ΑΝΑΦΟΡΑ 2ΗΣ ΕΡΓΑΣΙΑΣ

Μέρος Α:

Ο αλγόριθμος ταξινόμησης που επιλέξαμε να υλοποιήσουμε στο μέρος Α της εργασίας είναι η mergeSort.

Η ταξινόµηση µε συγχώνευση αποτελείται από 2 αναδροµικές κλήσεις σε πίνακες µε το µισό αρχικό µέγεθος του αρχικού, πάνω στα οποία θα ταξινομήσει. Η αναδροµή τελειώνει όταν ο πίνακας έχει µόνο ένα στοιχείο. Χρησιµοποιεί τη µέθοδο merge στο τέλος για να συγχωνεύσει τους 2 υποπίνακες που χωρίστηκαν νωρίτερα.

Η μέθοδος sort είναι αναδρομική και ταξινομεί τους δύο υποπίνακες οι οποίοι χωρίζονται στη μέση. Μόλις τελειώσει η κάθε αναδρομή, καλείται η μέθοδος merge η οποία συγχωνεύει τους πίνακες της κάθε αναδρομής ώστε στο τέλος να λάβουμε έναν τελικό ταξινομημένο πίνακα ο οποίος έχει προκύψει από όλη αυτήν τη διαδικασία.

Μέρος Β:

Η μέθοδος remove υλοποιήθηκε με πολυπλοκότητα Ο(logn), με την βοήθεια του IdArray. Αυτός ο πίνακας των 9999 θέσεων κρατούσε την θέση του κάθε τραγουδιού στο heap στο κελί που αντιστοιχούσε στον αριθμό ID κάθε τραγουδιού. Αρχικοποιούταν με τις εισαγωγές κάθε τραγουδιού, και ενημερώνοταν με τις ελάχιστη δυνατή επιβάρυνση του προγράμματος στις sink και swim. Όταν η remove καλείται, ανατρέχει σε αυτήν την θέση του πίνακα, βρίσκει την θέση του τραγουδιού που θα αφαιρεθεί στο heap, και το αντικαθιστά με ένα τραγούδι που έχει πιο πολλά “likes” από το πρώτο σε προτεραιότητα . Τέλος, η swim τοποθετεί το τραγούδι στην κορυφή, και η getMax το αφαιρέι, χωρίς καμία βλάβη στην προτεραιότητα των άλλων τραγουδιών.

Μέρος Γ:

Το μέρος Γ είναι ουσιαστικά η υλοποίηση του μέρους Β, στο οποίο αναπτύξαμε μια μεγιστοστρεφή ουρά προτεραιότητας. Στο μέρος Γ αρχικοποιούμε την άδεια αυτή ουρά με διπλάσιο μέγεθος από το απαιτούμενο και με μια υποκλάση του java.util.Comparator, την SongComparator (κατατάσει τα τραγούδια με φθίνοντα αριθμό “like”, και σαν δευτερεύουσα συγνκριση προηγείται το τραγούδι µε λεξικογραφικά µικρότερο όνοµα) . Στη συνέχεια, κατά το διάβασμα του αρχείου, η ουρά αυτή κρατάει πάντοτε τα κ καλύτερα τραγούδια στη μνήμη του. Μόλις γεμίσει αρχικά η ουρά προτεραιότητας με τα κ καλύτερα τραγούδια, κατά την πρόσθεση νέων τραγουδιών, ξεκινά η διαδικασία την αφαίρεσης του τραγουδιού με τη χαμηλότερο αριθμό “like”(ή του λεξικογραφικά μεγαλύτερου), δεδομένου ότι το νέο τραγούδι που εισάγουμε έχει likes μεγαλύτερα ή ίσα με εκείνα του χειρότερου τραγουδιού στην ουρά προτεραιότητας. Με το διάβασμα κάθε γραμμής δημιουργείται ένα αντικείμενο τραγουδιού που συγκρίνεται με το ‘χειρότερο’ τραγούδι. Προκειμένου να βρεθεί αυτό το τραγούδι που είναι υποψήφιο προς αφαίρεση, χρησιμοποιείται σειριακή αναζήτηση (πολυπλοκότητας Ο(ν) στην χειρότερη περίπτωση), συγκρίνεται με το υποψήφιο προς εισαγωγή τραγούδι και αν το 2ο αποδειχθεί ‘καλύτερο’, προστίθεται στην ουρά, ενώ το άλλο αφαιρείται. Οι διαδικασίες αυτές γίνονται με πολυπλοκότητα Ο(logn) στην χειρότερη περίπτωση. Θεωρούμε πως η συγκεκριμένη υλοποίηση συμφέρει για αρκετά μικρότερα κ από το μέγεθος του αρχείου, καθώς το άνω οριο είναι Ο(ν), που στις περισσότερες περιπτώσεις(γι αυτό μόνο το πρόγραμμα) είναι μικροτερο του Θ(nlogn) που είναι η πολυπλοκότητα της Merge sort σε κάθε περίπτωση.

Στην περίπτωση του υπολογισμού του Median, στο μέρος Δ χρησιμοποιούμε μια δομή που περιέχει 2 ουρές προτεραιότητας του μέρους Β( Ο(logn) πολυπλοκότητα), με αντίθετους Comparators. Η μέθοδος insert, που χρησιμοποιεί τις insert, max και getMax των ουρών είναι και αυτή πολυπλοκότητας O(logn), και η μέθοδος showMedian είναι πολυπλοκότητας Ο(1). Κάθε φορά που Ι%5==0 ο median βρίσκεται στην μέγιστη θέση (1) της ουράς above. Στην ουσία ο median υπολογίζεται δυναμικά ανά πάσα στιγμή με πολυπλοκότητα O(logn).