Problema Parcare

Fişier de intrare	`parcare.in`
Fişier de ieşire	`parcare.out`

În cel mai recent eveniment al companiei Tesla, Paul Musk a anunțat un nou produs inovativ: parcarea autonomă. Fiind cunoscut pentru lansările produselor incomplete, nici parcarea nu este completă, fiind nevoie de o automatizare pentru a atribui câte un loc mașinilor care vor să folosească parcarea.

Parcarea este formată din N locuri, numerotate de la 1 la N, și este deschisă timp de T secunde, începând cu secunda 1. Pe parcursul zilei, sosesc M mașini care vor să folosească parcarea, pentru fiecare dintre acestea știindu-se timpul de sosire s_i și timpul de plecare p_i. Mașinile vin în ordinea timpului de sosire s_i și ocupă locul de parcare în intervalul de timp $[s_i, p_i]$. Pentru fiecare dintre acestea, trebuie să afișați un loc liber de parcare (dacă sunt mai multe, se poate afișa oricare) în care aceasta se poate așeza sau -1 dacă parcarea este plină în momentul venirii mașinii. Dacă o mașină nu are loc în parcare la timpul de sosire, aceasta nu va mai intra în parcare la niciun timp viitor.

La final, Paul este interesat de mașinile care mai sunt rămase în parcare la închiderea parcării, de aceea, vă cere să afișați configurația parcării la timpul T.

Date Intrare

Pe prima linie se găsesc trei numere întregi N, M și T, reprezentând numărul de locuri din parcare, numărul de mașini care vin să folosească parcarea, respectiv numărul de secunde pentru care este deschisă parcarea.

Următoarele M linii conțin fiecare câte două numere întregi s_i, p_i, reprezentând venirea unei mașini la secunda s_i care va pleca la secunda p_i.

Mașinile apar în fișierul de intrare în ordine crescătoare după timpul de sosire s_i.

Date Ieșire

Se vor afișa $M + 1$ linii în total, primele M linii conținând fiecare câte un număr întreg între 1 și N reprezentând locul de parcare pe care îl va ocupa mașina, sau -1 dacă nu există niciun loc de parcare disponibil.

Ultima linie va conține N numere întregi, reprezentând configurația parcării la închidere, unde cel de-al i-lea număr reprezintă **timpul de sosire** al mașinii de pe locul de parcare i, sau -1 dacă locul de parcare i este gol.

Restricții

- $1 \leq N, M, T \leq 200\,000$
- $1 \leq s_i \leq T$
- $1 \leq s_i < p_i \leq 200\,000$
- Considerând următoarele $2M$ valori: $s_1, s_2, \ldots, s_M, p_1, p_2, \ldots, p_M$, acestea sunt distincte două câte două.
- **Dacă există mai multe soluții, se poate afișa oricare dintre acestea.**

#	Punctaj	Restricţii
1	24	$s_i + 1 = p_i$, adică fiecare mașină stă exact o secundă.
2	26	$p_i > s_j$, adică toate mașinile vin înainte ca vreo mașină să plece.
3	26	$N \leq 1\,000$
4	24	Fără restricții suplimentare.

Exemple

parcare.in	parcare.out	Explicații
2 4 6 1 3 2 10 4 6 5 8	2 1 2 -1 2 -1	Prima mașină sosește în secunda 1 și este parcată pe locul 2. A doua mașină sosește în secunda 2 și este parcată pe locul 1. În secunda 3 se eliberează locul 2. Cea de-a treia mașină sosește în secunda 4 și ocupă locul 2. Mașina sosită în secunda 5 nu găsește loc liber. În secunda 6 se eliberează locul 2. După închiderea parcării, pe locul 1 va fi parcată mașina venită în secunda 2, locul al doilea fiind liber.

Problema Turcane

Fişier de intrare `turcane.in`
Fişier de ieşire `turcane.out`

Pe un câmp asemănător cu o tablă de şah cu M linii şi N coloane, o ţurcană se află în pătrăţelul de coordonate $(1, 1)$ aflat în colţul din stânga-sus al tablei şi vrea să ajungă în pătrăţelul de coordonate (M, N) aflat în colţul din dreapta-jos al tablei. Ea poate efectua sărituri de lungime cel mult P la dreapta, de lungime cel mult Q în jos, de lungime cel mult R pe diagonală spre dreapta-jos, precum şi săritura calului, adică două pătrăţele la dreapta şi unul în jos sau două în jos şi unul la dreapta. Orice săritură trebuie să schimbe poziţia ţurcanei.

Se dă un număr întreg C.

