Πανεπιστήμιο Ιωαννίνων – Τμήμα Μηχανικών Η/Υ και Πληροφορικής
Δομές Δεδομένων [ΜΥΥ303] – Χειμερινό Εξάμηνο 2023

4η Εργαστηριακή Άσκηση
Ουρές Προτεραιότητας και Αλγόριθμος Dijkstra
Παράδοση έως Τετάρτη 29/11, 14:00 από το eCourse

ΠΡΟΣΟΧΗ: Γράψτε σε κάθε αρχείο που παραδίδετε τα ονόματα, τους Α.Μ. των μελών της ομάδας σας, καθώς και το group του εργαστηρίου σας. Συμπεριλάβετε όλα τα αρχεία σας (κώδικας Java και lab4results.txt) σε ένα zip αρχείο. Το όνομα που θα δώσετε στο συμπιεσμένο αρχείο θα αποτελείται από το group του εργαστηρίου στο οποίο έχετε τοποθετηθεί καθώς και από το ID της ομάδας σας (π.χ., G1_ID1.zip).

Συμπληρώστε το πρόγραμμα IndexMinPQ.java ώστε να υλοποιεί μια ουρά προτεραιότητας ελάχιστου με δυαδικό σωρό με *ευρετήριο*. Η δομή αποθηκεύει αντικείμενα τα οποία έχουν ένα κλειδί γενικού τύπου Key, καθώς και μια ακέραια ταυτότητα από 0 έως maxN − 1. Για το σκοπό αυτό χρησιμοποιούμε ένα πίνακα keys[] τύπου Key και δύο πίνακες ακεραίων pq[] και index[].

Το κλειδί ενός αντικειμένου με ταυτότητα j αποθηκεύεται στη θέση keys[j]. **Ο πίνακας pq[] αποθηκεύει τις ταυτότητες των αντικειμένων και είναι διατεταγμένος σε δυαδικό σωρό ως προς τα κλειδιά των αντικειμένων**. Η θέση του αντικειμένου με ταυτότητα j στον πίνακα pq[] δίνεται από την τιμή index[j], δηλαδή ισχύει

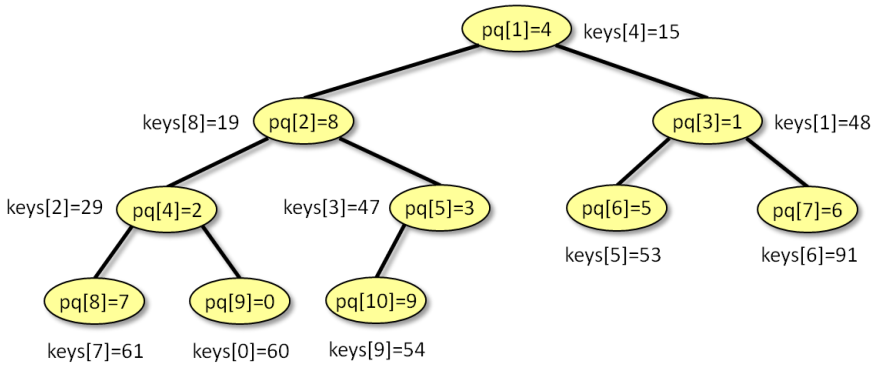
$$\text{pq}[\text{index}[j]] = \text{index}[\text{pq}[j]] = j.$$

Αν δεν έχει εισαχθεί στην ουρά προτεραιότητας το αντικείμενο με ταυτότητα j τότε έχουμε index[j] == −1.

Για παράδειγμα, μετά την εισαγωγή των αντικειμένων με ταυτότητες (με μαύρο χρώμα)και αντίστοιχα κλειδιά (με μπλε χρώμα)

$(0, \mathbf{60}),\ (1, \mathbf{48}),\ (2, \mathbf{29}),\ (3, \mathbf{47}),\ (4, \mathbf{15}),\ (5, \mathbf{53}),\ (6, \mathbf{91}),\ (7, \mathbf{61}),\ (8, \mathbf{19}),\ (9, \mathbf{54})$

έχουμε keys[0: 9] = [60,48,29,47,15,53,91,61,19,54], pq[1: 10] = [4,8,1,2,3,5,6,7,0,9] και index[0: 9] = [9,3,4,5,1,6,7,8,2,10], όπου η διάταξη σε δυαδικό σωρό απεικονίζεται στο ακόλουθο σχήμα.



