Laborator 11

Exerciții

1. **Roy-Floyd**[1]

Se dă un graf orientat ponderat cu `N` noduri și `M` arce – în care fiecare arc are asociat un cost, număr natural strict pozitiv. Folosind algoritmul *Roy-Floyd*, construiți matricea costurilor minime, $a_{i,j}$ fiind costul minim al unui drum de la `i` la `j`, dacă există un asemenea drum, sau `-1` în caz contrar.

Date de intrare

Fișierul de intrare roy-floyd.in conține pe prima linie numerele `N M`, iar următoarele linii câte un triplet `i j c`, cu semnificația: există arcul `(i,j)` și are costul `c`.

Date de ieșire

Fișierul de ieșire roy-floyd.out va conține matricea construită, câte o linie a matricei pe o linie a fișierului, elementele de pe fiecare linie fiind separate prin exact un spațiu.

Restricții

$1 \le n \le 100$.

Costul unui arc va fi mai mic decât 1000.

2. **Traseu**[2]

Sătul de atâtea olimpiade şi concursuri, Algorel a plecat la munte, să se relaxeze. Odată ajuns acolo s-a hotărât să urmeze un traseu cu peisaje cât mai frumoase. El are la dispoziție o hartă codificată sub forma unei matrice cu `M` linii şi `N` coloane, ale cărei elemente sunt numere naturale nenule reprezentând gradul de frumusețe corespunzător fiecărei zone (element al matricei).

Algorel trebuie să plece din colțul stânga sus al matricei (poziția `(1,1)`) şi să ajungă în colțul dreapta jos (poziția `(M,N)`), având voie să se deplaseze doar spre dreapta şi în jos (de pe poziția curentă `(i,j)` se poate deplasa fie pe poziția `(i+1,j)`, fie pe poziția `(i,j+1)`).

Definim gradul de frumusețe al unui traseu ca fiind suma gradelor de frumusețe ale elementelor matricei care-l compun. Se cere găsirea gradului maxim de frumusețe pe care îl poate avea un traseu care respectă restricțiile de mai sus.

[1] https://www.pbinfo.ro/probleme/589/roy-floyd
[2] http://varena.ro/problema/traseu

Date de intrare

Fişierul de intrare traseu.in conţine pe prima linie două numere naturale nenule `M` şi `N`. Pe următoarele `M` linii se află câte `N` numere naturale nenule, separate prin câte un spaţiu, reprezentând elementele matricei `A`, care codifică harta.

Date de ieşire

În fişierul de ieşire traseu.out se va scrie un singur număr `S` reprezentând gradul maxim de frumuseţe pe care îl poate avea un traseu care respectă restricţiile de mai sus. Gradul de frumuseţe al unui traseu este suma gradelor zonelor din matrice care îl compun.

Restricţii

$1 \le M \le 500$, $1 \le N \le 500$, $1 \le A_{i,j} \le 1000$.

3. **Roy-Warshall**[3]

Se dă lista arcelor unui graf orientat. Construiți matricea drumurilor, folosind algoritmul lui *Roy-Warshall*.

Date de intrare

Programul citește de la tastatură numărul `N` de noduri și numărul `M` de arce, iar apoi lista arcelor, formată din `M` perechi de forma `i j`, cu semnificația că există arc orientat de la `i` la `j`.

Date de ieșire

Programul va afișa pe ecran matricea drumurilor, câte o linie a matricei pe o linie a ecranului, elementele aceleiași linii fiind separate prin exact un spațiu.

Restricții

$1 \le n \le 100$.

4. **Firma**[4]

Într-o țară sunt `N` orașe, numerotate de la `1` la `N`, unite între ele prin `M` șosele bidirecționale de lungimi cunoscute, între oricare două orașe existând drum, fie șosea directă, fie prin alte orașe. O firmă dorește să-și stabilească sediul în unul dintre orașe, astfel încât suma lungimilor drumurilor minime de la orașul în care se află sediul la toate celelaltele orașe să fie minimă. Determinați orașul care va fi ales pentru sediul firmei. Dacă sunt mai multe orașe care pot fi alese, se va alege cel cu numărul de ordine mai mic.

