Ministerul Educației Na□ionale – ISJ Timiş
Olimpiada Na□ională de Informatică - Etapa Locală **Clasele XI–XII**
24 ianuarie 2014

Problema 1 – Benzinării **100 puncte**

Gigel locuie□te într-un ora□ în care există N intersec□ii, numerotate de la 1 la N, între care sunt construite M străzi bidirec□ionale. În P intersec□ii din cele N sunt construite benzinării. Gigel ar vrea să □tie pentru fiecare intersec□ie cea mai apropiată benzinărie de acea intersec□ie. Dacă o intersec□ie este la distan□ă egală fa□ă de două benzinării, Gigel va alege benzinăria aflată în intersec□ia cu numărul de identificare mai mic.

Cerință

Scrieți un program care determină pentru fiecare intersec□ie cea mai apropiată benzinărie.

Date de intrare

Fişierul de intrare *benzinarii.in* con□ine pe prima linie numerele `N`, `M`, `P`. Pe fiecare din următoarele `M` linii se află două numere `X` □i `Y`, indicând o stradă din intersec□ia `X` în intersec□ia `Y`. Pe ultimul rând se află `P` numere, reprezentând intersec□iile în care există benzinării.

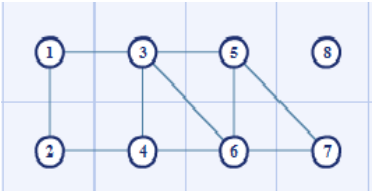
Date de ieşire

Fi□ierul de ie□ire *benzinarii.out* va con□ine o singură linie pe care se află *N* numere, al *i*-lea număr fiind *-1*, dacă din intersec□ia a *i*-a nu se poate ajunge la nicio benzinărie din cele *P*, sau numărul intersec□iei în care se află benzinăria cea mai apropiată de intersec□ia a *i*-a, în caz contrar.

Restricții şi precizări

- Distan□a dintre două intersec□ii este numărul de străzi parcurse între cele două intersec□ii
- $1 \le N \le 100\,000$; $1 \le M \le 200\,000$; $1 \le P \le 1\,000$
- Pentru 50% din teste : $1 \le N \le 1000$; $1 \le M \le 2000$; $1 \le P \le 100$

Exemplu

benzinarii.in	benzinarii.out	Explicații
8 10 2 1 2 1 3 2 4 3 4 3 5 3 6 4 6 5 6 5 7 6 7 7 1	1 1 1 1 7 7 7 -1	 Sunt 2 benzinării: una în intersec□ia 1 □i una în intersec□ia 7. Intersec□iile 1, 2, 3 □i 4 sunt mai apropiate fa□ă de benzinăria din intersec□ia 1. Intersec□iile 5, 6 □i 7 sunt mai apropiate fa□ă de benzinăria din intersec□ia 7. Intersec□ia 4 se află la distan□ă egală fa□ă de cele 2 benzinării, dar se ia în considerare benzinăria 1, deoarece 1 < 7.

Timp maxim de execuție/test : 1 secundă
Limita de memorie: 32 MB din care 8 MB pentru stivă
Dimensiune maximă a sursei: 10 KB.

Ministerul Educației Na□ionale – ISJ Timiş
Olimpiada Na□ională de Informatică - Etapa Locală **Clasele XI–XII**
24 ianuarie 2014

Problema 2 – Hashtag **100 puncte**

Yolo □i Swag joacă un nou joc numit Hashtag. Acest joc presupune N căsu□e consecutive numerotate de la 1 la N □i doi pioni. La început, pionul lui Yolo se află în căsu□a 1, iar pionul lui Swag se află în căsu□a K. La fiecare tură cei doi dau o dată cu banul. Dacă pică pajură, atunci pionul lui Yolo avansează o căsu□ă, în caz contrar pionul lui Swag avansează o căsu□ă. Yolo câ□tigă dacă îl ajunge din urmă pe Swag, adică dacă ajunge cu pionul în căsu□a în care se află pionul lui Swag, iar Swag câ□tigă dacă ajunge cu pionul în căsu□a N.

Cerință

Pentru `N` şi `K` date, Swag vrea să □tie câte jocuri posibile sunt, astfel încât la finalul lor el să fie câ□tigător. Două jocuri sunt diferite dacă diferă prin cel pu□in o mutare în succesiunea de mutări.

Date de intrare

Fişierul de intrare *hashtag.in* conține pe prima linie 2 numere naturale separate printr-un spațiu, `N` şi `K`, cu semnificația din enunț.

Date de ieşire

Datorită faptului ca numărul de posibilită□i este foarte mare, în fi□ierul de ie□ire *hashtag.out* se va scrie un singur număr natural reprezentând valoarea cerută modulo `666013` (restul împăr□irii rezultatului la numărul 666013).

