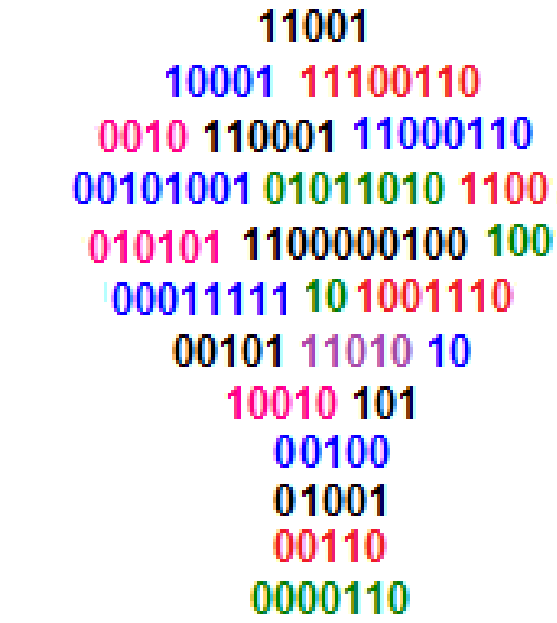
ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

UNIVERSITY OF PIRAEUS



Data Structure

Τεχνική Αναφορά

ΠΡΩΤΗΣ ΕΞΑΜΗΝΙΑΙΑΣ ΑΣΚΗΣΗΣ

Αλέξανδρος Μπαμπούνης-Τσάτσος | Δομές δεδομένων | 2/12/2020

Περιεχόμενα

ΣΗΜΕΙΩΣΕΙΣ ΚΩΔΙΚΑ

Μέθοδος insert(Record poi)

Η μέθοδος insert είναι υπεύθυνη για την εισαγωγή κόμβων σε απλά συνδεδεμένη λίστα. Εδώ έχει δοθεί έμφαση στο score ώστε όταν εισάγεται ένας νέος κόμβος να μπαίνει στη σωστή θέση ώστε η λίστα να είναι ταξινομημένη πάντα ως προς το score σε φθίνουσα σειρά. Με αυτό τον τρόπο επωφελούμαστε στην υλοποίηση της highScore. Επιπλέον αν θεωρήσουμε ότι ο κώδικας χρησιμοποιείται σε μια εφαρμογή εύρεσης π.χ. εστιατορίων, τα πρώτα αποτελέσματα θα εμφανίζονται ταξινομημένα ως προς score.
Εδώ, λόγω του ότι κατά την εισαγωγή νέου κόμβου διατρέχουμε όλη τη λίστα και ελέγχουμε κάθε κόμβο ως προς το score, στη χειρότερη περίπτωση ο αλγόριθμος θα φτάσει να εισάγει το νέο κόμβο στο τέλος της λίστας. Άρα η πολυπλοκότητα της insert είναι O(n).

Μέθοδος nearest(Point p, int k)

Η μέθοδος κατασκευάζει και επιστρέφει λίστα που περιέχει τις εγγραφές των k πιο κοντινών σημείων ενδιαφέροντος από ένα σημείο p. Εδώ χρησιμοποιούνται δύο λίστες, distances και result_list. Η distances όπου αποθηκεύονται οι επιμέρους αποστάσεις των σημείων της λίστας στην οποία καλείται η μέθοδος και η result_list που επιστρέφεται ως αποτέλεσμα. Κατά τη δημιουργία της distances οι κόμβοι ταξινομούνται κατά score με φθίνουσα σειρά και αυτό για να υπάρχει συσχετισμός μεταξύ της distances και της λίστας στην οποία καλείται η μέθοδος. Το κόστος εδώ είναι $T(n) = n*(n*(c)) = n^2 *c$ άρα η πολυπλοκότητα είναι τάξης $O(n^2)$.

Μέθοδος nearest(Point p, double maxDist)

Η μέθοδος κατασκευάζει και επιστρέφει λίστα που περιέχει τις εγγραφές των k σημείων ενδιαφέροντος από ένα σημείο p που απέχουν το πολύ maxDist.
Εδώ δημιουργείται μια νέα λίστα τύπου RankList με όνομα result_list στην οποία θα αποθηκεύονται οι εγγραφές των σημείων που πρέπει να επιστραφούν. Για να βρεθούν, διατρέχεται όλη τη λίστα και για κάθε κόμβο υπολογίζεται η απόσταση από το δοθέν p αποθηκεύεται στη μεταβλητή target_distance. Έπειτα ελέγχεται αν η απόσταση που υπολογίστηκε είναι μικρότερη από το maxDist και αν ναι, τότε το αντίγραφο του εκάστοτε κόμβου εισάγεται στη result_list και εν τέλει η μέθοδος επιστρέφει τη result_list. Το κόστος εδώ είναι $T(n) = n*(c+c) = n*2*c$ συνεπώς η πολυπλοκότητα του είναι O(n).