- Dacă $C = 1$, să se determine numărul minim de sărituri necesare pentru a ajunge în pătrăţelul de coordonate (M, N).

- Dacă $C = 2$, să se determine numărul de moduri în care poate să ajungă în pătrăţelul de coordonate (M, N), nu neapărat cu număr minim de sărituri.

Se garantează că pentru datele de intrare există cel puţin un mod de a ajunge în pătrăţelul (M, N).

Date Intrare

Pe prima linie a fişierului de intrare se află numărul C, pe a doua linie numerele întregi M şi N, iar pe a treia linie numerele întregi P, Q şi R.

Date Ieșire

În fişierul de ieşire se va afişa, corespunzător valorii lui C, numărul cerut. Rezultatul se va afişa modulo $1\,000\,000\,007$.

Restricții

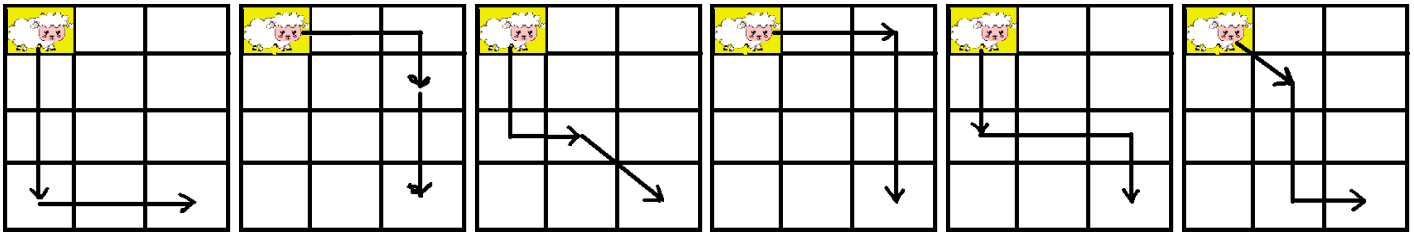
- $1 \leq M, N \leq 1\,000$
- $0 \leq P \leq N - 1$
- $0 \leq Q \leq M - 1$
- $0 \leq R \leq min(M - 1, N - 1)$
- dacă $P = 0$ nu poate sări la dreapta, dacă $Q = 0$ nu poate sări în jos, iar dacă $R = 0$ nu poate sări pe diagonală

#	Punctaj	Restricţii
1	26	$C = 1,\ M = 1$
2	15	$C = 1,\ M = 2$
3	7	$C = 1,\ M = 3$
4	7	$C = 1,\ 1 \leq M, N \leq 200$
5	8	$C = 1,\ 1 \leq M, N \leq 1000$
6	11	$C = 2,\ 1 \leq M, N \leq 5$
7	12	$C = 2,\ 1 \leq M, N \leq 200$
8	14	$C = 2,\ 1 \leq M, N \leq 1\,000$

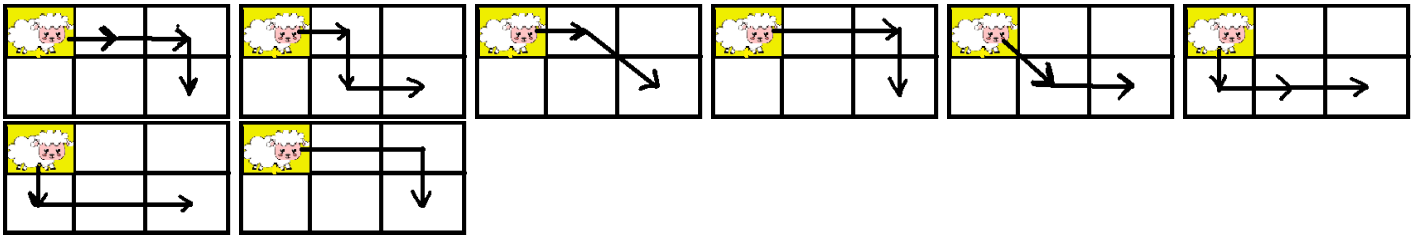
Exemple

turcane.in	turcane.out	Explicații
1 4 3 2 3 1	2	Notăm cu O_i săritura la dreapta cu i pătrăţele, cu V_i săritura în jos cu i pătrăţele, cu D_i săritura pe diagonală cu i pătrăţele, cu Cd săritura calului spre dreapta-jos şi cu Cj săritura calului spre jos-dreapta. Numărul minim de sărituri este 2, şi avem şase soluţii: $V_3 - O_2$ sau $Cd - V_2$ sau $Cj - D_1$ sau $O_2 - V_3$ sau $V_2 - Cd$ sau $D_1 - Cj$.
2 2 3 2 1 1	8	Cele opt moduri de a ajunge în pătrăţelul $(2, 3)$ sunt: $O_1 - O_1 - V_1$, $O_1 - V_1 - O_1$, $O_1 - D_1$, $O_2 - V_1$, $D_1 - O_1$, $V_1 - O_1 - O_1$, $V_1 - O_2$, Cd

