Tema 3 SD – grafuri neorientate, orientate şi ponderate

Autor: David Iancu

Deadline: 17 mai

Problema 1. Schimbare

Se dă un graf cu N noduri şi M muchii unidirecţionale. Se doreşte să se formeze un ciclu în graf, de lungime cel puţin 2, dar nu se ştie dacă graful iniţial are vreun ciclu. Se permite următoarea operaţie – se ia o muchie deja existenta şi i se schimba sensul – dacă aveam muchie de la nodul u la nodul v, putem schimba sensul astfel încât sa avem muchie de la nodul v la nodul u. Totuşi, schimbarea de sens are un cost asociat x. Se doreşte sa se obţină un ciclu cu un cost cat mai mic.

Input:
Se citeşte din fişierul “schimbare.in” N, M, apoi M linii de tip (u,v,x) semnificând ca avem muchie de la u la v, iar costul pentru schimbarea direcţiei în (v,u) este x.

Output:
Se afişează în fişierul “schimbare.out” costul cel mai mic care se poate obţine din schimbările de muchii astfel încât sa obţinem un ciclu. Dacă exista deja un ciclu în graf, se afişează 0. Dacă nu se poate obţine un ciclu indiferent cate muchii schimbăm, se afişează -1.

Restricţii:
1<=N<=1000
1<=M<=800
1<=x<=1000

Exemplu 1:

Input:

3 3
1 2 3
2 3 1
3 1 2

Output:
0

Explicaţie: avem deja un ciclu, 1-2-3-1

Exemplu 2:

Input:

3 3
1 2 3
2 3 1
1 3 2

Output:
2

Explicaţie: se schimbă muchia (1,3) în (3,1)

Problema 2. Reprezentanți.

Se da un graf cu N noduri şi M muchii bidirecţionale cu costuri asociate. Din cele N noduri, X noduri sunt mai speciale şi se numesc reprezentanți. Toate celelalte noduri trebuie să se conecteze la un reprezentant, astfel încât distanţa de la un nod pana la reprezentant să fie minimă. Să se găsească pentru fiecare nod reprezentantul cel mai apropiat.

Input: Se citeşte din fişierul “reprezentanti.in” N,M si X cu semnificaţia din enunţ, apoi M linii de tip (u,v,x) reprezentând un drum bidirecţional de la u la v cu costul x. Pe ultima linie se citesc cei X reprezentanți.

Output: Să se afişeze în fişierul “reprezentanti.out” N linii, reprezentând perechi cu reprezentantul şi distanţa minimă până la reprezentant. Dacă nu se poate ajunge la niciun reprezentant dintr-un nod, să se afişeze -1. Dacă se poate ajunge la doi reprezentanți în acelaşi timp, se ia reprezentantul cu indice minim.

Restricţii:
1<=N,X<= 10000
1<=M<=30000
1<=x<= 1000

Exemplu:

Input:
5 10 2
1 2 1
1 3 2
1 4 3
1 5 4
2 3 5
2 4 6
2 5 5
3 4 4
3 5 3
4 5 2
1 3

Output:
1 0
1 1
3 0
1 3
3 3

Explicație:

1 și 3 sunt reprezentanți, au costul asociat 0. La nodul 2 se ajunge din nodul 1 cu cost 1, la nodul 4 se ajunge din nodul 1 cu cost 3 iar la nodul 5 se ajunge din nodul 3 cu cost 3.

Problema 3. Drumuri.

Se da un graf cu N noduri şi M muchii fără costuri. Consideram ca lungimea drumului intre 2 noduri u şi v reprezintă numărul de muchii care se află între cele 2 noduri (cel puțin o muchie, dacă u şi v sunt conectate direct). Între 2 noduri putem avea mai multe drumuri de distanță minimă. Dintre aceste drumuri, ne interesează să aflăm nodurile care se află pe toate drumurile de lungime minimă între 2 noduri date.

Input:
Din fișierul “drumuri.in” se citește pe prima linie N si M şi Q, numărul de query-uri. Pe următoarele M linii se citesc perechi (u,v) reprezentând un drum bidirecțional de la u la v. Pe următoarele Q linii, se citesc perechi de tip (x,y), pentru care dorim să aflăm nodurile care se află pe toate drumurile de distanță minimă.

Output:
În fișierul “drumuri.out” se vor afișa Q linii, pentru fiecare query, în felul următor – numărul de noduri care se află pe toate drumurile de cost minim, apoi nodurile în ordine crescătoare. Dacă nu avem astfel de noduri, se afișează doar “0”.

Restricții:
1<=N<=10000
1<=M<=50000
1<=Q<=100

Exemplu:

Input:
5 6 2
1 3
1 4
1 5
2 4
3 4
3 5
1 4
2 5

Output:
2 1 4
3 2 4 5

Explicație:
Pentru primul query, avem un singur drum minim, şi anume $1 \rightarrow 4$, de lungime 1.
Pentru al doilea query, putem merge $2 \rightarrow 4 \rightarrow 3 \rightarrow 5$ sau $2 \rightarrow 4 \rightarrow 1 \rightarrow 5$, folosind de fiecare dată nodurile 2,4 şi 5.