Μέθοδος highScore(int k)

Η μέθοδος κατασκευάζει και επιστρέφει λίστα που περιέχει τις k πρώτες εγγραφές των σημείων ενδιαφέροντος με την υψηλότερη βαθμολογία. Εδώ, λόγω του τρόπου κατασκευής της insert, κάθε λίστα που παράγεται είναι ήδη ταξινομημένη κατά score με φθίνουσα σειρά. Άρα αρκεί να πάρουμε τις k πρώτες εγγραφές. Εδώ το κόστος είναι T(n) = n*c άρα η πολυπλοκότητα του είναι O(n).

Μέθοδος highScore(double minScore)

Η μέθοδος κατασκευάζει και επιστρέφει λίστα που περιέχει τις εγγραφές των σημείων ενδιαφέροντος με βαθμολογία τουλάχιστον minScore από τη λίστα στην οποία κλήθηκε. Εδώ διατρέχεται όλη η λίστα στην οποία κλήθηκε η μέθοδος και για κάθε σημείο ενδιαφέροντος που έχει βαθμολογία τουλάχιστον minScore, το αντίγραφο του αποθηκεύεται στη λίστα result_list, η οποία εν τέλει επιστρέφεται. Το κόστος εδώ είναι T(n) = n*c άρα η πολυπλοκότητα του είναι O(n).

Μέθοδος inCommonWith(RankList rankList)

Η μέθοδος επιστρέφει λίστα με τις κοινές εγγραφές ανάμεσα σε δύο λίστες. Εδώ χρησιμοποιούνται δύο αναφορές τύπου Node cur1 και cur2 που διατρέχουν αντίστοιχα τις λίστες. Με χρήση δύο εμφολευμένων while βρόχων, ένας για κάθε λίστα, για κάθε επανάληψη του έξω, διατρέχεται η λίστα rankList του ορίσματος. Αν βρεθούν κοινά id πεδία τότε το αντίγραφο του κόμβου που βρέθηκε στη rankList αποθηκεύεται σε τρίτη λίστα result_list, η οποία εν τέλει επιστρέφεται. Το κόστος εδώ είναι T(n) = n*(n*(c)) = n^2 *c άρα η πολυπλοκότητα του είναι τάξης O(n^2).

ΕΡΩΤΗΣΕΙΣ ΕΚΦΩΝΗΣΗΣ

Δεδομένου ότι υπάρχουν n στοιχεία στη λίστα.

Εύρεση:

- των k εγγύτερων σημείων ενδιαφέροντος σε δεδομένο σημείο με δεδομένη ελάχιστη βαθμολογία.

Αρχικά θα πρέπει να βρεθεί το σημείο με την ελάχιστη βαθμολογία και δεδομένης της υλοποίησης της insert θα πρέπει να πάρουμε το τελευταίο στοιχείο. Συνεπώς n επαναλήψεις. Έπειτα δεδομένου ενός σημείου θα πρέπει να βρούμε τα k κοντινότερα. Άρα δεδομένης της υλοποίησης της πρώτης nearest n^2 πράξεις. Εν τέλει θα είναι T(n) = n + n^2 άρα O(n^2)

- των k πιο υψηλόβαθμων σημείων ενδιαφέροντος σε απόσταση το πολύ d από δεδομένο σημείο.

Εδώ θα χρησιμοποιήσουμε τη δεύτερη nearest, και επειδή η λίστα που επιστρέφεται είναι ταξινομημένη, αρκεί να πάρω τις k πρώτες εγγραφές. Άρα η πολυπλοκότητα είναι O(n).

- των σημείων ενδιαφέροντος με δεδομένη ελάχιστη βαθμολογία, σε απόσταση το πολύ d από δεδομένο σημείο.

Εδώ θα χρησιμοποιήσουμε την πρώτη highScore και τη δεύτερη nearest. Θα πρέπει πρώτα να βρεθούν τα σημεία που απέχουν το πολύ d από ένα δεδομένο σημείο και για κάθε τέτοιο να εκτελεστεί η highScore. Άρα το πολύ n πράξεις για το πρώτο κομμάτι, ομοίως για το δεύτερο το πολύ n πράξεις. Άρα θα είναι $T(n) = n * n = n^2$ άρα $O(n^2)$

- των σημείων ενδιαφέροντος που απέχουν το πολύ d από δύο διαφορετικά σημεία.

Θα χρησιμοποιήσουμε τη δεύτερη nearest για κάθε δεδομένο σημείο, εδώ 2 φορές, και συνεπώς θα δημιουργηθούν 2 λίστες. Η πολυπλοκότητα της nearest είναι O(n). Έπειτα από τις δυο νέες λίστες θα πρέπει να βρούμε τα κοινά σημεία ενδιαφέροντος. Θα χρησιμοποιήσουμε την inCommonWith που έχει πολυπλοκότητα $O(n^2)$. Άρα συνολικά η πολυπλοκότητα θα είναι $2 *n + n^2$ άρα εν τέλει $O(n^2)$.