

6<sup>η</sup> Εργαστηριακή Άσκηση  
Κατακερματισμός

Παράδοση έως Τετάρτη 13/12, 14:00 από το eCourse

**ΠΡΟΣΟΧΗ:** Γράψτε σε κάθε αρχείο που παραδίδετε τα ονόματα, τους Α.Μ. των μελών της ομάδας σας, καθώς και το group του εργαστηρίου σας. Συμπεριλάβετε όλα τα αρχεία σας (κώδικας Java και lab6results.txt) σε ένα zip αρχείο. Το όνομα που θα δώσετε στο συμπιεσμένο αρχείο θα αποτελείται από το group του εργαστηρίου στο οποίο έχετε τοποθετηθεί καθώς και από το ID της ομάδας σας (π.χ., G1\_ID1.zip).

Συμπληρώστε τα προγράμματα ChainingHT.java και LinearProbingHT.java έτσι ώστε να υλοποιούν, αντίστοιχα, κατακερματισμό με χωριστές αλυσίδες και κατακερματισμό μεταβλητών διευθύνσεων (ανοικτής διευθυνσιοδότησης) με γραμμική βολιδοσκόπηση. Και στα δύο προγράμματα, διατηρούμε ένα πίνακα κατακερματισμού  $T[0:m-1]$  ο οποίος αποθηκεύει ζεύγη  $\langle \text{key}, \text{value} \rangle$ , όπου **key** ένα κλειδί γενικού τύπου Key και **value** μια τιμή γενικού τύπου Value που έχουμε αντιστοιχίσει στο κλειδί. Το μέγεθος  $m$  του πίνακα κατακερματισμού καθορίζεται από τον συντελεστή πληρότητας  $\text{loadFactor} = 100 \times n/m$ , όπου  $n$  το πλήθος των αντικειμένων που έχουν εισαχθεί.

Ο κατακερματισμός υποστηρίζει τις ακόλουθες μεθόδους που καλείστε να υλοποιήσετε:

Value contains(Key key) Επιστρέφει την τιμή *value* που αντιστοιχεί στο κλειδί *key*. Αν το *key* δεν έχει εισαχθεί στη δομή τότε επιστρέφει *null*.

void insert(Key key, Value value) Εισάγει στη δομή κατακερματισμού το κλειδί *key* με αντίστοιχη τιμή *value*. Αν το *key* υπάρχει ήδη στη δομή, τότε αντικαθιστά την προηγούμενη τιμή του με την τιμή *value*.

Η θέση στην οποία θα τοποθετηθεί ένα ζεύγος  $\langle \text{key}, \text{value} \rangle$  καθορίζεται από την τιμή της συνάρτησης κατακερματισμού.

### Κατακερματισμός με Χωριστές Αλυσίδες

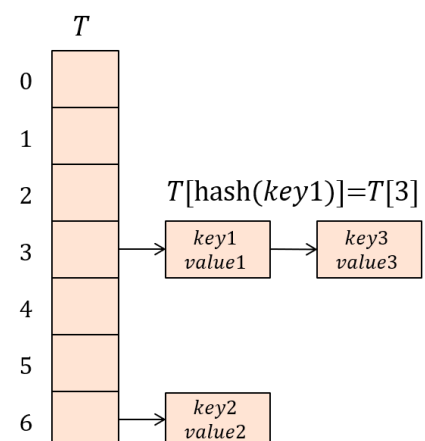
Χρησιμοποιούμε τη συνάρτηση κατακερματισμού

$$\text{hash}(\text{key}) = (\text{key.hashCode()} \& 0x7fffffff) \bmod m.$$

Στο ChainingHT.java τα ζεύγη  $\langle \text{key}, \text{value} \rangle$  αποθηκεύονται σε συνδεδεμένες λίστες, όπου κάθε λίστα αντιστοιχεί σε μια θέση του πίνακα κατακερματισμού  $T$ . Η λίστα στην οποία θα τοποθετηθεί ένα ζεύγος  $\langle \text{key}, \text{value} \rangle$  δίνεται από την αναφορά  $T[\text{hash}(\text{key})]$ , όπως φαίνεται στο διπλανό σχήμα.

Οι συνδεδεμένες λίστες κατασκευάζονται από κόμβους τύπου Node, με πεδία *key*, *value* και *next* (αναφορά στον επόμενο κόμβο της λίστας). Επομένως, ο πίνακας  $T$  είναι τύπου Node [].

