



ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΡΗΤΗΣ

Δομές Δεδομένων

Ιωάννης Γ. Τόλλης
Τμήμα Επιστήμης Υπολογιστών
Πανεπιστήμιο Κρήτης

Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «**Ανοικτά Ακαδημαϊκά Μαθήματα στο Πανεπιστήμιο Κρήτης**» έχει χρηματοδοτήσει μόνο τη αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.



Σημείωμα αδειοδότησης

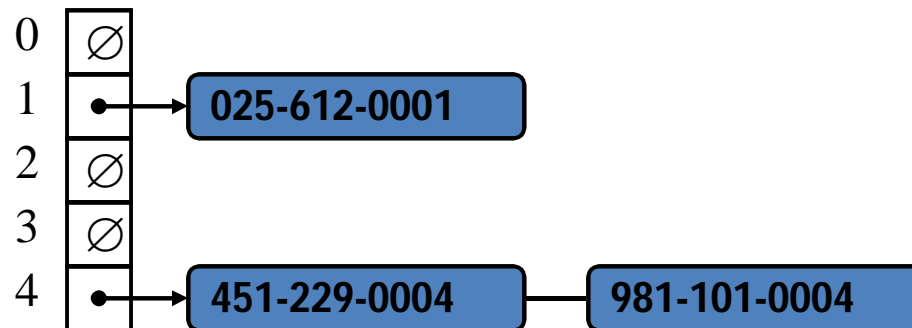
- Το παρόν υλικό διατίθεται με τους όρους της άδειας χρήσης Creative Commons Αναφορά, Μη Εμπορική Χρήση, Όχι Παράγωγο Έργο 4.0 [1] ή μεταγενέστερη, Διεθνής Έκδοση. Εξαιρούνται τα αυτοτελή έργα τρίτων π.χ. φωτογραφίες, διαγράμματα κ.λ.π., τα οποία εμπεριέχονται σε αυτό και τα οποία αναφέρονται μαζί με τους όρους χρήσης τους στο «Σημείωμα Χρήσης Έργων Τρίτων».

[1] <http://creativecommons.org/licenses/by-nc-nd/4.0/>



- Ως **Μη Εμπορική** ορίζεται η χρήση:
 - που δεν περιλαμβάνει άμεσο ή έμμεσο οικονομικό όφελος από την χρήση του έργου, για το διανομέα του έργου και αδειοδόχο
 - που δεν περιλαμβάνει οικονομική συναλλαγή ως προϋπόθεση για τη χρήση ή πρόσβαση στο έργο
 - που δεν προσπορίζει στο διανομέα του έργου και αδειοδόχο έμμεσο οικονομικό όφελος (π.χ. διαφημίσεις) από την προβολή του έργου σε διαδικτυακό τόπο
- Ο δικαιούχος μπορεί να παρέχει στον αδειοδόχο ξεχωριστή άδεια να χρησιμοποιεί το έργο για εμπορική χρήση, εφόσον αυτό του ζητηθεί.

Πίνακες Κατακερματισμού



Περιγραφή και Υλικό Ανάγνωσης

- Συναρτήσεις κατακερματισμού και πίνακες κατακερματισμού (§2.5.2)
- Λεπτομέρειες συνάρτησης κατακερματισμού
 - Hash code map (§2.5.3)
 - Compression map (§2.5.4)
- Χειρισμός συγκρούσεων (§2.5.5)
 - Chaining
 - Linear probing
 - Double hashing

