Δομές Δεδομένων

Εργασία 2η

Μυρτώ-Χριστίνα Ελευθέρου|3170046

Χρυσούλα Οικονόμου|3170127

- *Μέρος Α*: Για την δημιουργία της ουράς προτεραιότητας, έχουμε προσαρμόσει τον κώδικα από τα εργαστήρια. Επίσης, δεν έχουμε κάνει χρήση generics.
- *Μέρος Β*: Για την υλοποίηση του Greedy αλγόριθμου, ξεκινάμε έχοντας έναν δίσκο με χωρητικότητα 1TB, ο οποίος αποθηκεύεται στην ουρά προτεραιότητας. Στην ουρά προτεραιότητας αποθηκεύονται αντικείμενα τύπου Disk, και το αντικείμενο με την μεγαλύτερη προτεραιότητα είναι αυτό που έχει τον μεγαλύτερο ελεύθερο χώρο. Διαβάζοντας το αρχείο στην αρχή, αποθηκεύουμε τους φακέλους σε μία λίστα files. Στην συνέχεια εκτελούμε ένα while loop μέχρι να αδειάσει η λίστα files. Αν ο τρέχον φάκελος χωράει στον δίσκο, τότε προσθέτουμε τον φάκελο στον folders του συγκεκριμένου δίσκου. Αν ο φάκελος δεν χωράει στον δίσκο, τότε δημιουργούμε έναν καινούργιο δίσκο. Στην συνέχεια ενημερώνουμε τον υπολειπόμενο χώρο του δίσκου, ο οποίος, αφού έχει τροποποιηθεί, θα πρέπει να ελέγξουμε αν πλέον έχει μικρότερη χωρητικότητα από κάποιον άλλον δίσκο, οπότε η προτεραιότητα του θα μειωθεί. Ταυτόχρονα, ταξινομούμε και τους δίσκους έτσι ώστε να εμφανιστούν ταξινομημένοι αν χρειαστεί.Τέλος, αφαιρούμε τον φάκελο από την λίστα files.

- *Μέρος Γ*:Για την ταξινόμηση των φακέλων στην λίστα files χρησιμοποιήσαμε τον αλγόριθμο ταξινόμησης MergeSort. Ο αλγόριθμος αυτός, ενημερώνει το head της λίστας, το οποίο είναι πλέον η μεγαλύτερη τιμή, ενώ κάθε κόμβος δείχνει στον αμέσως μικρότερο του. Έτσι έχουμε πετύχει φθίνουσα ταξινόμηση της λίστας. Επειδή με τον αλγόριθμο αυτόν ενημερώνεται μόνο το head, έχουμε τροποποιήσει ελάχιστα την μέθοδο remove(), της κλάσης List, επειδή το tail δεν ενημερωνόταν, οπότε σε κάποιες περιπτώσεις ο αλγόριθμος έβγαζε λανθασμένο αποτέλεσμα.
- *Μέρος Δ*:Για την παραγωγή των αρχείων, αρχικά θέτουμε 3 τιμές, οι οποίες θα είναι το πλήθος των φακέλων για τα 10 αρχεία. Στην συνέχεια, έχοντας 3 διαφορετικά for loops(ένα για κάθε τιμή του N), εκτελούμε την επαναληπτική διαδικασία 10 φορές. Το όνομα του αρχείου προκύπτει από την τιμή του N και τον αριθμό του i. Τέλος, δημιουργούμε ένα αντικείμενο τύπου Random και χρησιμοποιώντας την συνάρτηση nextInt δημιουργείται ένας αριθμός ανάμεσα στο 0 και το 1000000. Για να τρέξουμε ταυτόχρονα και τους δύο αλγόριθμους σε ένα αρχείο, αντιγράφουμε την λίστα files, η οποία περιέχει τις τιμές του αρχείου, σε μία λίστα sortedList, την οποία ταξινομούμε. Έπειτα, εκτελούμε ακριβώς την ίδια διαδικασία που κάναμε για τον greedy αλγόριθμο. Δηλαδή, φτιάχνουμε μη $2^{η}$ ουρά προτεραιότητας, η οποία περιέχει μόνο τα στοιχεία της ταξινομημένης λίστας. Στην συνέχεια, εκτελούμε ένα while loop, ακριβώς ίδιο με αυτό που χρειάστηκε για τον greedy αλγόριθμο.

  Για **N=80**, ο greedy αλγόριθμος μας δίνει μέσο όρο 47,6 ενώ, ο greedy-decreasing μας δίνει μέσο όρο 43.

Για **N=200**, ο greedy αλγόριθμος μας δίνει μέσο όρο 118,1 ενώ, ο greedy-decreasing μας δίνει μέσο όρο 104,4.

Για **N=800**, ο greedy αλγόριθμος μας δίνει μέσο όρο 469,5 ενώ, ο greedy-decreasing μας δίνει μέσο όρο 408,8.

Παρατηρούμε ότι και για τις 3 περιπτώσεις του N, ο δεύτερος αλγόριθμος είναι πιο αποδοτικός καθώς ο Μ.Ο. που προκύπτει είναι μικρότερος. Είναι χρήσιμο να σημειωθεί ότι σε κανένα από τα 30 αρχεία, δεν προέκυψε ο πρώτος αλγόριθμος να είναι πιο αποδοτικός, δηλαδή να μας δώσει λιγότερους δίσκους. Επίσης, είναι εμφανές ότι όσο αυξάνεται η τιμή του N, τόσο μεγαλύτερη θα είναι η διαφορά των μέσων όρων, καθώς για πολύ μικρές του N(8,10,15), ο μέσος όρος ήταν αρκετά κοντά, αλλά και πάλι μικρότερος ήταν αυτός του $2^{ου}$ αλγορίθμου. Συμπερασματικά, ο αλγόριθμος greedy-decreasing είναι αποδοτικότερος.