ΕΡΓΑΣΙΑ 2-ΜΕΡΟΣ Ε

Γεωργιος Κοτσιφος,ΑΜ:3190093

Κωνσταντινος Μαρκο,ΑΜ:3190112

ΜΕΡΟΣ Α:

Ξεκιναμε φτιαχνοντας την διεπαφη CityInterface με τις δοσμενες μεθοδους απ'την εκφωνηση και στη συνεχεια την κανουμε implement στη κλαση City μαζι με την Comparable<City> διοτι θα την χρειαστουμε αργοτερα στην υλοποιηση της quicksort.**Στη συνεχεια φτιαχνοντας την Covid_k με 2 Scanners διαβαζουμε το ονομα του file με καταληξη .txt(π.χ inputfile.txt) με τις πολεις(το οποιο θα πρεπει να βρισκεται στην ιδια τοποθεσια με το source file) και στη συνεχεια διαβαζουμε το k που πληκτρολογει ο χρηστης(Δεν χρειαζεται να μπει το path του file αλλα μονο το ονομα του!).**

Στο μερος Α υλοποιησαμε τον αλγοριθμο ταξινομησης QuickSort. Η QuickSort είναι κατηγοριας διερει και βασιλευε δηλαδη διαλεγει ένα στοιχειο που το ονομαζουμε "pivot" και διαμεριζουμε τον πινακα σε 2 υποπινακες αναλογα με το αν είναι μεγαλυτερα του pivot ή όχι. Επειτα παιρνει αυτους τους δυο υποπινακες και εκτελει ξανα την ιδια διαδικασια αναδρομικα. Θα καταληξει να εχει διαμερισει τον πινακα σε μια σειρα 1-στοιχειου πινακα. Αυτά τα κανει με την χρηση της μεθοδου "partition". Στο τελος όλα τα στοιχεια θα εχουν γινει pivot αρα θα είναι στην σωστη τους θεση. Όλα αυτά τα κανει σε ένα μερος δηλαδη δεν χρησιμοποιει βοηθητικο πινακα σε αντιθεση με την MergeSort.

Στην εργασια μας αυτό που κανει η QuickSort είναι να ταξινομει τις πολεις με βαση τα κρουσματα ανα 50.000 κατοικους.

ΜΕΡΟΣ Β:

Η remove δεχεται ως ορισμα το id μιας πολης και αναλογα με αυτο την αφαιρει και την επιστρεφει. Δυστηχως δεν καταφεραμε να την

υλοποιησουμε σε πολυπλοκοτητα O(logN) αλλα σε O(N). Αρχικα κανουμε ένα iteration στην ουρα μεχρι να βρεθει το στοιχειο με το ID που ζηταμε. Μετα κραταει μια αναφορα στο root στοιχειο και αντικατιστα το στοιχειο της ριζας με αυτό στο δεξι φυλλο. Επισης αφαιρει το “size” κατά ενα δηλαδη το μεγεθος της ουρας και καλει την μεθοδο “sink” για να επαναφερει την ιδιοτητα της ουρας.

ΜΕΡΟΣ Γ:

Στο μερος Γ αρχικα δημιουργουμε μια ουρα PQ (με ονομα “priorityQueue”) στην οποια θα εισαγουμε αντικειμενα τυπου “City”.Η PQ αρχικα είναι αδεια και επειτα γεμιζει με τις πολεις του file(μολις γεμισει, ταξινομει αμεσως τις πολεις). Το αντικειμενο τυπου City το ονομασαμε “NewCity”. Χρησιμοποιουμε μια while για να εισαγουμε στο NewCity τα χαρακτηριστικα της πολης(ID,population,covid cases…) με την χρηση μια μεταβλητης με ονομα “antikeimeno” η οποια αυξανεται κατά 1 κάθε φορα που τελειωνει το loop. Όταν το antikeimeno==0 εισαγει το ID, όταν το antikeimeno==1 εισαγει το ονομα της πολης και παει λεγοντας, μεχρι να εχει εισαγει όλα τα χαρακτηριστικα της πολης δηλαδη όταν antikeimeno==4 και τοτε το “antikeimeno” μηδενιζεται. Επιπλεον τοτε εισαγουμε το NewCity στην PQ (γραμμη 66 στον κωδικα). Τελος εχουμε και μια μεταβλητη “numberOFIterations” η οποια κάθε φορα που διαβαζεται από το file μια πολύ αυξανεται κατά 1. Όταν αυτή η μεταβλητη εχει τον ιδιο αριθμο με το k τοτε το προγραμμα σταματαει και εκτυπωνει μονο τις k αρχικες πολεις του file ταξινομημενες με βαση τα κρουσματα ανα 50.000 κατοικους. Επειδη χρησιμοποιουμε την μεθοδο insert η οποια είναι πολυπλοκοτητας O(logN) και χρησιμοποιυμε ένα while loop O(N) συνολικα το “DynamicCovid_k_with_PQ” θα είναι πολυπλοκοτητας O(logN*N)=O(NlogN).

ΜΕΡΟΣ Δ:

Στο μερος Δ εγινε η ιδια διαδικασια με αυτή στο μερος Γ για να εισαγουμε τις πολεις με τα στοιχεια τους σε μια ουρα PQ με ονομα “priorityQueue1”. Επισης θα χρησιμοποιησουμε και δευτερη ουρα με

ονομα “priorityQueue2”. Αρχικα ελεγχουμε εάν οι πολεις που διαβασαμε δινουν υπολοιπο 0 εάν τις διαιρεσουμε με τον 5(γραμμη 70 στο κωδικα). Τοτε σε αυτή την περιπτωση αρχικουποιουμε μια μεταβλητη “median” με τον αριθμο των επαναληψεων(δηλαδη των αριθμων τον πολεων που εχουμε διαβασει εως τωρα) δια 2. Π.χ Εάν διαβαστηκαν 5 πολεις το Median θα είναι =3.Μετα παμε σε μια for loop(γραμμη 75 στον κωδικα) η οποια διαγραφει τις πολεις που βρισκονται στο priorityQueue1 μεχρι να φτασει στην μεσαια(δηλαδη στην median) και αυτές που διαγραφει μετα τις εισαγει στην priorityQueue2(Οι πολεις είναι ηδη ταξινομημενες με βαση τα κρουσματα ανα 50.000 κατοικους στην priorityQueue1 αρα η επιθυμητη πολη θα βρισκεται στην μεση της ουρας). Μολις φτασει στην μεση δηλαδη στην πολη που ζηταμε, την διαγραφει και την εκτυπωνει και επιπλεον την εισαγει στην priorityQueue2. Τελος αυτό που κανει το προγραμμα είναι να διαγραψει ολες τις πολεις που υπαρχουν στην priorityQueue2 και να τις εισαγει ξανα στην priorityQueue1 ώστε να επαναλαβει ξανα την ιδια διαδικασια από την αρχη(σε περιπτωση που οι πολεις του file είναι περισσοτερες από 5). Επειδη χρησιμοποιυμε ξανα την inset O(logN) και χρησιμοποιουμε μια loop O(N) για τις N εισαγωγες, συνολικα το Μερος Δ θα είναι πολυπλοκοτητας O(NlogN).