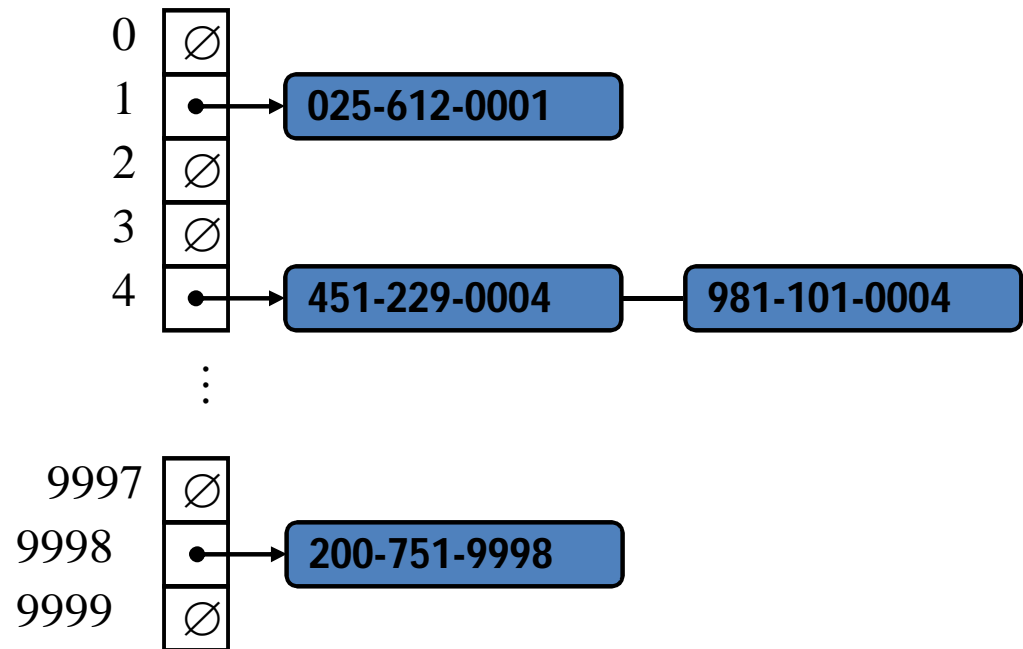
Πίνακες Κατακερματισμού

Συναρτήσεις Κατακερματισμού και Πίνακες Κατακερματισμού

- Μια συνάρτηση κατακερματισμού (hash function) h απεικονίζει κλειδιά ενός δοσμένου τύπου σε ακεραίους ενός σταθερού διαστήματος $[0, N - 1]$
- Παράδειγμα:
$$h(x) = x \bmod N$$
είναι μια συνάρτηση κατακερματισμού για ακέραια κλειδιά
- Ο ακέραιος $h(x)$ ονομάζεται τιμή κατακερματισμού (hash value) του κλειδιού x
- Ο σκοπός μιας συνάρτησης κατακερματισμού είναι η ομοιόμορφη διασπορά κλειδιών στο πεδίο $[0, N - 1]$
- Ένας πίνακας κατακερματισμού (hash table) για ένα δοσμένο τύπο κλειδιών αποτελείται από
 - Μία συνάρτηση κατακερματισμού h
 - Ένα πίνακα μεγέθους N
- Όταν υλοποιούμε ένα λεξικό με ένα πίνακα κατακερματισμού, ο στόχος είναι να αποθηκεύσουμε το αντικείμενο (k, o) στη θέση $i = h(k)$
- Μια σύγκρουση (collision) συμβαίνει όταν δύο κλειδιά του λεξικού έχουν την ίδια τιμή κατακερματισμού
- Σχήματα χειρισμού συγκρούσεων:
 - Chaining: τα συγκρουόμενα αντικείμενα αποθηκεύονται σε ακολουθία
 - Open addressing: τα συγκρουόμενα αντικείμενα τοποθετούνται σε διαφορετικά κελιά του πίνακα

Παράδειγμα

- Σχεδιάζουμε ένα πίνακα κατακερματισμού για ένα λεξικό που αποθη-κεύει αντικείμενα (SSN, Name), όπου SSN (social security number) είναι ένας εννιαψήφιος θετικός ακέραιος
- Ο πίνακας κατακερματισμού χρησιμοποιεί ένα πίνακα μεγέθους $N = 10,000$ και η συνάρτηση κατακερματισμού είναι η $h(x) = \text{last four digits of } x$
- Χρησιμοποιούμε chaining για να χειριστούμε τις συγκρούσεις