Restricții şi precizări

- $1 \leq K \leq N \leq 2000$
- Pentru 40% din teste : $1 \leq K \leq N \leq 15$
- Pentru 35% din teste, numărul de posibilită□i este mai mic decât 666013
- Swag va atrage aten□ia că (A+B) % 666013 = (A%666013+B%666013) % 666013

Exemplu

hashtag.in	hashtag.out	Explicații
4 2	2	Ini□ial, pionul lui Yolo e în căsu□a 1, iar pionul lui Swag în căsu□a 2. Există două jocuri posibile în urma cărora să câ□tige Swag: în primul joc mută Swag de două ori consecutiv (SS) □i ajunge în căsu□a 3, apoi în căsu□a 4; în al doilea joc mută Swag o dată, apoi Yolo, apoi Swag (SYS), adică pionul lui Swag ajunge în căsu□a 3, apoi pionul lui Yolo ajunge în căsu□a 2, apoi pionul lui Swag ajunge în căsu□a 4 □i câ□tigă.
5 2	5	Sunt 5 posibilită□i de mutări pentru ca Swag să câ□tige acest joc: SSS, SYSS, SYSYS, SSYS, SSYYS.
7 4	28	Sunt 28 de posibilită□i pentru ca Swag să câ□tige acest joc.

Timp maxim de execuție/test : 0.3 secunde
Limita de memorie: 32 MB din care 16 MB pentru stivă
Dimensiune maximă a sursei: 10 KB.

Benzinării - Descrierea soluției

prof. Lupșe-Turpan Mircea,
Liceul Teoretic Grigore Moisil Timișoara

Varianta 1 (100 puncte) - Complexitate O(N+M+P log P)

Se utilizează un algoritm BFS cu noduri de plecare multiple. Toate intersecțiile în care sunt construite benzinării sunt adăugate inițial în coada BFS. Se reține în memorie un vector Sol[i] cu semnificația "benzinăria cea mai apropiată de intersecția i". Se observă că pentru a respecta precizarea cu determinarea benzinăriei cu număr minim este suficientă o sortare inițială a benzinăriilor.

Varianta 2 (100 puncte) - Complexitate O(N+M)

Aceași rezolvare ca și la varianta 1, dar benzinăriile nu se sortează inițial, urmând ca la fiecare pas din BFS să se facă o verificare în plus și eventual o actualizare.

Varianta 3 (70 puncte) - Complexitate O((N+M)*P)

Se aplică un algoritm BFS pentru fiecare benzinărie în parte.

Varianta 4 (50 puncte) - Complexitate O((N+M)*N)

Se aplică un algoritm BFS din fiecare intersecție.

O soluție care implementează graful cu ajutorul matricei de adiacență poate obține cel mult 50 puncte.

O soluție care nu ține cont de precizarea cu determinarea benzinăriei care se află în intersecția cu număr minim poate obține cel mult 30 puncte.

Hashtag - Descrierea soluției

stud. Okros Alexandru,
Universitatea Politehnica Timișoara

Varianta 1 (35-40 puncte) - Complexitate $O(2^N)$ (Mircea Lupșe-Turpan)

Se utilizează un algoritm backtracking. Se generează un șir binar, în care valoarea 0 corespunde mutării pionului lui Yolo, iar valoarea 1 corespunde mutării pionului lui Swag. O soluție parțială este validă dacă diferența dintre numărul de mutări al lui Yolo (numărul de zerouri) și numărul de mutări al lui Swag (numărul de valori de 1) este mai mică decât K-1, unde K-1 reprezintă diferența inițială de căsuțe dintre Yolo și Swag. O soluție parțială este și soluție dacă numărul de mutări al lui Swag este N-K (diferența dintre poziția N și poziția inițială a lui Swag), adică numărul de mutări necesare lui Swag pentru a ajunge pe poziția N.

Varianta 2 (35-40 puncte) - Complexitate $O(2^N)$ (Mircea Lupșe-Turpan)

Se poate optimiza algoritmul de mai sus prin renunțarea la generarea șirului binar și prin utilizarea a două variabile YoloMoves și SwagMoves care reţin în orice moment numărul de mutări al lui Yolo, respectiv numărul de mutări al lui Swag. În acest fel verificarea dacă o soluție parțială este validă, respectiv verificarea dacă o soluție parțială este soluție se face prin compararea acestor 2 variabile.

Varianta 3 (40 puncte) - Complexitate $O(2^N)$

Se bazează pe programare dinamică. Se scrie funcția L(A, B) care calculează numărul de jocuri câștigate de Swag dacă Yolo începe pe poziția A și Swag pe poziția B.

Recurența este L(A,B) = L(A+1,B) + L(A, B+1). Trebuie sa calculăm L(1, K), deoarece Yolo începe pe poziția 1, iar Swag pe poziția K.

Varianta 4 (100 puncte) - Complexitate $O(N^2)$

Este optimizarea soluției precedente folosind memoizare. Valorile deja calculate se reţin într-o matrice, astfel că se evită calcularea unor valori de doua ori.

Complexitatea de memorie $O(N^2)$.

Varianta 5 (100 puncte) - Complexitate $O(N^2)$

Folosim programare dinamica. DP[i][j] = numărul de jocuri câștigate de Swag dacă el e pe poziția j, iar jocul se termină pe poziția i. Recurența: DP[i][j] = DP[i][j+1] + DP[i-1][j-1] (primul caz Swag se mută o căsuță la dreapta, cazul doi Yolo se mută o căsuță și astfel swag și finish-ul se aproprie cu o căsuță, de unde "-1" la parametri).

Complexitatea de memorie $O(N^2)$.

Varianta 6 (100 puncte) - Complexitate $O(N^2)$

La soluția precedentă folosim doar ultimile două linii ale matricei și observăm că putem obține număr de operații N*(N-K) deoarece nu trebuie calculate toate valorile din matrice (rezultă din recurență).

Prin utilizarea unei matrici cu 2 linii si N coloane, complexitatea de memorie scade la O(N).