# *Δομές Δεδομένων και Αρχείων*

## Δυναμικός Κατακερματισμός (Dynamic Hashing)

Φλέγγας Γιώργος
2014030161

Τα προγράμματα που υλοποίησα για να την 2η άσκηση εκτελούν την μέθοδο του δυναμικού κατακερματισμού . Για την υλοποίηση του κώδικα κατακερματισμού αξιοποιήθηκαν οι μέθοδοι : α) Ανοιχτών διευθύνσεων και πιο συγκεκριμένα αυτή της γραμμικής εξέτασης(για την διαχείριση των συγκρούσεων ) και β) δυναμική κατανομή μνήμης .

## 1)Κεντρική Μνήμη Project2.1:

Το προγράμματα αυτό έχει 4 λειτουργίες : i)Εισαγωγή ενός αριθμού στοιχείων ,ii)Αναζήτηση ενός στοιχείου με βάση το κλειδί του , iii)Διαγραφή όλων των στοιχείων και iv) έξοδος από το πρόγραμμα και εκτύπωση του αριθμού συγκρίσεων που έγιναν.
i)put(**int** key, **int** value) Είναι η μέθοδος εισαγωγής ενός στοιχείου στο πρόγραμμα , η οποία τοποθετεί στον πινάκα table , ο οποίος έχει αρχικά μέγεθος 100, ένα HashEntry(**int** key, **int** value), στην θέση table[hash] με hash=key%ts. Το ts είναι το μέγεθος του πινάκα. Αρχικά θα ελέγξει αν στην θέση αυτή υπάρχει άλλο στοιχείο και αυτό δεν είναι DeletedEntry(εξήγηση στο iii).Αν στην θέση αυτή υπάρχει στοιχείο τότε το hash ,γίνετε hash = (hash + 1) % ts (δηλαδή η επόμενη θέση στον πίνακα) και ελέγχει αν υπάρχει ο αντιστοίχως χώρος στην μνήμη . Αν δεν υπάρχει ο απαραίτητος χώρος στην μνήμη τότε τον δεσμεύει δυναμικά , διπλασιάζοντας το μέγεθος του πινάκα . Μόλις βρεθεί θέση οπού δεν υπάρχει στοιχείο ή υπάρχει στοιχείο τύπου DeletedEntry, τότε τοποθετείτε το νέο HashEntry στην table[hash]. (NoCP είναι ο αριθμός συγκρίσεων στην εισαγωγή)
ii)search(**int** key) Είναι η μέθοδος της αναζήτησης ενός στοιχείου με βάση το key , η οποία ξεκινάει από την θέση hash=key%ts και ψάχνει μέχρι το τέλος του πίνακα και εμφανίζει όλα τα στοιχειά με table[hash].key=key .Αν δεν βρεθεί κάποιο στοιχειό τότε εμφανίζει: “ Not Found”. (NoSP είναι ο αριθμός συγκρίσεων στην αναζήτηση)
iii)remove(**int** key) Είναι η μέθοδος διαγραφής ενός στοιχείου από τον πινάκα με βάση το key,η οποία ξεκινάει από την θέση hash=key%ts και ψάχνει μέχρι το τέλος του πίνακα και διαγράφει όλα τα στοιχειά με table[hash].key=key. Κατα την διαγραφή αντικαθιστώ τα δεδομένα του table[hash] με αυτά ενός DeletedEntry (value=-1,key=-1) και αυτό γιατί στην περίπτωση που αφαιρούσαμε το στοιχείο εντελώς από τον πινάκα , η δομή του θα αλλοιωνόταν και δεν θα δούλευε σωστά ο αλγόριθμος της αναζήτησης (και της εισαγωγής) . (NoRP είναι ο αριθμός συγκρίσεων στην διαγραφή)
iv)Κατά την έξοδο από το πρόγραμμα εκτυπώνονται οι αριθμοί:NoCP,NosP,NoRP με σκοπό να συγκρίνουμε τα αποτελέσματα μεταξύ διαφορετικό αριθμό κλειδιών .

