PA - TEMA 2 - GRAFURI

Responsabili:
Cristian Banu, Matei Teodorescu, Mihai Nan, Neculai Balaban, Radu Visan

Deadline soft: **18.05.2018**
Deadline hard: **25.05.2018**

CUPRINS

PROBLEMA 1: MINLEX BFS

Enunt

Se da un graf neorientat, conex cu N noduri si M muchii. In urma rularii algoritmului BFS, se obtine un sir de N numere care reprezinta parcurgerea BFS a grafului. Acest sir nu este neaparat unic, un graf putand avea una sau mai multe parcurgeri BFS valide. Sa se determine parcurgerea minim lexicografica.

Fie 2 siruri $A = (A_1, A_2, ..., A_n)$ si $B = (B_1, B_2, ..., B_n)$. Sirul A este mai mic lexicografic decat B daca exista o pozitie p pentru care $A_1 = B_1$, $A_2 = B_2$, ..., $A_{p-1} = B_{p-1}$ si $A_p < B_p$.

Date de intrare

Pe prima linie a fisierului **minlexbfs.in** se afla doua numere intregi N si M, reprezentand numarul de noduri, respectiv numarul de muchii ale grafului.

Pe urmatoarele M linii se afla cate doua numere intregi X si Y, cu semnificatia ca exista o muchie neorientata intre nodul X si nodul Y.

Date de iesire

In fisierul **minlexbfs.out** se va scrie parcurgerea BFS minim lexicografica, sub forma unui sir de N numere separate prin spatiu.

Restrictii si precizari

- $1 \leqslant N, M \leqslant 10^5$

Testare si punctare

- Punctajul maxim este de **50** puncte.
- Timpul de executie:
  - C/C++: **TODO s**
  - Java: **TODO s**
- Sursa care contine functia **main** trebuie obligatoriu denumita: **minlexbfs.c**, **minlexbfs.cpp** sau **Minlexbfs.java**.

Exemplu

minlexbfs.in	minlexbfs.out	Explicatie
5 5 1 3 1 5 3 2 3 5 5 4	1 3 5 2 4	Acest graf are 2 parcurgeri BFS valide: (1, 3, 5, 2, 4) (1, 5, 3, 4, 2) Prima parcurgere este si cea minim lexicografica.

PROBLEMA 2: DISJCNT

Enunt

Se da un graf neorientat conex cu N noduri si M muchii. Sa se determine numarul de perechi de noduri (X, Y) pentru care exista cel putin un mod de a alege 2 drumuri intre nodul X si nodul Y care sa nu aiba muchii comune.

Date de intrare

Pe prima linie a fisierului **disjcnt.in** se afla doua numere intregi N si M, reprezentand numarul de noduri, respectiv numarul de muchii ale grafului.

Pe urmatoarele M linii se afla cate doua numere intregi X si Y, cu semnificatia ca exista o muchie neorientata intre nodul X si nodul Y.

Date de iesire

In fisierul **disjcnt.out** va contine un singur numar, egal cu numarul de perechi de noduri care respecta proprietatea de mai sus.

Restrictii si precizari

- $1 \leqslant N, M \leqslant 10^5$
- Se vor numara doar perechile (X, Y) cu $X < Y$.

Testare si punctare

- Punctajul maxim este de **40** puncte.
- Timpul de executie:
  - C/C++: **TODO s**
  - Java: **TODO s**
- Sursa care contine functia **main** trebuie obligatoriu denumita: **disjcnt.c**, **disjcnt.cpp** sau **Disjcnt.java**.

Exemple

Exemplu 1

disjcnt.in	disjcnt.out	Explicatie
5 4 1 2 2 3 3 4 4 5	0	Raspunsul este 0 deoarece oricum am alege 2 noduri diferite, exista un singur drum intre ele.

Exemplu 2

disjcnt.in	disjcnt.out	Explicatie
5 6 1 2 2 3 3 1 3 4 4 5 5 4	4	Cele 4 perechi sunt $(1,2)$, $(1,3)$, $(2,3)$ si $(4,5)$.

