Δομές Δεδομένων - Εργασία 2

Δανοπούλου Έμυ (3170033)

Μπαλή Νίκη (3170114)

Μέρος Α

Στο Μέρος Α χρησιμοποιήσαμε τον αλγόριθμο ταξινόμησης Quicksort.

Η Quicksort αποτελεί την καλύτερη μέθοδο ταξινόμησης ως σήμερα και λειτουργεί με βάση την ιδέα «διαίρει και βασίλευε».

Αυτό που συμβαίνει στην Quicksort είναι ότι, με τη χρήση πινάκων, επιλέγουμε τυχαία μία θέση του πίνακα – τη λεγόμενη pivot – και με βάση αυτή τη θέση, χωρίζουμε τον αρχικό μας πίνακα σε δύο υποπίνακες.

Με τη μέθοδο της διαμέρισης (partition) βρίσκουμε την τελική θέση του pivot και ταξινομούμε τους υποπίνακες, έτσι ώστε όλα τα στοιχεία του αριστερού υποπίνακα να είναι μικρότερα ή ίσα του pivot και όλα τα στοιχεία του δεξιού υποπίνακα να είναι μεγαλύτερα ή ίσα του pivot.

Μετά την ταξινόμηση των υποπινάκων, το pivot παίρνει την τιμή της αμέσως προηγούμενης θέσης του πίνακα και η διαδικασία επαναλαμβάνεται.

Πολυπλοκότητα: O(NlogN) [average case]

Μέρος Β

Song remove(int id):

Η μέθοδος remove παίρνει ως όρισμα έναν ακέραιο αριθμό που αντιπροσωπεύει το id ενός αντικειμένου Song. Μέσω ενός for loop, διατρέχουμε όλα τα στοιχεία του σωρού και με κάθε επανάληψη ελέγχουμε εάν το id του ορίσματος μας είναι το ίδιο με το id του τρέχοντος στοιχείου. Όταν βρούμε το Song με το id που ψάχνουμε, δημιουργούμε ένα νέο αντικείμενο Song με τα ίδια χαρακτηριστικά. Μετά αντιμεταθέτουμε αυτό το στοιχείο με το τελευταίο στοιχείο του σωρού και διαγράφουμε το στοιχείο με το ζητούμενο id. Έπειτα, κάνουμε swim το τελευταίο στοιχείο του σωρού έτσι ώστε να αποκατασταθεί η ιδιότητα του σωρού και επιστρέφουμε το Song που είχαμε δημιουργήσει. Εάν το id που δόθηκε ως όρισμα δεν αντιστοιχεί σε κάποιο Song, απλά επιστρέφουμε ένα κενό Song.

Μέρος Γ

Στην κλάση Top_k_withPQ θα χρησιμοποιήσουμε μία ουρά προτεραιότητας ώστε να υπολογίζουμε δυναμικά τα κορυφαία k τραγούδια.

Όλη η διαδικασία υλοποιείται με τη μέθοδο topsongs.

Η μέθοδος topsongs είναι τύπου void και παίρνει ως όρισμα τη μεταβλητή k που έχει διαβαστεί απ'το πληκτρολόγιο. Ξεκινάμε δημιουργώντας μία ουρά προτεραιότητας τύπου PQ, η οποία έχει χωρητικότητα 2*k και χρησιμοποιεί ως comparator ένα αντικείμενο SongComparator.

Στη συνέχεια διαβάζουμε το αρχείο μας ανά γραμμή με τη βοήθεια ενός Scanner. Σε κάθε γραμμή διαβάζουμε το id, τον τίτλο και τον αριθμό likes ενός τραγουδιού και τα θέτουμε ως ορίσματα για ένα νέο αντικείμενο Song, το οποίο ονομάζουμε s.

Μετά ελέγχουμε εάν ο σωρός είναι άδειος με τη βοήθεια της μεθόδου isEmpty() και εάν είναι, κάνουμε insert το αντικείμενο που μόλις φτιάξαμε στο σωρό. Αυτή η ενέργεια θα πραγματοποιηθεί μόνο μία φορά, για την πρώτη γραμμή κειμένου.

Εάν ο σωρός δεν είναι άδειος, τότε δημιουργούμε ένα νέο αντικείμενο Song το οποίο αντιστοιχεί στο τελευταίο στοιχείο του σωρού της ουράς προτεραιότητας. Το αντικείμενο αυτό έχει προφανώς τη μικρότερη προτεραιότητα.

Έπειτα ελέγχουμε εάν έχουμε ήδη k τραγούδια αποθηκευμένα στην ουρά και εάν ναι, τότε αφαιρούμε το τελευταίο στοιχείο από τη σωρό. Στη συνέχεια εισάγουμε στην ουρά το αντικείμενο s και τέλος, εκτυπώνουμε τους τίτλους των k κορυφαίων τραγουδιών με χρήση των getmax() και getTitle().

Πολυπλοκότητα προγράμματος = O(NlogN)

Τελικά δεν συμφέρει να χρησιμοποιήσουμε το Top_k_withPQ σε περιπτώσεις όπου το k είναι αρκετά μικρότερο του συνολικού αριθμού τραγουδιών διότι γίνονται πολλές περιττές συγκρίσεις και προσθήκες/αφαιρέσεις στη ουρά προτεραιότητας, οι οποίες θα μπορούσαν να είχαν αποφευχθεί με κάποια άλλη υλοποίηση.