Να σημειωθεί ότι το πρόγραμμα διαχειρίζεται σωστά τις συγκρούσεις σε περίπτωση ίδιου κλειδιού, άλλα κατά την εκτέλεση τους δεν θα υπάρξει κάποια εκτός εάν προσθέτουμε πχ. 5 στοιχειά την φορά , τότε αν θέλουμε 20 στοιχειά θα έχουμε για παράδειγμα 4 στοιχειά με key=1 και θα βρίσκονται στις θέσεις :
hash[1]=1 value=4425 Key=1
hash[1]=6 value=5675 Key=1
hash[1]=11 value=5618 Key=1
hash[1]=16 value=-716 Key=1

Από την εκτέλεση με βάση την υπόδειξη της εκφώνησης προκύπτει ότι:

α)Για 20 στοιχειά με αναζήτηση για ένα κλειδί(πχ το 2) :
Number of compares during inputting:20
Number of compares during searching:19
Number of compares during deleting:230

β)Για 100 στοιχειά με αναζήτηση για ένα κλειδί(πχ το 65) :
Number of compares during inputting:100
Number of compares during searching:101
Number of compares during deleting:5150

γ)Για 1000 στοιχειά με αναζήτηση για ένα κλειδί(πχ το 734) :
Number of compares during inputting:2500
Number of compares during searching:267
Number of compares during deleting:501500

δ)Για 10000 στοιχειά με αναζήτηση για ένα κλειδί(πχ το 8999) :
Number of compares during inputting:22700
Number of compares during searching:1002
Number of compares during deleting:50015000

Παρατηρούμε μια μεγάλη αύξηση στα Number of compares during inputting/deleting μεταξύ των 100 και των 1000 εισαγωγών κάτι που πιθανός προκύπτει από το hash=key%ts , καθώς θα έχουμε αποτελέσματα (hash) που θα έχουν ήδη χρησιμοποιηθεί και θα έχουμε συγκρούσεις και θα αυξάνετε επομένως ο αριθμός συγκρίσεων .

## 2)Δίσκος Project2.2 :

Αποτελεί μια παραλλαγή του προγράμματος για το 1), με την διάφορα ότι έχουμε μια έξτρα μέθοδο την ToFile ,η οποία γράφει αντικείμενα τύπου HashEntry στο αρχείο table , η HashEntry κάνει implement το Serializable και η μέθοδος της αναζήτησης έχει τροποποιηθεί κατάλληλα (δεν υπάρχει μέθοδος για την διαγραφή ενός object από το αρχείο άλλα η remove() απλά αδειάζει το αρχείο).

Η μέθοδος ToFile() μεταφέρει ένα ένα τα object του table σε ένα αρχείο table, το οποίο αποτελείτε από σελίδες 128bytes . Αρχικά ελέγχει αν το object χωράει στην υπάρχουσα σελίδα,αν χωράει τότε κάνει την εισαγωγή, αλλιώς αλλάζει σελίδα και κάνει την εισαγωγή.

Η search(**int** key) κάνει αναζήτηση από την αρχή μέχρι το τέλος του αρχείου και εκτυπώνει ολα τα αντικειμενα με key=key.

Από την εκτέλεση με βάση την υπόδειξη της εκφώνησης προκύπτει ότι:

α)Για 20 στοιχειά με αναζήτηση για ένα κλειδί(πχ το 2) :
Number of times that accessed the drive during search:21
Number of times that accessed the drive during remove:1

β)Για 100 στοιχειά με αναζήτηση για ένα κλειδί(πχ το 65) :
Number of times that accessed the drive during search:101
Number of times that accessed the drive during remove:1

γ)Για 1000 στοιχειά με αναζήτηση για ένα κλειδί(πχ το 734) :
Number of times that accessed the drive during search:1001
Number of times that accessed the drive during remove:1

δ)Για 10000 στοιχειά με αναζήτηση για ένα κλειδί(πχ το 8999) :
Number of times that accessed the drive during search:10001
Number of times that accessed the drive during remove:1

Το αποτέλεσμα σε αυτή την περίπτωση είναι αναμενόμενο ,καθώς αφού κατά την αναζήτηση διαβάζουμε όλο το αρχείο μέχρι το τέλος του, τότε θα έχουμε N+1 προσβάσεις στον δίσκο , ενώ κατά την διαγραφή όλων των στοιχείων γίνετε 1 πρόσβαση στον δίσκο,όπου άπλα ανοίγουμε και κλείνουμε το αρχείο.

**Παράρτημα:**

1)Σημειώσεις του μαθήματος από το courses
2)Βιβλίο Δομές Δεδομένων και Αλγόριθμοι σε Java
3)http://www.algolist.net/Data_structures/Hash_table
4)https://en.wikipedia.org/wiki/Open_addressing
5)Βιβλίο Απόλυτη Java Savitch
Επίσης βοήθησα τον συμφοιτητή μου Ανδρέα Λάσκαρη .