Date de intrare

Fişierul de intrare firma.in conţine pe prima linie numerele `N M`, iar următoarele `M` linii câte trei numere `i j L`, cu semnificatia: între oraşele `i` şi `j` există o şosea directă de lungime `L`.

[3] https://www.pbinfo.ro/probleme/580/roy-warshall
[4] https://www.pbinfo.ro/probleme/591/firma

Date de ieșire

Fișierul de ieșire firma.out va conține pe prima linie numărul X, reprezentând numărul de ordine al orașului care va ales ca sediu pentru firmă.

Restricții

$1 \le n \le 100$.

5. **Dragoni**[5]

Într-o lume în care vikingii pot zbura cu dragonii există N insule. Hiccup, șeful tribului de vikingi aflat pe insula 1, știe M rute directe de zbor bidirecționale între insule. Pentru fiecare j intre 1 si M, ruta j unește insulele A_j și B_j și are lungime D_j.

Pe fiecare insulă i, ($1 \le i \le n$) există dragoni din specia i care pot zbura fără a se opri pentru odihnă o distanță maximă $Dmax_i$. Cu alte cuvinte, dragonii de pe insula i vor putea parcurge orice rută j, ($1 \le j \le m$) pentru care $D_j \le Dmax_i$, indiferent de ce alte drumuri au făcut anterior.

Hiccup dorește să ajungă de pe insula 1 pe insula N pentru a-l salva pe Toothless, dragonul lui. Pentru a ajunge acolo, el va lua inițial un dragon din specia 1 (de pe insula 1). Apoi, dacă la un moment dat Hiccup se află pe o insula i, ($1 \le i \le n$) având cu el un dragon din specia t, el poate:

- Să zboare de pe insula i pe o altă insulă x cu dragonul pe care îl are, folosind o rută directă j între insulele i si x, bineînțeles doar dacă $D_j \le Dmax_t$.
- Să schimbe dragonul din specia t pe care îl are cu un dragon din specia i aflat pe insula respectivă.

1. Să se determine distanța maxima $Dmax_i$ caracteristică unui dragon la care Hiccup poate ajunge fără a schimba dragonul pe care l-a luat inițial de pe insula 1.
2. Să se determine distanța minimă pe care Hiccup trebuie să o parcurgă pentru a ajunge de pe insula 1 pe insula N.

Date de intrare

Fișierul de intrare dragoni.in conține pe prima linie numărul p. Pentru toate testele de intrare, numărul p poate avea doar valoarea 1 sau valoarea 2. Pe a doua linie se găsesc două numere naturale N și M reprezentând numărul de insule, respectiv numărul de rute directe între insule. Pe a treia linie se găsesc N numere naturale, al i-lea dintre acestea reprezentând distanta maximă $Dmax_i$ pe care o poate zbura un dragon de pe insula i. Pe următoarele M linii sunt descrise cele M rute directe. Pe fiecare dintre aceste linii se găsesc câte trei numere naturale A, B și D cu semnificația că există rută bidirecțională de lungime D între insulele A și B.

Date de ieșire

[5] https://www.pbinfo.ro/probleme/1136/dragoni

În fișierul de ieșire dragoni.out se va afișa un singur număr natural.

Dacă valoarea lui `p` este `1`, se rezolvă numai cerința a.

În acest caz numărul afișat va reprezenta distanța maximă $Dmax_i$ a unui dragon `i` la care Hiccup poate ajunge fără a schimba dragonul pe care l-a luat inițial de pe insula `1`.

Daca valoarea lui `p` este `2`, se va rezolva numai cerința b.

În acest caz numărul afișat va reprezenta distanța minima pe care Hiccup trebuie să o parcurgă pentru a ajunge de pe insula `1` pe insula `N`.