Problema 4. Labirint.

Se dă o matrice NxM reprezentând un labirint. David încearcă sa iasă din labirint, dar ce nu ştie e ca de fapt nu exista ieşire. În labirint exista ziduri, notate cu 1, şi celule în care se poate merge, notate cu 0. Poziția lui David este codificată în labirint cu 2. Se ştie ca David a încercat sa exploreze labirintul, se cunosc cele K direcţii în care a mers, dar nu se ştie şi cat a mers în labirint pe fiecare direcție. Se ştie ca pentru fiecare direcție, David s-a deplasat cel puțin o căsuță (posibil şi mai multe). Să se afişeze în cate locuri posibile se poate afla în urma efectuării deplasărilor. Deplasările pot fi în sus (codificat cu 0), în jos (codificat cu 1), la stânga (codificat cu 2) sau la dreapta (codificat cu 3).

Input:
Se citesc din fişierul “labirint.in” N, M si K, apoi pe următoarele N linii se citesc cate M caractere codificând matricea (0 înseamnă drum, 1 zid, 2 înseamnă poziția lui David). Pe ultimul rand, se afişează cele K direcţii pe care a mers.

Output:
Sa se afişeze în fişierul “labirint.out” numărul de locuri posibile în care se poate afla David după cele K mişcări (0 dacă nu există o configurație validă de mişcări).

Restricţii:
1<=N,M<=100
1<=K<=1000

Exemplu:

Input:

4 5 3
0 0 0 0 0
0 1 0 1 0
1 0 0 0 0
0 1 1 0 2
0 2 1

Output:
4

Explicație:
David se afla în căsuța cu coordonatele (4,5). A mers în sus, la stânga, apoi în jos. La final, se poate afla în căsuțele (4,4), (3,3), (2,3) sau (2,1).

Problema 5. Bonus.

Se dă un graf cu N noduri şi M muchii bidirecţionale cu costuri. Se pune următoarea problemă – dându-se două noduri, care este drumul între cele două noduri care minimizează costul maxim al unei muchii? Altfel spus, cum putem merge pe un drum astfel încât costul maxim al unei muchii pe care o întâlnim să fie minim? Atenție, nu este vorba de costul total al drumului, ci de muchia cu cel mai mare cost – este de preferat un drum mai lung, cu cost total mai mare, dar pe care costul cel mai mare întâlnit este mai mic, de exemplu dacă avem de ales un drum care trece prin muchii de cost 2,5 şi 4 sau alt drum care trece prin muchii de cost 2,2,2,3,3, vom prefera al doilea drum, chiar dacă are cost total mai mare, dar are muchia cea mai mare 3 față de 5. Pentru că un simplu query ar fi prea simplu, se dau K query-uri de acest tip. Să se afişeze pentru fiecare query (pereche de 2 noduri) costul minim al muchiei celei mai mari de pe un drum dintre cele 2 noduri.

Input:
Din fişierul ”bonus.in” se citesc numerele N,M şi K, apoi pe următoarele M linii perechi de tip (u,v,x) cu semnificația că avem drum bidirecțional de la u la v de cost x. Pe următoarele K linii se dau query-uri de tip (a,b) pentru care vrem să afişăm drumurile optime.

Output:
În fişierul ”bonus.out” se afişează pe K linii răspunsul la cele K query-uri – costul minim cel mai mare al muchiei de pe un drum între nodurile a şi b. Se garantează că graful este conex.

Restricţii:
1<=N,K<=10000
1<=M<=50000
1<=x<=1000000

Exemplu:

Input:
4 5 2
1 2 10
1 3 5
1 4 3
2 3 2
2 4 6
1 2
2 3

Output:
5
2

Explicație:
Pentru primul exemplu, deşi avem muchie directă de cost 10, e mai optim să mergem de la nodul 1 la 3, apoi la nodul 2, muchia cea mai mică întâlnită având cost 5. Pentru al doilea exemplu, se merge direct pe muchia de la 2 la 3, de cost 2.

Explicații suplimentare temă:

Timp de rulare pentru fiecare problema – 1 secunda.
Limbaj permis: C.
Punctaj:
P1 – 2p
P2 – 2p
P3 – 2p
P4 – 3p
Bonus – 2p
Readme + coding style – 1p
Testele vor fi publice, exista un singur răspuns corect, astfel încât verificarea se face cu diff – fiți atenți la modul în care afișați.
Este nevoie de un makefile cu regulile clean, build p1, build p2, build p3, build p4, build bonus – executabilele se vor numi p1,p2, p3, p4 și bonus.