- **Εισαγωγή κλειδιού *key* με τιμή *value*:** Υπολογίζουμε τη θέση  $k = \text{hash}(\text{key})$  του πίνακα κατακερματισμού και αναζητούμε το κλειδί *key* στη συνδεδεμένη λίστα με αναφορά  $T[k]$ . Για να



Πανεπιστήμιο Ιωαννίνων – Τμήμα Μηχανικών Η/Υ και Πληροφορικής  
Δομές Δεδομένων [ΜΥΥ303] – Χειμερινό Εξάμηνο 2023

ελέγχουμε αν το  $key$  είναι αποθηκευμένο σε ένα κόμβο  $x$ , ελέγχουμε αν  $x.key.equals(key) == true$ .

Αν το  $key$  βρεθεί στον κόμβο  $x$ , τότε αρκεί να θέσουμε  $x.value = value$ . Διαφορετικά, τοποθετούμε το κλειδί  $key$  και την τιμή  $value$  στη συνδεδεμένη λίστα με αναφορά  $T[k]$ . Η τοποθέτηση γίνεται με τη δημιουργία ενός νέου κόμβου  $Node\ x$ , όπου θέτουμε  $x.key = key$  και  $x.value = value$ . Αν ο συντελεστής πληρότητας του  $T$  είναι μεγαλύτερος από ένα κατώφλι  $f$  τότε δημιουργούμε ένα νέο πίνακα  $T'$  με διπλάσιο μέγεθος και τοποθετούμε εκεί όλα τα αντικείμενα του  $T$ . Δηλαδή εισάγουμε τα αντικείμενα του  $T$  στην κατάλληλη θέση του  $T'$ , χρησιμοποιώντας τη συνάρτηση κατακερματισμού  $hash(key)$  αλλά με διπλάσιο  $m$ .

- **Αναζήτηση κλειδιού  $key$ :** Υπολογίζουμε τη θέση  $k = hash(key)$  του πίνακα κατακερματισμού και αναζητούμε το  $key$  στη συνδεδεμένη λίστα με αναφορά  $T[k]$ . Για να ελέγχουμε αν το  $key$  είναι αποθηκευμένο σε ένα κόμβο  $x$ , ελέγχουμε αν  $x.key.equals(key) == true$ .

### Κατακερματισμός με Γραμμική Βολιδοσκοπήση

Σε αυτή τη μέθοδο, τα αντικείμενα αποθηκεύονται απευθείας στον πίνακα  $T[0:m-1]$  τύπου  $Pair$ , με πεδία  $key$  και  $value$ . Επομένως, ο πίνακας  $T$  είναι τύπου  $Pair[]$ .

Η θέση ενός αντικειμένου  $item$  στον πίνακα κατακερματισμού υπολογίζεται με τη συνάρτηση κατακερματισμού

$$hashL(k, i) = (k + i) \bmod m$$

όπου  $k = hash(key)$ .

Έστω ζεύγος  $\langle key, value \rangle$  με  $hash(key) = k$ .

$T$	
0	
1	
2	
3	$key1$ $value1$
4	$key3$ $value3$
5	
6	$key2$ $value2$

$T[hash(key1)] = T[3]$

- **Εισαγωγή κλειδιού  $key$  με τιμή  $value$ :** Βολιδοσκοπούμε τις θέσεις  $hashL(k, 0)$ ,  $hashL(k, 1)$ ,  $hashL(k, 2)$ , ..., του πίνακα κατακερματισμού  $T$  μέχρι να βρούμε την πρώτη θέση  $j = hashL(k, i)$  τέτοια ώστε να ισχύει η συνθήκη  $(T[j] == null) \vee (T[j].key.equals(key))$ .

Στην πρώτη περίπτωση ( $T[j] == null$ ) η θέση  $j$  είναι κενή και επομένως το κλειδί  $key$  δεν είχε εισαχθεί προηγουμένως στον πίνακα κατακερματισμού. Τότε, τοποθετούμε το ζεύγος  $\langle key, value \rangle$  στη θέση  $j$  του πίνακα  $T$ , θέτοντας  $T[j] = new\ Pair(key, value)$ . Αν ο συντελεστής πληρότητας του  $T$  είναι μεγαλύτερος από ένα κατώφλι  $f$  τότε δημιουργούμε ένα νέο πίνακα  $T'$  με διπλάσιο μέγεθος και τοποθετούμε εκεί όλα τα αντικείμενα του  $T$ . Δηλαδή εισάγουμε τα αντικείμενα του  $T$  στην κατάλληλη θέση του  $T'$ , χρησιμοποιώντας τις συναρτήσεις κατακερματισμού  $hash(key)$  και  $hashL(k, i)$  αλλά με διπλάσιο  $m$ .

