Πανεπιστήμιο Ιωαννίνων – Τμήμα Μηχανικών Η/Υ και Πληροφορικής
Δομές Δεδομένων [ΜΥΥ303] – Χειμερινό Εξάμηνο 2020

7η Εργαστηριακή Άσκηση
Κατακερματισμός
Παράδοση έως Πέμπτη 17/12, 12:00 από το eCourse

ΠΡΟΣΟΧΗ: Γράψτε σε κάθε αρχείο που παραδίδετε τα ονόματα και τους Α.Μ. των μελών της ομάδας σας. Συμπεριλάβετε όλα τα αρχεία σας (κώδικας Java και lab7results.txt) σε ένα zip αρχείο το οποίο να έχει το όνομα ενός μέλους της ομάδας σας με λατινικούς χαρακτήρες.

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 $< key, value >$, 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 $loadFactor = 100 \times n/m$, όπου n το πλήθος των αντικειμένων που έχουν εισαχθεί.

Ο κατακερματισμός υποστηρίζει τις ακόλουθες μεθόδους που καλείστε να υλοποιήσετε:

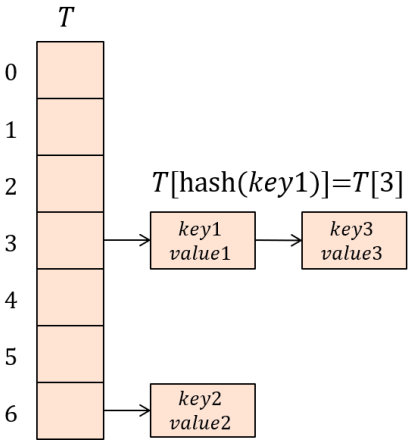
Value contains(Key key)	Επιστρέφει την τιμή $value$ που αντιστοιχεί στο κλειδί key. Αν το key δεν έχει εισαχθεί στη δομή τότε επιστρέφει $null$.
void insert(Key key, Value $value$)	Εισάγει στη δομή κατακερματισμού το κλειδί key με αντίστοιχη τιμή $value$. Αν το key υπάρχει ήδη στη δομή, τότε αντικαθιστά την προηγούμενη τιμή του με την τιμή $value$.

Η θέση στην οποία θα τοποθετηθεί ένα ζεύγος $< key, value >$ καθορίζεται από την τιμή της συνάρτησης κατακερματισμού

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

Κατακερματισμός με Χωριστές Αλυσίδες

Στο ChainingHT.java τα ζεύγη $< key, value >$ αποθηκεύονται σε συνδεδεμένες λίστες, όπου κάθε λίστα αντιστοιχεί σε μια θέση του πίνακα κατακερματισμού T. Η λίστα στην οποία θα τοποθετηθεί ένα ζεύγος $< key, value >$ δίνεται από την αναφορά $T[\text{hash}(key)]$, 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 $k = \text{hash}(key)$ του πίνακα κατακερματισμού και αναζητούμε το κλειδί key στη συνδεδεμένη λίστα με αναφορά $T[k]$. Για να ελέγξουμε αν το key είναι αποθηκευμένο σε ένα κόμβο x, ελέγχουμε αν $x.key.\text{equals}(key) == \text{true}$.

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 $k = \text{hash}(key)$ του πίνακα κατακερματισμού και αναζητούμε το key στη συνδεδεμένη λίστα με αναφορά $T[k]$. Για να ελέγξουμε αν το key είναι αποθηκευμένο σε ένα κόμβο x, ελέγχουμε αν $x.key.\text{equals}(key) == \text{true}$.

Κατακερματισμός με Γραμμική Βολιδοσκόπηση

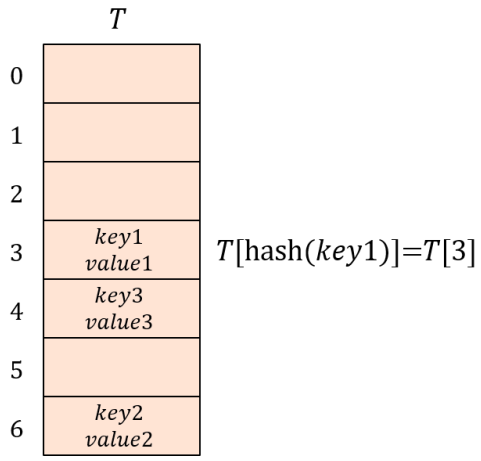
Σε αυτήν την μέθοδο, τα αντικείμενα αποθηκεύονται απευθείας στον πίνακα $T[0: m-1]$ τύπου Pair, με πεδία key και $value$. Επομένως, ο πίνακας T είναι τύπου Pair [].

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

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

όπου $k = \text{hash}(key)$.

Έστω ζεύγος $< key, value >$ με $\text{hash}(key) = k$.



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

  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 $< key, value >$ στη θέση j του πίνακα T, θέτοντας $T[j] = \text{new Pair}(key, value)$. 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 $\text{hashL}(k, i)$ αλλά με διπλάσιο m.

  Στη δεύτερη περίπτωση $(T[j].key.\text{equals}(key))$ 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 $\text{hashL}(k, 0)$, $\text{hashL}(k, 1)$, $\text{hashL}(k, 2)$, … , του πίνακα κατακερματισμού T μέχρι να βρούμε την πρώτη θέση $j =$

hashL(k, i) όπου είτε α) $T[j].key$.equals(key) == true, οπότε η αναζήτηση είναι επιτυχής, είτε β) $T[j] == null$ 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 $< key, value >$, 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 TomSawyerB. txt.

Για τον πίνακα κατακερματισμού ξεκινάμε με μια μικρή τιμή $m = 11$ 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 < TomSawyerB. txt

και

java LinearProbingHT < TomSawyerB. txt.

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

- Υπολογίστε το πλήθος των συγκρίσεων, δηλαδή των κλήσεων στη μέθοδο equals(), που γίνονται κατά την εισαγωγή όλων των λέξεων στον πίνακα κατακερματισμού. (Συμπεριλαμβανομένου και του πλήθους των συγκρίσεων που συμβαίνουν κατά το διπλασιασμό του πίνακα κατακερματισμού και την εισαγωγή των αντικειμένων στον νέο πίνακα.)

- Στη συνέχεια τυπώστε το πλήθος των εμφανίσεων των λέξεων and, astonished, boat, path, the, train, tom και wondered. Για κάθε μια από αυτές τις αναζητήσεις τυπώστε το πλήθος των συγκρίσεων που έγιναν.

- Δοκιμάστε την απόδοση της δομής όταν χρησιμοποιούμε 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, καθώς και με το πρόγραμμά σας από το 1ο εργαστήριο.

Αποθηκεύστε τα παραπάνω αποτελέσματα στο αρχείο lab7results. 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 lab7results. txt. Το zip αρχείο πρέπει να έχει το όνομα ενός μέλους της ομάδας σας με λατινικούς χαρακτήρες.