Αναφορα Παραδοσης Εργασιας 2 στο Μαθημα των Δομων Δεδομενων (INF231)

Σασσαλος Ρηγας (3220178)
Λαζανα Ευγενια (3220104)

Τμημα Πληροφορικης / Οικονομικο Πανεπιστημιο Αθηνων
Διδασκουσα: Κατια Παπακωνσταντινοπουλου
Έτος: 2023 - 2024

ΥΛΟΠΟΙΗΣΗ ΤΑΞΙΝΟΜΗΣΗΣ ΜΕ ΤΗΝ ΜΕΘΟΔΟ QUICKSORT (ΜΕΡΟΣ Α):

Στο μέρος Α της εργασίας χρησιμοποιήσαμε την μέθοδο Quicksort για να ταξινομήσουμε τον πίνακα.

Η Quicksort χρησιμοποιεί ένα pivot element το οποίο χωρίζει τον πίνακα σε δύο υποπίνακες τους ταξινομεί και τους ενώνει. Είναι πολύ χρήσιμος γιατι μπορει να κάνει ταξινόμηση αποδοτικά με πολυπλοκότητα O(n*logn). Αφού χωρίσει τον πίνακα σε δύο υποπίνακες ξεκινάει να ταξινομεί τον κάθε έναν ξεχωριστά με την χρήση της αναδρομής και καταλήγει να μετακινεί τις πόλεις με το μικροτερο density αριστερά απο το pivot element και τις πόλεις με το μεγαλύτερο density δεξία απο το pivot element. Επαναλαμβάνει την διαδικασία και ενώνει σιγά σιγά τα ταξινομημένα κομμάτια του πίνακα μέχρι να ενωθεί όλος ο πίνακας.

Στην συγκεκριμένη περίπτωση η μέθοδος quicksort στο Influenza_k.java χρησιμοποιεί την μέθοδο partition που χωρίζει τον πίνακα ανάλογα και την μέθοδο swap που αλλάζει την θέση δυο στοιχείων στον πίνακα.

ΥΛΟΠΟΙΗΣΗ ΜΕΘΟΔΟΥ REMOVE ΣΤΗΝ PQ.JAVA (ΜΕΡΟΣ Β):

Η μέθοδος remove στην υλοποίηση της ουράς προτεραιότητας με χρήση πίνακα φτιάχτηκε ως εξής:

Αρχικά, ελέγχουμε αν η ουρά προταιρεοτητάς μας είναι άδεια με την χρήση της μεθόδου isEmpty. Αν η ουρά είναι άδεια βγάζουμε error στον χρήστη “Priority queue is empty”. Αλλιώς, χρησιμοποιώντας τον υποβοηθητικό πίνακα ακεραίων position που αποθηκεύει την θέση της κάθε πολης με βάση το ID ελέγχουμε αν το ID που μας έδωσε ο χρήστης αντιστοιχεί σε κάποια πόλη. Αν δεν αντιστοιχεί βγάζουμε error στον χρήστη “City not found”. Αλλιώς, χρησιμοποιώντας την swap βάζουμε το στοιχείο στο τέλος της ουράς και αλλάζουμε την τιμή σε null. Ταυτόχρονα, αλλάζουμε την θέση position σε -1 και μειωνουμε το μέγεθος των ενεργών στοιχείων στην ουρα καθώς και χρησιμοποιούμε την μέθοδο sink για να κατεβάσουμε το στοιχείο στο τέλος της ουράς.

ΥΛΟΠΟΙΗΣΗ ΜΕΡΟΥΣ Γ:

Το μέρος Γ χρησιμοποιεί την υλοποιήση της ουράς προτεραιότητας με χρήση πίνακα του μέρους Β για να εφαρμόσει πιο αποδοτικά την διαδικασία στο μέρος Α. Δέχεται απο τον χρήστη τις ίδιες ακριβώς παραμέτρους (k και μονοπάτι αρχείου) και τις αποθηκεύει στις αντίστοιχες μεταβλητές. Δημιουργεί ενα αντικείμενο της ουράς και αρχίζει να διαβάζει μια μια τις σειρές του txt αρχείου και αφου χωρίζει ανα κενό ολα τα στοιχεία και τα αποθηκεύει στον πίνακα data ξεκινάει να τα προσθέτει στις αντίστοιχες μεταβλητές. Τέλος, φτιάχνει ένα αντικείμενο της κλασης City με τα στοιχεία αυτα και προσθέτει το αντικείμενο αυτό στην ουρά. Αυτό γίνεται επαναληπτικά μέχρι να προσθεθούν k πόλεις στην ουρά. Μόλις γινει αυτό συνεχίζει το πρόγραμμα να διαβάζει και να ξεχωρίζει τα στοιχεία του αρχείου και να δημιουργεί αντικείμενα City. Βρίσκει την πόλη με το μεγαλύτερο density που βρίσκεται μέσα στην ουρά με την χρήση της μεθόδου findMaxCity που υλοποιήσαμε στο PQ.java και συγκρίνει την επόμενη πόλη στο αρχείο με την πόλη που επιστρέφεται απο την

μέθοδο. Αν η πόλη που βρήκαμε στο αρχειο εχει μικρότερο density κρουσμάτων απο την maxCity αφαιρούμε απο την ουρά την maxCity και προσθέτουμε την πόλη απο το αρχείο.

Έτσι, αφού τελειώσει η επανάληψη βρίσκονται στην ουρα οι k πόλεις με την μικρότερη πυκνότητα κρουσμάτων.

Όσον αφορά την πολυπλοκότητα της υλοποίησης του μέρους Γ εξαρτάται κυρίως απο το k και ειναι O(n log k) οπου n ο αριθμός των στοιχείων που βρίσκονται στο .txt αρχείο. Το μέρος Γ ειναι πολύ πιο συμφέρον σε περιπτώσεις που το k ειναι πολυ μικρότερο απο τον n αριθμό πόλεων στο .txt αρχειο αφου στο μερος Α η πολυπλοκότητα είναι O(n log n) ενω υπάρχει και η περίπτωση να έχει πολυπλοκότητα O(n^2) λόγω της Quicksort.