Στη δεύτερη περίπτωση ( $T[j].key.equals(key)$ ) το κλειδί  $key$  έχει ήδη εισαχθεί στον πίνακα κατακερματισμού, επομένως αρκεί να αλλάξουμε την τιμή του κλειδιού του θέτοντας  $T[j].value = value$ .

- **Αναζήτηση κλειδιού  $key$ :** Βολιδοσκοπούμε τις θέσεις  $hashL(k, 0)$ ,  $hashL(k, 1)$ ,  $hashL(k, 2)$ , ..., του πίνακα κατακερματισμού  $T$  μέχρι να βρούμε την πρώτη θέση  $j =$

## Πανεπιστήμιο Ιωαννίνων – Τμήμα Μηχανικών Η/Υ και Πληροφορικής Δομές Δεδομένων [ΜΥΥ303] – Χειμερινό Εξάμηνο 2023

$\text{hashL}(k, i)$  όπου είτε α)  $T[j].\text{key.equals}(key) == \text{true}$ , οπότε η αναζήτηση είναι επιτυχής, είτε β)  $T[j] == \text{null}$  οπότε η αναζήτηση είναι ανεπιτυχής.

### Εκτέλεση Προγράμματος

Η `main()` μέθοδος των προγραμμάτων `ChainingHT.java` και `LinearProbingHT.java` διαβάζει από ένα αρχείο εισόδου μία ακολουθία αλφαριθμητικών, τα οποία εισάγει σε ένα πίνακα κατακερματισμού  $T[0:m-1]$ . Όπως αναφέραμε παραπάνω, ο πίνακας αποθηκεύει ζεύγη  $\langle key, value \rangle$ , όπου στην περίπτωση μας το  $key$  είναι ένα αλφαριθμητικό και  $value$  είναι το πλήθος των εμφανίσεών του. Για την εκτέλεση του προγράμματος θα χρησιμοποιήσουμε για αρχείο εισόδου το `Dickens.txt` που περιέχει έργα του Καρόλου Ντίκενς.

Για τον πίνακα κατακερματισμού ξεκινάμε με μέγεθος  $m = 3000$  το οποίο διπλασιάζουμε κάθε φορά που ο συντελεστής πληρότητας του  $T$  γίνεται μεγαλύτερος από το κατώφλι  $f$  (πχ, για  $f = 80\%$ ).

Για την εκτέλεση των προγραμμάτων γράψτε

```
java ChainingHT < Dickens.txt
```

και

```
java LinearProbingHT < Dickens.txt.
```

Προκειμένου να αξιολογήσετε την απόδοση των δομών, πραγματοποιήστε τις παρακάτω μετρήσεις:

- Τυπώστε το πλήθος των διαφορετικών λέξεων που υπάρχουν στο αρχείο εισόδου, καθώς και το πλήθος των εμφανίσεων των λέξεων `and`, `astonished`, `boat`, `carol`, `city`, `scrooge`, `the`, `train` και `wondered`.
- Δοκιμάστε την απόδοση της δομής όταν χρησιμοποιούμε διαφορετικό αρχικό μέγεθος  $m$  του πίνακα κατακερματισμού, καθώς και διαφορετικές τιμές του  $f$ , πχ  $70\%$  ή  $90\%$ .
- Συγκρίνετε την απόδοση των προγραμμάτων σας με το έτοιμο πρόγραμμα `HashMapTest.java`, το οποίο χρησιμοποιεί τη δομή `HashMap` της Java.

Αποθηκεύστε τα παραπάνω αποτελέσματα στο αρχείο `lab6results.txt`.

### Παραδοτέα

Ανεβάστε στο eCourse ένα zip αρχείο με τα τελικά πρόγραμμα σας `ChainingHT.java` και `LinearProbingHT.java`, το έτοιμο πρόγραμμα `In.java`, καθώς και με το αρχείο των αποτελεσμάτων `lab6results.txt`.

Το zip αρχείο πρέπει να έχει όνομα που περιλαμβάνει το group του εργαστηρίου στο οποίο έχετε τοποθετηθεί καθώς και από το ID της ομάδας σας (π.χ., **G1\_ID1.zip**).