Restricții

$1 \le N \le 800$, $1 \le M \le 6000$, $1 \le D\max_i \le 100$ pentru orice $1 \le i \le N$.

$1 \le A_j, B_j \le N$, pentru orice $1 \le j \le M$; $1 \le D_j \le 50.000$ pentru orice $1 \le j \le M$.

Se garantează că Hiccup poate ajunge pe insula N.

Se garantează că răspunsul oricărei cerințe este un număr natural mai mic decât 109.

6. **Biscuit**[6]

Jocul Biscuit se desfăşoară pe un caroiaj dreptunghiular cu `M` linii şi `N` coloane de puncte. Pe rând, doi jucători trasează segmente între două puncte vecine pe orizontală sau pe verticală. Jucătorul care închide un pătrat (de latură `1`) primeşte un punct şi mai mută o dată. Când toate segmentele au fost trasate, câştigă jucătorul care a închis mai multe pătrate.

Dându-se o tablă pe care s-au făcut deja nişte mutări, să se determine câte pătrate poate închide jucătorul care este la mutare.

Date de intrare

Fişierul de intrare biscuit.in conține, pe prima linie, numerele `M` şi `N`.

Pe următoarele `M` linii sunt descrise segmentele orizontale. Fiecare linie conține `N-1` caractere. Al `j`-lea caracter de pe linia `i` este `1` sau `0` după cum al `j`-lea segment orizontal de pe linia `i` a fost trasat sau nu.

Pe ultimele `M-1` linii sunt descrise segmentele verticale. Fiecare linie conține `N` caractere. Al `j`-lea caracter de pe linia `i` este `1` sau `0` după cum al `j`-lea segment vertical de pe linia `i` a fost trasat sau nu.

Date de ieşire

În fişierul de ieşire biscuit.out se va scrie numărul de pătrate pe care le poate închide jucătorul la mutare.

Restricții

$2 \le M, N \le 1.000$.

[6] http://varena.ro/problema/biscuit

7. **Bellman-Ford**[7]

Se dă un graf orientat conex cu `N` noduri şi `M` muchii cu costuri. Definim un lanț ca fiind un şir de noduri cu proprietatea că între oricare două consecutive există o muchie. Costul unui lanț este dat de suma costurilor muchiilor care unesc nodurile ce îl formează. Definim un ciclu ca fiind un lanț cu proprietatea că primul element al său este egal cu ultimul.

Să se determine dacă în graful dat există un ciclu de cost negativ. Dacă nu există, să se determine costul minim al unui lanț de la nodul `1` la fiecare dintre nodurile `2, 3, ..., N-1, N`.

Date de intrare

Fişierul de intrare bellmanford.in conține pe prima linie numerele `N` şi `M` cu semnificația din enunț. Pe următoarele `M` linii se vor afla câte `3` numere `x, y` şi `c` cu semnificația că există o muchie de cost `c` de la nodul `x` la nodul `y`.

Date de ieşire

În fişierul de ieşire bellmanford.out se va afişa pe prima linie mesajul "`Ciclu negativ!`" dacă în graf există un astfel de ciclu sau, în caz contrar, `N-1` numere separate printr-un spațiu. Al `i`-lea număr va reprezenta costul minim al unui lanț de la nodul `1` la nodul `i+1`.

Restricții

$1 \le N \le 50.000$, $1 \le M \le 250.000$

Costurile muchiilor sunt numere întregi cel mult egale în modul cu `1000`.

8. **Ciclu**[8]

Acarie, fericit că a trecut examenul la algebră pleacă în oraş să sărbătorească. Este atât de fericit, încât nu se mulțumeşte cu un simplu traseu între terase, ci vrea un ciclu pentru a fi sigur că poate continua să petreacă cât timp doreşte. Din păcate, drumurile dintre terase au costuri diferite şi nu sunt bidirecționale. Acarie nu este interesat neapărat să minimizeze costul acestui ciclu, ci doreşte să minimizeze costul mediu al ciclului (evident, vrea să meargă cât mai puțin între terase, adică costul ciclului împărțit la numărul de terase să fie minim).