Pentru primul exemplu, numărul minim de sărituri este 2. Cele șase soluții cu număr minim de sărituri sunt ilustrate în figurile următoare:



Pentru al doilea exemplu, numărul soluțiilor distincte este 8. Pentru fiecare soluție, săriturile țurcanei sunt ilustrate în figurile următoare:



Problema Veri

Fişier de intrare	`veri.in`
Fişier de ieşire	`veri.out`

Se dă un graf **orientat** cu n noduri și m muchii. Fiecare muchie are costul 1 (poate fi parcursă într-un minut). Doi "prieteni" (*veri*) pornesc din nodul S. Unul dintre ei vrea să ajungă în nodul A, iar celălalt vrea să ajungă în nodul B.

Cei doi prieteni se vor plimba împreună până când *ciclează*, adică până când vor ajunge în același nod a doua oară, notat cu Z. După ciclare, ei își pot continua drumurile separat. Totuși, dacă vor, pot să meargă amândoi în continuare pe același drum: doar dispare obligația de a merge împreună.

Fiecare dintre ei trebuie să-și termine drumul doar după ciclare, adică după ce nu mai sunt obligați să meargă împreună. Totuși, este în regulă dacă drumul unuia se termină exact în nodul în care au ciclat (adică ciclează în A sau B).

Care este numărul minim de minute necesar, astfel încât să fie posibil ca amândoi să ajungă la destinațiile lor, în timpul alocat, în A, respectiv B?

Cu alte cuvinte, dacă cei doi veri ciclează pentru prima oară după exact t minute, apoi își continuă drumurile pentru alte t_A, respectiv t_B minute, vrem să aflăm valoarea minimă a lui $\max(t + t_A, t + t_B)$.

Există două tipuri de cerințe, reprezentate printr-un număr c:

- Dacă $c = 1$, trebuie calculată valoarea minimă a lui $\max(t + t_A, t + t_B)$.
- Dacă $c = 2$, trebuie afișat un triplet de drumuri care poate fi urmat de cei doi veri (drumul comun din S până în Z, drum urmat ulterior de primul văr din Z până în A, drum urmat ulterior de al doilea văr din Z până în B), astfel încât valoarea asociată drumurilor, adică $\max(t + t_A, t + t_B)$ să fie minimă. Orice triplet corect cu valoarea asociată minimă poate fi afișat.

Date Intrare

Pe prima linie se găsește c. Pe a doua linie se găsesc doi întregi n și m. Pe a treia linie se găsesc trei întregi S, A și B.

Pe următoarele m linii se găsesc câte doi întregi X și Y, reprezentând că există o muchie direcționată de la nodul X la nodul Y, care poate fi parcursă într-un minut (de cost 1).

Date Ieșire

Dacă $c = 1$, afișați un singur număr, valoarea minimă a lui $\max(t + t_A, t + t_B)$.

Dacă $c = 2$, afișati trei drumuri. Primul drum este format de la S până la Z. Al doilea drum este format de la Z până la A. Al treilea drum este format de la Z până la B, unde S, A, B, Z sunt definite anterior.

Fiecare drum se va tipări pe două linii separate.

- Pe prima linie va apărea lungimea drumului, adică numărul de muchii.
- Pe a doua linie vor apărea nodurile drumului, separate prin câte un spațiu.

Valoarea asociată drumurilor, adică $\max(t + t_A, t + t_B)$, trebuie să fie minimă.

Restricții

- $1 \leq S, A, B, Z \leq n \leq 5000$.
- Nodurile sunt numerotate de la 1 la n.
- $A \neq B$.
- $1 \leq m \leq n(n-1)$.
- Se garantează că pentru orice test dat spre rezolvare există cel puțin o soluție.
- Nu există muchii de la un nod la el însuși. Există maxim o muchie orientată între oricare două noduri distincte.
- Dacă verii se despart în A, primul văr poate să nu mai facă nimic (drumul lui ulterior ar avea 0 muchii și l-ar conține doar pe A: vezi exemplul 3). Analog pentru B.
- Pentru fiecare subtask, testele cu $c = 1$ vor conta pentru 60% din punctaj.