Συναρτήσεις Κατακερματισμού

- Μια συνάρτηση κατακερματισμού συνήθως καθορίζεται ως μια σύνθεση δύο συναρτήσεων :
Hash code map:
 $h_1: \text{keys} \rightarrow \text{integers}$
Compression map:
 $h_2: \text{integers} \rightarrow [0, N - 1]$
- Η hash code map εφαρμόζεται πρώτα, και η compression map εφαρμόζεται αργότερα στο αποτέλεσμα, π.χ.,
$$h(x) = h_2(h_1(x))$$
- Ο στόχος της συνάρτησης κατακερματισμού είναι να “διασκορπίσει” τα κλειδιά με ένα φαινομενικά τυχαίο τρόπο

Hash Code Maps

- **Memory address:**
 - Επαναερμηνεύουμε τη διεύθυνση μνήμης του key object σαν ακέραιο (προκαθορισμένος hash code για όλα τα αντικείμενα Java)
 - Γενικά καλό, εκτός από αριθμητικά και αλφαριθμη-τικά κλειδιά
- **Integer cast:**
 - Επαναερμηνεύουμε τα bits του κλειδιού σαν ακέραιο
 - Κατάλληλο για κλειδιά μήκους μικρότερου ή ίσου του αριθμού των bits του τύπου ακεραίου (π.χ., byte, short, int και float στη Java)
- **Component sum:**
 - Χωρίζουμε τα bits του κλειδιού σε συστατικά σταθερού μήκους (π.χ., 16 ή 32 bits) και προσθέ-
τουμε τα συστατικά (αγνοώντας τα overflows)
 - Κατάλληλο για αριθμητικά κλειδιά σταθερού μήκους μεγαλύτερου ή ίσου του αριθμού των bits του τύπου ακεραίου (π.χ., long και double στη Java)

Hash Code Maps (cont.)

- **Polynomial accumulation:**

- Χωρίζουμε τα bits του κλειδιού σε μια ακολουθία συστατικών σταθερού μεγέθους (π.χ., 8, 16 ή 32 bits)

$$a_0 a_1 \dots a_{n-1}$$

- Αποτιμούμε το πολυώνυμο

$$p(z) = a_0 + a_1 z + a_2 z^2 + \dots \\ \dots + a_{n-1} z^{n-1}$$

σε μια σταθερή τιμή z , αγνοώντας τα overflows

- Ιδιαίτερα κατάλληλο για αλφαριθμητικά (π.χ., η επιλογή $z = 33$ δίνει το πολύ 6 συγκρούσεις σε ένα σύνολο 50,000 Αγγλικών λέξεων)

- Το πολυώνυμο $p(z)$ μπορεί να αποτιμηθεί σε χρόνο $O(n)$ χρησιμοποιώντας τον κανόνα του Horner:

- Τα ακόλουθα πολυώνυμα υπολογίζονται διαδοχικά, το κάθε ένα από το προηγούμενο σε χρόνο $O(1)$