PROBLEMA 3: RTD

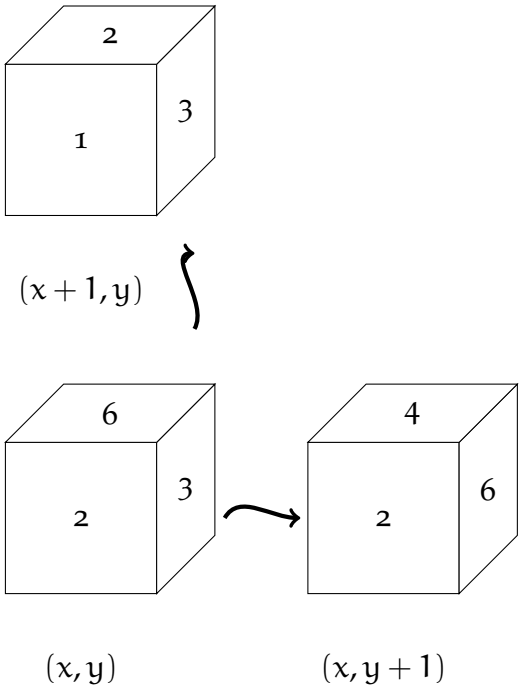
Enunt

Se da un grid de N x M si un zar cu costuri pe fiecare dintre cele 6 fete. Initial, zarul se afla pe pozitia (S_x, S_y) din grid, asezat ca in figura de mai jos, cu fata numerotata cu 1 in contact cu casuta de start a gridului. Putem sa mutam zarul in oricare dintre cele 4 directii (sus, jos, stanga, dreapta), iar o mutare consta in rostogolirea zarului in directia corespunzatoare.

Stiind ca exista casute blocate in matrice, se cere costul minim de a deplasa zarul din pozitia (S_x, S_y) in pozitia (F_x, F_y).

Costul deplasarii zarului intre 2 pozitii date este dat de suma costurilor fetelor pe care este asezat zarul pe parcursul deplasarii.

In figura de mai jos este ilustrat modul in care este asezat zarul la inceput, precum si configuratiile pe care le obtinem daca facem mutari in cele 2 directii.



Date de intrare

Pe prima linie a fisierului **rtd.in** se afla 7 numere intregi, N, M, S_x, S_y, F_x, F_y, K, reprezentand dimensiunile gridului, pozitia de start si pozitia de final a zarului, respectiv numarul de casute blocate din matrice.

Pe a doua linie se afla 6 numere intregi, al i-lea numar fiind egal cu costul de pe a i-a fata.

Pe urmatoarele K linii se afla cate 2 numere intregi X si Y, cu semnificatia ca pozitia (X, Y) este blocata.

Date de iesire

Fisierul **rtd.out** va contine un singur numar, egal cu costul minim de deplasare intre cele 2 pozitii date.

Restrictii si precizari

- $1 \leqslant N, M \leqslant 500$
- $1 \leqslant K \leqslant N * M$
- $1 \leqslant S_x, F_x \leqslant N$
- $1 \leqslant S_y, F_y \leqslant M$
- $1 \leqslant c \leqslant 100$, unde c este costul unei fete.
- Suma indicilor fetelor opuse este mereu 7.
- Se garanteaza ca pozitiile (S_x, S_y) si (F_x, F_y) nu sunt blocate.

Testare si punctare

- Punctajul maxim este de **30** puncte.
- Timpul de executie:
  - C/C++: **TODO s**
  - Java: **TODO s**
- Sursa care contine functia **main** trebuie obligatoriu denumita: **rtd.c**, **rtd.cpp** sau **Rtd.java**.

Exemplu

1 5 1 1 1 5 0 1 5 7 9 3 8	26	Trebuie sa rostogolim zarul de 4 ori la dreapta. Pe parcursul deplasarii, fetele pe care este asezat zarul sunt 1, 3, 6, 4, 1. Costul deplasarii este $26 = 1 + 7 + 8 + 9 + 1$.

BONUS: REVEDGES

Enunt

Se da un graf orientat cu N noduri si M muchii. Se dau Q query-uri, sub forma unor perechi de noduri (X, Y), iar pentru fiecare pereche se cere numarul minim de muchii carora trebuie sa le schimbam orientarea, astfel incat sa existe cel putin un drum de la X la Y.

Date de intrare

Pe prima linie a fisierului **revedges.in** se afla trei numere intregi, N, M si Q, reprezentand numarul de noduri din graf, numarul de muchii din graf, respectiv numarul de query-uri.

Pe urmatoarele M linii se afla cate doua numere intregi X si Y, cu semnificatia ca exista o muchie orientata de la nodul X la nodul Y.

Pe urmatoarele Q linii se afla cate doua numere intregi X si Y, reprezentand un query.

Date de iesire

Fisierul **revedges.out** va contine Q linii, pe linia i aflandu-se raspunsul pentru al i-lea query din input.

Restrictii si precizari

- $1 \leqslant N \leqslant 400$
- $1 \leqslant M \leqslant N * (N-1)/2$
- $1 \leqslant Q \leqslant 10^6$
- Cele Q query-uri sunt independente, iar modificarile facute asupra grafului la un query nu persista si la urmatorul query.

Testare si punctare

- Punctajul maxim este de **25** puncte.
- Timpul de executie:
  - C/C++: **TODO s**
  - Java: **TODO s**
- Sursa care contine functia **main** trebuie obligatoriu denumita: **revedges.c**, **revedges.cpp** sau **Revedges.java**.

Exemplu

revedges.in	revedges.out	Explicatie
4 3 3 1 2 1 3 3 4 1 3 3 2 4 1	0 1 2	Pentru query-ul $(1, 3)$ avem drum direct de la nodul 1 la nodul 3. Pentru query-ul $(3, 2)$ este suficient sa inversam sensul muchiei $(3, 1)$. Pentru query-ul $(4, 1)$ trebuie sa inversam sensul muchiilor $(4, 3)$ si $(3, 1)$.

