PA - TEMA 2 - GRAFURI

Responsabili:
Cristian Banu, Teodor Popescu, Radu Visan

Deadline soft: **19.05.2019, ora 23:59**
Deadline hard: **24.05.2019, ora 23:59**

CUPRINS

PROBLEMA 1

Enunt

Se da un vector cu N numere intregi $v_1, v_2, ..., v_N$. Se cere construirea unui graf neorientat cu N noduri pentru care, daca pornim o parcurgere in latime din nodul 1, vectorul de distante obtinut coincide cu v.

ATENTIE! Vor fi acceptate doar solutiile care folosesc cel mult 10^6 muchii.

Date de intrare

Pe prima linie a fisierului **p1.in** se afla un numar intreg N.

Pe urmatoarea linie se afla N numere intregi $v_1, v_2, v_3, ..., v_N$ reprezentand vectorul de distante dorit.

Date de iesire

Fisierul **p1.out** va contine pe prima linie o valoare intreaga M, reprezentand numarul de muchii din graf.

Urmatoarele M linii vor avea cate doua numere intregi, separate printr-un singur spatiu, care reprezinta capetele muchiilor.

Restrictii si precizari

- $1 \leqslant N \leqslant 10^5$
- $0 \leqslant v_1, v_2, v_3, ..., v_N \leqslant N$
- E garantat ca $v_1 = 0$.
- Nodurile sunt numerotate cu numere intregi intre 1 si N.
- Daca nu exista un graf pentru care sa se obtina vectorul respectiv de distante, se va afisa -1.

Testare si punctare

- Punctajul maxim este de **25** puncte.
- Timpul de execuție:
  - C/C++: **0.6 s**
  - Java, Python: **1.2 s**
- Sursa care contine functia **main** trebuie obligatoriu denumita: **p1.c**, **p1.cpp** sau **P1.java**.

Exemple

Exemplu 1

Exemplu 1		
p1.in	p1.out	Explicatie
4 0 1 1 2	3 1 2 1 3 3 4	Pentru a ajunge din nodul 1 in nodul 2 avem muchie directa 1-2. Pentru a ajunge din nodul 1 in nodul 3 avem muchie directa 1-3. Pentru a ajunge din nodul 1 in nodul 4 avem lantul 1-3-4.

PROBLEMA 2

Enunt

Se da o matrice de dimensiuni NxM cu valori intregi. Se cere aria celei mai mari zone de casute conectate pentru care diferenta dintre valoarea maxima si valoarea minima din acea zona este cel mult K.

Date de intrare

Pe prima linie a fisierului **p2.in** se afla trei numere intregi, N, M, respectiv K.

Pe urmatoarele N linii se vor afla cate M numere separate prin spatii, unde al j-lea numar de pe a i-a linie reprezinta $c_{i,j}$, valoarea aflata pe pozitia (i, j) in matrice.

Date de iesire

In fisierul **p2.out** se va scrie aria maxima a zonei gasite.

Restrictii si precizari

- $1 \leqslant N, M \leqslant 100$
- $0 \leqslant K \leqslant 10^9$
- $0 \leqslant c_{i,j} \leqslant 10^9$
- Definim aria unei zone de casute conectate ca fiind numarul de casute care fac parte din zona respectiva.
- Doua casute sunt conectate daca sunt vecine pe verticala sau orizontala.

Testare si punctare

- Punctajul maxim este de **35** puncte.
- Timpul de execuție:
  - C/C++: **0.6 s**
  - Java, Python: **1.2 s**
- Sursa care contine functia **main** trebuie obligatoriu denumita: **p2.c**, **p2.cpp** sau **P2.java**.

Exemple

Exemplu 1

Exemplu 1		
p2.in	p2.out	Explicatie
3 3 2 1 2 4 1 1 5 0 10 5	5	Valorile din zonele pe care le putem gasi sunt: 1. 0, 1, 1, 1, 2 2. 2, 4 3. 4, 5, 5 4. 10 Se observa ca cea mai mare zona contine 5 casute.

