros"] = "34567";
```

```
for (pair<string, string>
    k_v : agenta) {
    cout << k_v.first <<
        "___>" <<
        k_v.second << endl;
}
//outputs
// Petros___>34567
// Maria___>23456
// Christos___>12345
```