PUNCTARE

- Punctajul temei este de 125 puncte, distribuit astfel:
  - Problema 1: 50p
  - Problema 2: 40p
  - Problema 3: 30p
  - 5 puncte vor fi acordate pentru coding style si README

Punctajul pe README, comentarii și coding style este condiționat de obținerea a unui punctaj strict pozitiv pe cel puțin un test.

Se poate obține un bonus de 25p rezolvând problema Revedges. Acordarea bonusului **NU** este condiționată de rezolvarea celorlate probleme. În total se pot obține 150 de puncte (**NU** se trunchiază).

Pentru detalii puteți să vă uitați și peste regulile generale de trimitere a temelor.

- O temă care **NU** compilează va fi punctată cu 0.
- O temă care **NU** trece niciun test pe vmchecker va fi punctată cu 0.
- Vor exista mai multe teste pentru fiecare problemă în parte. Punctele pe teste sunt independente, punctajul pe un anumit test nefiind condiționat de alte teste.
- Fiecare problemă va avea o limită de timp pe test (precizată mai jos și pe pagina cu enunțul). Dacă execuția programului pe un test al acelei probleme va dura mai mult decât limita de timp, veți primi automat o puncte pe testul respectiv și execuția va fi întreruptă.
- În fișierul README va trebui să descrieți soluția pe care ați ales-o pentru fiecare problemă, să precizați complexitatea pentru fiecare și alte lucruri pe care le considerați utile de menționat.

Checker

- Arhiva se va trimite pe vmchecker, unde tema se va testa folosind un set de teste private.
- Pentru testarea locala, aveți disponibil un set de teste publice (de aceeași dificultate) pe pagina cu resurse a temei.
- **Punctajul pe teste** este cel de pe vmchecker și se acordă ruland tema doar cu testele private.
- Checkerul verifică doar existența unui README cu denumire corectă și conținut nenul. **Punctajul final pe README și comentarii** se acordă la corectarea manuală a temei.
- La corectare se poate depuncta pentru **erori de coding style** care nu sunt semnalate de checker.
- Corectorii își rezervă dreptul de a scădea puncte pentru orice problemă găsită în implementare, dacă vor considera acest lucru necesar.

FORMAT ARHIVĂ

- Temele pot fi testate automat pe vmchecker. Acesta suportă temele rezolvate în C/C++ și Java. Dacă doriți să realizați tema în alt limbaj, trebuie să-i trimiteți un email lui Traian Rebedea (traian.rebedea@cs.pub.ro), în care să îi cereți explicit acest lucru.
- Arhiva cu rezolvarea temei trebuie să fie **.zip**, având un nume de forma **Grupa_NumePrenume_Tema1.zip** (ex: 399CX_PuiuGigel_Tema1.zip sau 399CX_BucurGigel_Tema1.zip) și va conține:

  - Fișierul/fișierele sursă
  - Fișierul **Makefile**
  - Fișierul **README** (fără extensie)

- Fișierul pentru make trebuie denumit obligatoriu **Makefile** și trebuie să conțină următoarele reguli:

  - **build**, care va compila sursele și va obține executabilele
  - **run-p1**, care va rula executabilul pentru problema 1
  - **run-p2**, care va rula executabilul pentru problema 2
  - **run-p3**, care va rula executabilul pentru problema 3
  - **clean**, care va șterge executabilele generate
  - **run-p4**, care va rula executabilul pentru problema bonus (**doar dacă** ați implementat și bonusul)

- **ATENȚIE!** Funcția **main** din rezolvarea unei probleme se va găsi într-o sursă ce trebuie obligatoriu denumită astfel:

  - **minlexbfs.c**, **minlexbfs.cpp** sau **Minlexbfs.java** - pentru problema 1
  - **disjcnt.c**, **disjcnt.cpp** sau **Disjcnt.java** - pentru problema 2
  - **rtd.c**, **rtd.cpp** sau **Rtd.java** - pentru problema 3
  - **revedges.c**, **revedges.cpp** sau **Revedges.java** - pentru problema 4

- **ATENȚIE!** Tema va fi compilată și testată **DOAR pe Linux**.
- **ATENȚIE!** Numele regulilor și a surselor trebuie să fie exact cele de mai sus. Absența sau denumirea diferită a acestora va avea drept consecință obținerea a 0 puncte pe testele asociate problemei rezolvate de regula respectivă.
- **ATENȚIE!** Pentru cei ce folosesc C/C++ **NU** este permisă compilarea cu opțiuni de optimizare a codului (O1, O2, etc.).
- **ATENȚIE!** Orice nerespectare a restricțiilor duce la pierderea punctajului (după regulile de mai sus).

LINKS

- Regulament general teme PA
- Google C++ Style Guide
- Google Java Style Guide
- Debugging și Structuri de Date