PROBLEMA 3

Enunt

Se da un graf cu N noduri si M muchii bidirectionale, unde a i-a muchie are un cost c_i si un tip t_i, cu $1 \leqslant t_i \leqslant T$. Dorim sa gasim un drum de cost minim, care porneste din nodul 1 si se termina in nodul N, unde costul unui drum este definit ca suma costurilor muchiilor individuale de pe drum, la care se adauga o penalizare definita astfel: daca drumul nostru contine X muchii, iar tipurile muchiilor sunt $t_1, t_2, ..., t_X$, atunci penalizarea este egala cu $\sum_{i=1}^{X-1} p_{t_i,t_{i+1}}$.

Date de intrare

Pe prima linie a fisierului **p3.in** se afla trei numere intregi N, M, respectiv T.

Pe urmatoarele M linii se vor afla cate patru numere intregi, u, v, c, respectiv t, reprezentand o muchie bidirectionala intre nodul u si nodul v, cu cost c si de tip t.

Pe urmatoarele T linii se vor afla cate T numere intregi, unde al j-lea numar de pe linia i reprezinta $p_{i,j}$, si anume penalizarea care se adauga unui drum care contine doua muchii adiacente, prima de tip i si a doua de tip j.

Date de iesire

In fisierul **p3.out** se va scrie costul minim al unui drum de la nodul 1 la nodul N.

Restrictii si precizari

- $1 \leqslant N \leqslant 200$
- $1 \leqslant M \leqslant N * (N-1)/2$
- $1 \leqslant T \leqslant 100$
- $1 \leqslant u_i, v_i \leqslant N$, $1 \leqslant i \leqslant M$
- $1 \leqslant c_i \leqslant 10^9$, $1 \leqslant i \leqslant M$
- $1 \leqslant t_i \leqslant T$, $1 \leqslant i \leqslant M$
- $0 \leqslant p_{i,j} \leqslant 10^9$
- $p_{i,j} = p_{j,i}$, $\forall i, j$ cu $i \neq j$
- $p_{i,i} = 0$, $1 \leqslant i \leqslant T$
- In cazul in care un astfel de drum nu exista, se va afisa -1.

Testare si punctare

- Punctajul maxim este de **40** puncte.
- Timpul de execuție:

  - C/C++: **0.8 s**
  - Java, Python: **1.6 s**
- Sursa care contine functia **main** trebuie obligatoriu denumita: **p3.c**, **p3.cpp** sau **P3.java**.

Exemple

Exemplu 1

Exemplu 1		
p3.in	p3.out	Explicatie
4 4 2 1 2 4 1 2 3 3 2 1 3 1 1 3 4 3 1 0 2 2 0	4	Avem 2 drumuri posibile intre 1 si 4 si anume: 1. 1 - 2 - 3 - 4: in acest caz, costul este 4 (costul muchiei 1 - 2) + 3 (costul muchiei 2 - 3) + 3 (costul muchiei 3 - 4) + 2 (penalizarea data de faptul ca prima muchie e de tip 1, iar a doua e de tip 2) + 2 (penalizarea data de faptul ca a doua muchie e de tip 2, iar a treia e de tip 1) = 14. 2. 1 - 3 - 4: in acest caz, costul este 1 (costul muchiei 1 - 3) + 3 (costul muchiei 3 - 4) + 0 (penalizarea nu exista deoarece ambele muchii sunt de tipul 1) = 4.