#	Punctaj	Restricţii
1	30	$n \leq 500$, $m = n$ și toate muchiile sunt de forma $i \to (i \mod n) + 1$, unde $i \in \{1, \ldots, n\}$.
2	50	$n \leq 500$
3	20	$n \leq 5000$ și $m \leq 4n$.

Exemple și Explicații

Exemplul 1		Exemplul 2		Exemplul 3		Exemplul 4	
veri.in	**veri.out**	**veri.in**	**veri.out**	**veri.in**	**veri.out**	**veri.in**	**veri.out**
2 7 8 1 3 4 1 2 2 5 5 7 7 6 6 7 6 5 5 3 7 4	5 1 2 5 7 6 5 1 5 3 2 5 7 4	2 7 8 1 2 5 1 3 1 4 3 2 4 5 6 4 7 3 3 6 4 7	5 1 4 7 3 6 4 3 4 7 3 2 1 4 5	2 5 6 1 3 5 1 2 2 3 3 4 4 3 3 1 3 5	4 1 2 3 4 3 0 3 1 3 5	1 4 4 1 2 4 1 3 3 2 2 3 2 4	5

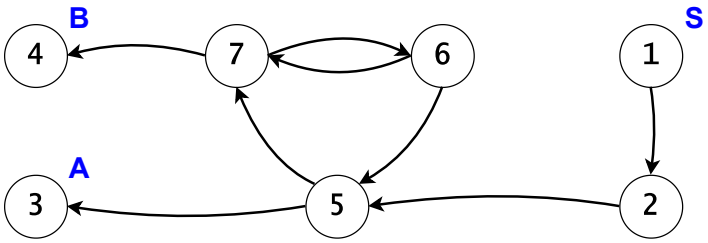


Figura 1: Drumul urmat în comun de cei doi este $1 \to 2 \to 5 \to 7 \to 6 \to 5$. Drumul urmat de primul văr în continuare este $5 \to 3$. Drumul continuat de al doilea văr este $5 \to 7 \to 4$. Astfel, primul văr are nevoie de 6 minute pentru a ajunge în A, iar al doilea de 7 minute pentru a ajunge în B, deci răspunsul pentru $c = 1$ este 7. Cei doi ar fi putut să cicleze în 7, dacă ar fi urmat drumul $1 \to 2 \to 5 \to 7 \to 6 \to 7$. Totuși, deși al doilea văr ar fi ajuns în B în doar 6 minute ($7 \to 4$), primul văr ar fi avut nevoie de cel puțin 8 minute ca să ajungă în A ($7 \to 6 \to 5 \to 3$).

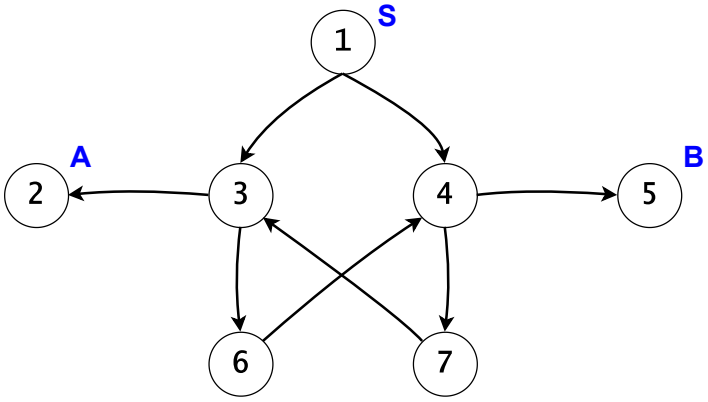


Figura 2: Răspunsul corect pentru $c = 1$ este 8. Pentru acest exemplu există două soluții corecte. A doua soluție este tripletul de drumuri ($1 \to 3 \to 6 \to 4 \to 7 \to 3$, $3 \to 2$, $3 \to 6 \to 4 \to 5$).

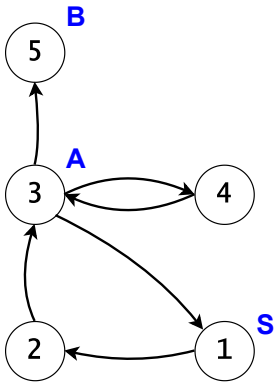


Figura 3: Răspunsul corect pentru $c = 1$ este 5. Este folosit un ciclu care se termină în $A = 3$.

Figura 4: Pentru $c = 2$, singurul triplet corect de drumuri este ($1 \to 3 \to 2 \to 3$, $3 \to 2$, $3 \to 2 \to 4$).
Atenție! Tripletul ($1 \to 3 \to 2 \to 3 \to 2$, 2, $2 \to 4$) este greșit, deoarece primul nod vizitat a doua oară nu este 2.