$$p_0(z) = a_{n-1}$$

$$p_i(z) = a_{n-i-1} + zp_{i-1}(z) \\ (i = 1, 2, \dots, n-1)$$

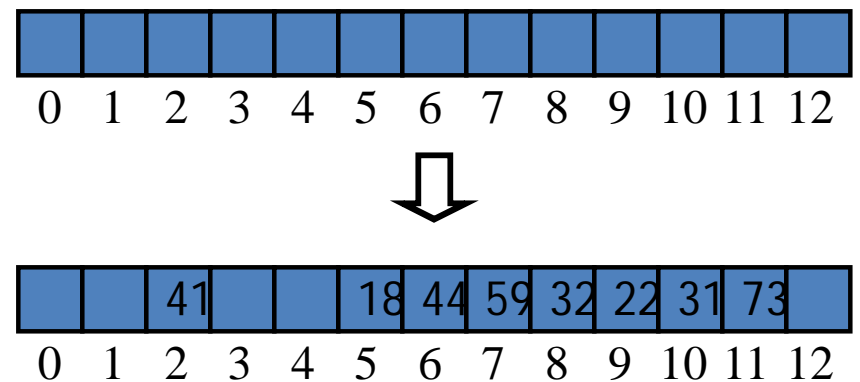
- Ισχύει ότι $p(z) = p_{n-1}(z)$

Compression Maps

- Division:
 - $h_2(\mathbf{y}) = \mathbf{y} \bmod N$
 - Το μέγεθος N του πίνακα κατακερματισμού συνήθως επιλέγεται να είναι πρώτος αριθμός
 - Ο λόγος έχει να κάνει με τη θεωρία αριθμών και είναι πέρα από τους σκοπούς του μαθήματος
- Multiply, Add and Divide (MAD):
 - $h_2(\mathbf{y}) = (a\mathbf{y} + \mathbf{b}) \bmod N$
 - Οι a και b είναι θετικοί ακέραιοι έτσι ώστε $a \bmod N \neq 0$
 - Αλλιώς, κάθε ακέραιος θα απεικονιζόταν στην ίδια τιμή b

Linear Probing

- Το Linear probing χειρίζεται τις συγκρούσεις τοποθετώντας το συγκρουόμενο αντικείμενο στο επόμενο (κυκλικά) διαθέσιμο κελί του πίνακα
 - Κάθε κελί του πίνακα που επιθεωρείται αναφέρεται ως "probe"
 - Συγκρουόμενα αντικείμενα συσσωρεύονται, με συνέπεια μελλοντικές συγκρούσεις να προκαλούν μεγαλύτερες ακολουθίες από probes
- Παράδειγμα:
 - $h(x) = x \bmod 13$
 - Εισαγωγή των κλειδιών 18, 41, 22, 44, 59, 32, 31, 73, σε αυτή την σειρά



Search with Linear Probing

- Ας θεωρήσουμε ένα πίνακα κατακερματισμού A ο οποίος χρησιμοποιεί linear probing
- $\text{findElement}(k)$
 - Ξεκινάμε στο κελί $h(k)$
 - Ερευνούμε συνεχόμενες θέσεις μέχρι ένα από τα ακόλουθα να συμβεί
 - Ένα αντικείμενο με κλειδί k να βρεθεί, ή
 - Ένα άδειο κελί να βρεθεί, ή
 - N κελιά να ερευνηθούν ανεπιτυχώς

```
Algorithm findElement(k)
   $i \leftarrow h(k)$ 
   $p \leftarrow 0$ 
  repeat
     $c \leftarrow A[i]$ 
    if  $c = \emptyset$ 
      return NO_SUCH_KEY
    else if  $c.key() = k$ 
      return  $c.element()$ 
    else
       $i \leftarrow (i + 1) \bmod N$ 
       $p \leftarrow p + 1$ 
  until  $p = N$ 
  return NO_SUCH_KEY
```

Updates with Linear Probing

- Για να χειριστούμε εισαγωγές και διαγραφές, εισάγουμε ένα ειδικό αντικείμενο, ονόματι *AVAILABLE*, το οποίο αντικαθιστά διαγραφέντα στοιχεία
- *removeElement(k)*
 - Ψάχνουμε για ένα αντικείμενο με κλειδί *k*
 - Αν ένα τέτοιο αντικείμενο (*k, o*) βρεθεί, το αντικαθιστούμε με το ειδικό αντικείμενο *AVAILABLE* και επιστρέφουμε το στοιχείο *o*
 - Αλλιώς, επιστρέφουμε *NO_SUCH_KEY*
- *insertItem(k, o)*
 - Πετάμε exception αν ο πίνακας είναι πλήρης
 - Ξεκινάμε στο κελί *h(k)*
 - Ερευνούμε συνεχόμενα κελιά μέχρι ένα από τα ακόλουθα να συμβεί
 - Ένα κελί *i* να βρεθεί το οποίο είναι άδειο ή έχει αποθηκευμένη την τιμή *AVAILABLE*, ή
 - *N* κελιά να ερευνηθούν ανεπιτυχώς
 - Αποθηκεύουμε το αντικείμενο (*k, o*) στο κελί *i*