BONUS

Enunt

Se da un graf cu N noduri si M muchii bidirectionale, unde a i-a muchie are un cost c_i si un tip t_i, cu $1 \leqslant t_i \leqslant T$. Dorim sa gasim un drum de cost minim, care porneste din nodul 1 si se termina in nodul N, unde costul unui drum este definit ca suma costurilor muchiilor individuale de pe drum, la care se adauga o penalizare corespunzatoare fiecarui tip de muchie pe care l-am folosit cel putin o data pe drum. Daca pe drum avem muchii de tipurile $t_1, t_2, ..., t_K$, distincte intre ele, atunci penalizarea care se adauga este $\sum_{i=1}^{K} p_{t_i}$.

Date de intrare

Pe prima linie a fisierului **p4.in** se afla trei numere intregi N, M, respectiv T.

Pe urmatoarele M linii se vor afla cate patru numere intregi, u, v, c, respectiv t, reprezentand o muchie bidirectionala intre nodul u si nodul v, cu cost c si de tip t.

Pe urmatoarea linie se afla T numere intregi, $p_1, p_2, ..., p_T$, unde p_i este penalizarea care se adauga daca pe drumul dintre 1 si N se gaseste cel putin o muchie de tip i.

Date de iesire

In fisierul **p4.out** se va scrie costul minim al unui drum de la nodul 1 la nodul N.

Restrictii si precizari

- $1 \leqslant N \leqslant 200$
- $1 \leqslant M \leqslant N * (N-1)/2$
- $1 \leqslant T \leqslant 7$
- $1 \leqslant u_i, v_i \leqslant N$, $1 \leqslant i \leqslant M$
- $1 \leqslant c_i \leqslant 10^9$, $1 \leqslant i \leqslant M$
- $1 \leqslant t_i \leqslant T$, $1 \leqslant i \leqslant M$
- $1 \leqslant p_i \leqslant 10^9$
- In cazul in care un astfel de drum nu exista, se va afisa -1.

Testare si punctare

- Punctajul maxim este de **20** puncte.
- Timpul de execuție:
  - C/C++: **1.4 s**
  - Java, Python: **2.8 s**

- Sursa care conține funcția **main** trebuie obligatoriu denumită: **p4.c**, **p4.cpp** sau **p4.java**.

Exemple

Exemplu 1

Exemplu 1		
p4.in	p4.out	Explicatie
4 4 2 1 2 4 1 2 3 3 2 1 3 1 1 3 4 3 1 4 7	8	Avem 2 drumuri posibile intre 1 si 4 si anume: 1. 1 - 2 - 3 - 4: in acest caz, costul este 4 (costul muchiei 1 - 2) + 3 (costul muchiei 2 - 3) + 3 (costul muchiei 3 - 4) + 4 (penalizarea data de faptul ca avem cel putin o muchie de tip 1; cu toate ca sunt doua muchii de tip 1, penalizarea se adauga o singura data) + 7 (penalizarea data de faptul ca avem cel putin o muchie de tip 2) = 21. 2. 1 - 3 - 4: in acest caz, costul este 1 (costul muchiei 1 - 3) + 3 (costul muchiei 3 - 4) + 4 (penalizarea data de faptul ca avem cel putin o muchie de tip 1) = 8.

PUNCTARE

- Punctajul temei este de 130 de puncte, distribuit astfel:
  - Problema 1: 25p
  - Problema 2: 35p
  - Problema 3: 40p
  - Bonus: 20p
  - 5 puncte vor fi acordate pentru comentarii si README
  - 5 puncte vor fi acordate pentru coding style

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 20p rezolvând problema bonus. Acordarea bonusului **NU** este condiționată de rezolvarea celorlate probleme. În total se pot obține 130 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.
- Punctajul pe teste este cel de pe vmchecker și se acordă ruland tema pe același set de teste pus la dispozitie pe Moodle.
- 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** 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:
  - **p1.c**, **p1.cpp** sau **P1.java** - pentru problema 1
  - **p2.c**, **p2.cpp** sau **P2.java** - pentru problema 2
  - **p3.c**, **p3.cpp** sau **P3.java** - pentru problema 3
  - **p4.c**, **p4.cpp** sau **P4.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