Προσέξτε ότι στον πίνακα (δυαδικό σωρό) pq[] δε χρησιμοποιούμε τη θέση 0 για να απλοποιήσουμε τον τρόπο υπολογισμού του γονέα και των παιδιών κάθε θέσης. (Ο γονέας της θέσης k είναι ο $k/2$ και τα παιδιά της τα $2k$ και $2k + 1$.) Αντίθετα, στους πίνακες keys[] και index[], οι θέσεις keys[0] και index[0] δίνουν το κλειδί και τη θέση του αντικειμένου με ταυτότητα 0.

Το αρχείο IndexMinPQ.java περιλαμβάνει έτοιμες τις ακόλουθες μεθόδους.

IndexMinPQ(int maxN)	δημιουργεί μια ουρά προτεραιότητας για maxN αντικείμενα με ταυτότητες από 0 έως maxN − 1
boolean isEmpty()	ελέγχει αν η ουρά προτεραιότητας είναι κενή
boolean contains(int *j*)	ελέγχει αν η ουρά προτεραιότητας περιέχει το αντικείμενο με ταυτότητα *j*
Key getKey(int *j*)	επιστρέφει το κλειδί του αντικειμένου με ταυτότητα *j*

Συμπληρωματικές μέθοδοι

Συμπληρώστε τις παρακάτω μεθόδους στο πρόγραμμα IndexMinPQ.java.

void insert(int *j*, Key key)	εισάγει αντικείμενο με ταυτότητα *j* και κλειδί key
int delMin()	διαγράφει ένα αντικείμενο με ελάχιστο κλειδί και επιστρέφει την ακέραια ταυτότητα του
int minItem()	επιστρέφει την ακέραια ταυτότητα ενός αντικειμένου με ελάχιστο κλειδί
void change(int *j*, Key key)	αλλάζει το κλειδί του αντικειμένου με ταυτότητα *j* και του αναθέτει την τιμή key

Για την υλοποίηση των παραπάνω μεθόδων θα χρειαστεί να υλοποιήσετε βοηθητικές μεθόδους, όπως οι fixUp και fixDown, με κατάλληλες τροποποιήσεις έτσι ώστε να ενημερώνονται σωστά οι πίνακες pq[], index[] και keys[].

Εκτέλεση Προγράμματος

Η main() μέθοδος διαβάζει από την είσοδο τον αριθμό των αντικειμένων που θα εισαχθούν στην ουρά προτεραιότητας και αναθέτει σε κάθε αντικείμενο ένα τυχαίο ακέραιο κλειδί. Στη συνέχεια διαγράφεται διαδοχικά το αντικείμενο με το ελάχιστο κλειδί μέχρι να αδειάσει η ουρά προτεραιότητας. Εκτελέστε το πρόγραμμα IndexMinPQ.java για 20 αντικείμενα και αποθηκεύστε τα αποτελέσματα της εκτέλεσης στο αρχείο lab4results.txt. Για την εκτέλεση του προγράμματος γράψτε

java IndexMinPQ 20.

Τα αντικείμενα που επιστρέφει η delMin() θα πρέπει να εμφανίζονται **σε αύξουσα σειρά** ως προς τις τιμές των κλειδιών τους.

Αλγόριθμος Dijkstra

Το πρόγραμμα Dijkstra.java υλοποιεί τον αλγόριθμο του Dijkstra για τον υπολογισμό των ελαφρύτατων διαδρομών σε κατευθυνόμενο γράφημα (το οποίο υλοποιείται από τα αρχεία Collection.java, DirectedEdge.java και EdgeWeightedDigraph.java) με μη αρνητικά βάρη στις ακμές.

Ο αλγόριθμος χρησιμοποιεί μια ουρά προτεραιότητας όπου τα αντικείμενα είναι οι κόμβοι, αριθμημένοι με τους ακέραιους από 0 έως $N-1$, και κλειδιά οι υπολογισμένες αποστάσεις από έναν αφετηριακό κόμβο s. Το πρόγραμμα τυπώνει στην έξοδο την απόσταση από τον s προς τους κόμβους $N/2$ και $N-2$, καθώς και το χρόνο εκτέλεσης.