Scrieți un program care determină costul mediu minim al unui ciclu de terase.

Date de intrare

Din fişierul ciclu.in se citesc `N` (numărul de terase), `M` (numărul de drumuri unidirecționale) şi cele `M` drumuri directe dintre terase. Acestea sunt date fiecare pe câte o linie, reprezentate de trei întregi - terasa de unde porneşte drumul, terasa unde ajunge drumul (numere între `1` şi `N`) şi costul drumului (întreg strict pozitiv mai mic decât `100.000`)

[7] https://infoarena.ro/problema/bellmanford
[8] https://infoarena.ro/problema/ciclu

Date de ieşire

Să se afişeze în fişierul ciclu.out costul mediu minim al ciclului găsit, cu o precizie de `2` zecimale.

Restricții

$2 \le N \le 1.000$, $2 \le M \le 4.000$

Se garantează că există cel puțin un ciclu.

Rezultatul se va verifica cu o precizie de `0.1`

9. **RF**[9]

Se dă un graf orientat în care arcele au asociate costuri (numere naturale nenule). Să se determine câte arce `(x,y)` din graf au costul egal cu costul drumului de cost minim de la `x` la `y`.

Date de intrare

Programul citește de la tastatură numerele `N M`, reprezentând numărul de vârfuri şi numărul de arce din graf, apoi `M` triplete `i j p`, reprezentând arcele, date prin extremități şi cost.

Date de ieșire

Programul va afișa pe ecran numărul `C`, reprezentând valoare cerută.

Restricții

$1 \le N \le 100$

Costurile arcelor vor fi mai mici decât `1000`.

10. **Retea1**[10]

Pentru a testa o nouă topologie s-a construit o rețea de calculatoare în care fiecare calculator transmite informația unidirecțional către un singur calculator din rețea. Numim conexiune o pereche ordonată de calculatoare, nu neapărat distincte, în care primul este cel care trimite informația iar al doilea este cel care o recepționează direct. Fiind dată o astfel de rețea şi conexiunile existente între calculatoarele care o alcătuiesc, să se determine submulțimea cu număr maxim de *calculatoare-feed-back*. Un *calculator-feed-back* are proprietatea că informația ce pleacă de la acesta ajunge, prin intermediul conexiunilor succesive, înapoi la calculatorul de la care a plecat.

Să se realizeze un program care, pentru o rețea cu `N` calculatoare numerotate de la `1` la `N` şi conexiuni precizate, determină submulțimea cu număr maxim de *calculatoare-feed-back*.

Date de intrare

Pe prima linie a fişierului de intrare retea1.in se află un număr natural nenul `N`, reprezentând numărul de calculatoare din rețea, iar pe fiecare dinre următoarele `N` linii, separate prin câte un spațiu, câte `N-1` valori `0` şi o valoare `1`, cu semnificația: pe linia `i`, elementul de pe poziția `j` este `1` dacă şi numai dacă există conexine între calculatorul `i` şi calculatorul `j`.

[9] https://www.pbinfo.ro/probleme/1652/rf
[10] https://www.pbinfo.ro/probleme/2232/retea1

Date de ieșire

În fişierul de ieşire retea1.out se vor scrie pe prima linie *calculatoarele-feed-back* din submulțimea determinată. Elementele submulțimii sunt scrise în ordine crescătoare, separate prin câte o virgulă, fără spații iar submulțimea începe cu caracterul { şi se termină cu caracterul }. În fişierul de ieşire nu se vor scrie spații înainte, între sau după elementele mulțimii.

Restricții

$1 \leq N \leq 300$

Un calculator poate să aibă o conexiune cu el însuşi.

Un calculator poate avea o conexiune doar cu singur calculator.