Double Hashing

- Το Double hashing χρησιμοποιεί μια δευτερεύουσα συνάρτηση κατακερματισμού $d(k)$ και χειρίζεται τις συγκρούσεις τοποθετώντας το αντικείμενο στο πρώτο διαθέσιμο κελί από τις σειρές
 $(i + jd(k)) \bmod N$
για $j = 0, 1, \dots, N - 1$
- Η δευτερεύουσα συνάρτηση κατακερματισμού $d(k)$ δεν μπορεί να έχει μηδενικές τιμές
- Το μέγεθος N του πίνακα πρέπει να είναι πρώτος αριθμός έτσι ώστε να επιτρέπει την εξερεύνηση όλων των κελιών
- Συνήθης επιλογή του compression map για τη δευτερεύουσα συνάρτηση κατακερματισμού :
$$d_2(k) = q - k \bmod q$$

όπου
 - $q < N$
 - q είναι πρώτος αριθμός
- Οι πιθανές τιμές για το $d_2(k)$ είναι
 $1, 2, \dots, q$

Παράδειγμα Double Hashing

- Ας θεωρήσουμε ένα πίνακα κατακερματισμού ο οποίος αποθηκεύει ακέραια κλειδιά και χειρίζεται συγκρούσεις με double hashing

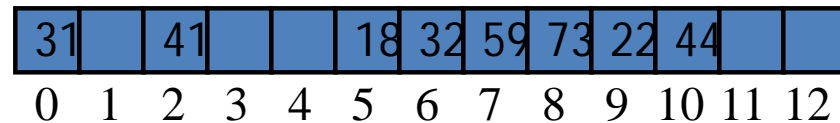
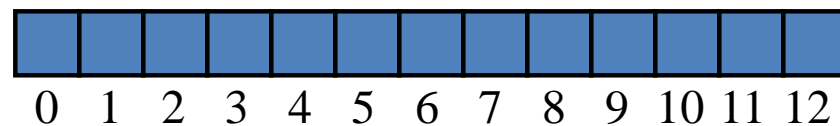
– $N = 13$

– $h(k) = k \bmod 13$

– $d(k) = 7 - k \bmod 7$

- Εισαγωγή των κλειδιών 18, 41, 22, 44, 59, 32, 31, 73, σε αυτή την σειρά

k	$h(k)$	$d(k)$	Probes	
18	5	3	5	
41	2	1	2	
22	9	6	9	
44	5	5	5	10
59	7	4	7	
32	6	3	6	
31	5	4	5	9 0
73	8	4	8	



Απόδοση του Hashing

- Στη χειρότερη περίπτωση, αναζητήσεις, εισαγωγές και διαγραφές σε ένα πίνακα κατακερματισμού εκτελείται σε χρόνο $O(n)$
 - Η χειρότερη περίπτωση συμβαίνει όταν όλα τα κλειδιά που εισάγονται στο λεξικό συγκρούονται
 - Ο παράγοντας φόρτου (load factor) $\alpha = n/N$ α επηρεάζει την απόδοση του πίνακα κατακερματισμού
 - Υποθέτοντας ότι οι τιμές κατακερματισμού συμπεριφέρονται σαν τυχαίοι αριθμοί, μπορεί ναδειχτεί ότι ο αναμενόμενος αριθμός από probes για μια εισαγωγή με open addressing είναι
- Ο αναμενόμενος χρόνος εκτέλεσης για όλες τις πράξεις του ΑΤΔ του λεξικού σε ένα πίνακα κατακερματισμού είναι $O(1)$
 - Στην πράξη, το hashing είναι πολύ γρήγορο αν ο παράγοντας φόρτου δεν είναι κοντά στο 100%
 - Εφαρμογές των πινάκων κατακερματισμού:
 - Μικρές βάσεις δεδομένων
 - Μεταφραστές (compilers)
 - browser caches

$$1 / (1 - \alpha)$$

Πίνακες Κατακερματισμού

Τέλος Ενότητας



Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο



ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ & ΘΡΗΣΚΕΥΜΑΤΩΝ, ΠΟΛΙΤΙΣΜΟΥ & ΑΘΛΗΤΙΣΜΟΥ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ

Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης



ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