Η δοθείσα υλοποίηση χρησιμοποιεί μια απλοϊκή ουρά προτεραιότητας με μη ταξινομημένο πίνακα, η οποία δίνεται στο πρόγραμμα NaiveIndexMinPQ.java. Η ουρά αυτή έχει ακριβώς την ίδια διασύνδεση με τη IndexMinPQ αλλά διαφέρει ως προς την υλοποίηση. Χρησιμοποιεί τους ίδιους πίνακες pq[], index[] και keys[] που αναφέραμε παραπάνω, με τη σημαντική διαφορά ότι ο πίνακας pq[] δεν είναι διατεταγμένος σε σωρό, αλλά είναι απλά ένας μη ταξινομημένος πίνακας αντικειμένων. Κάθε εισαγωγή τοποθετεί το νέο στοιχείο στην επόμενη κενή θέση του πίνακα (σε $O(1)$ χρόνο), ενώ οι μέθοδοι minItem() και delMin() πρέπει να ψάξουν ολόκληρο τον πίνακα pq[] για να βρουν το αντικείμενο της ουράς προτεραιότητας με το ελάχιστο κλειδί (δηλαδή χρειάζονται $O(n)$ χρόνο).

Αντικαταστήστε την απλοϊκή ουρά προτεραιότητας στο αρχείο Dijkstra.java με αυτήν που υλοποιήσατε στο πρόγραμμα IndexMinPQ.java. Το μόνο που θα χρειαστεί να αλλάξετε είναι **η δήλωση της ουράς προτεραιότητας** που χρησιμοποιεί ο αλγόριθμος.

Η main() μέθοδος του Dijkstra.java διαβάζει ένα γράφημα από την είσοδο χρησιμοποιώντας το πρόγραμμα In.java. Αποθηκεύστε τα αποτελέσματα της εκτέλεσης του προγράμματος Dijkstra.java με είσοδο το δοκιμαστικό αρχείο exampleGraph, καθώς και με τα αρχεία Roma και NewYork, και αποθηκεύστε τα αποτελέσματα στο αρχείο lab4results.txt. Για την εκτέλεση του προγράμματος με είσοδο, π.χ. το αρχείο NewYork, γράψτε

java Dijkstra < NewYork.

Υπολογισμός της διαμέτρου ενός γραφήματος

Συμπληρώστε στο Dijkstra.java την ακόλουθη μέθοδο:

long diameter()	υπολογίζει τη διάμετρο του γραφήματος, δηλαδή τη μέγιστη απόσταση μεταξύ οποιωνδήποτε δύο κόμβων του γραφήματος

Για την υλοποίηση της παραπάνω μεθόδου, μπορείτε να κάνετε κλήσεις στη μέθοδο Dijkstra(G, s) του αρχείου Dijkstra.java, ή σε κάποια παραλλαγή της, για διαφορετικές αφετηριακές κορυφές s.

Χρησιμοποιήστε τη μέθοδό σας για να υπολογίσετε τη διάμετρο των γραφημάτων exampleGraph και Roma και αποθηκεύστε το αποτέλεσμα που βρήκατε στο αρχείο lab4results.txt. (Ο υπολογισμός της διαμέτρου του γραφήματος NewYork θα χρειαστεί πολλές ώρες!)

Παραδοτέα

Ανεβάστε στο eCourse ένα zip αρχείο με τα τελικά προγράμματα σας IndexMinPQ.java και Dijkstra.java, τα έτοιμα προγράμματα Collection.java, In.java, DirectedEdge.java, EdgeWeightedDigraph.java και NaiveIndexMinPQ.java καθώς και με το αρχείο των αποτελεσμάτων lab4results.txt.

Το zip αρχείο πρέπει να έχει όνομα που περιλαμβάνει το group του εργαστηρίου στο οποίο έχετε τοποθετηθεί καθώς και από το ID της ομάδας σας (π.χ., G1_ID1.zip).

Μη συμπεριλάβετε στο zip τα αρχεία εισόδου!