

Problema Dominoes

Fișier de intrare `dominoes.in`
Fișier de ieșire `dominoes.out`

Se dă o matrice cu 2 linii și n coloane care are k celule ocupate.

Se dau q interogări de forma (x_1, y_1, x_2, y_2) , cu următoarea semnificație: dacă se ocupă **două celule libere distincte** ale matricii inițiale, (x_1, y_1) și (x_2, y_2) , se poate pava complet matricea cu piese de domino de dimensiuni 2×1 și 1×2 ? După efectuarea unei interogări celulele ocupate asociate acestora vor deveni din nou libere (modificările aduse matricii nu persistă între interogări).

Cerință

Să se determine, pentru fiecare interogare, dacă este posibil ca matricea să fie pavată complet cu piese de domino de dimensiuni 2×1 și 1×2 .

Date de intrare

Pe prima linie a fișierului `dominoes.in` se află n și k .

Pe următoarele k linii se află coordonatele pozițiilor ocupate, de forma (x, y) .

Pe a $(k + 2)$ -a linie se află q . Pe fiecare dintre următoarele q linii se găsesc valorile x_1, y_1, x_2 și y_2 , separate prin câte un spațiu, cu semnificația din cerință.

Date de ieșire

Fișierul de ieșire `dominoes.out` va conține q linii, pe fiecare linie aflându-se răspunsul (1 dacă se poate, respectiv 0 dacă nu se poate) corespunzător câte unei interogări, în ordinea în care acestea se găsesc în fișierul de intrare.

Restricții

- Se garantează faptul că matricea inițială se poate pava complet cu piese de domino.
- $1 \leq x, x_1, x_2 \leq 2$
- $1 \leq y, y_1, y_2 \leq n$
- $0 \leq k \leq 100\,000$, k par
- $1 \leq q \leq 100\,000$
- $1 \leq n \leq 1\,000\,000\,000$

#	Punctaj	Restricții
1	9	$n, k, q \leq 8$
2	6	$k = 0$
3	11	$n, k, q \leq 5\,000$
4	12	$k, q \leq 5\,000$
5	23	$n \leq 100\,000$
6	6	$y_1 = y_2$ pentru fiecare interogare
7	8	$y_1 \neq y_2$ pentru fiecare interogare
8	25	fără alte restricții suplimentare

Exemple

dominoes.in	dominoes.out	Explicații
5 2 1 1 2 5 4 1 3 2 3 2 1 1 5 2 3 2 4 1 2 2 4	0 1 1 1	Dedesubt se găsesc configurațiile matricei pentru fiecare interogare, în ordinea din input, cu negru fiind marcate celulele blocate:
6 4 1 1 2 3 1 4 2 6 3 1 2 2 2 1 2 1 3 1 2 1 5	0 1 0	Configurațiile matricei pentru fiecare interogare:
6 2 1 1 2 3 3 1 4 2 6 1 4 2 5 1 3 2 5	1 0 0	Configurațiile matricei pentru fiecare interogare:

Problema Seqstr

Fișier de intrare `seqstr.in`
Fișier de ieșire `seqstr.out`

Se dau două șiruri, A și B , cu valori din mulțimea $\{0, 1\}$.

Cerință

1. Să se afle numărul de subsecvențe distincte din B care sunt subșiruri în A .
2. Să se afle, pentru o subsecvență $B_{p...q}$, numărul de subșiruri din A egale cu aceasta.
3. Să se afle numărul de subșiruri din A care sunt subsecvențe în B .

Date de intrare

Pe prima linie a fișierului `seqstr.in` se află n reprezentând lungimea șirului A . Pe linia a doua se află cele n elemente ale șirului A separate prin câte un spațiu. Pe linia a treia se află numărul m reprezentând lungimea șirului B . Pe linia a patra se află cele m elemente ale șirului B separate prin câte un spațiu. Pe linia a cincea se află numărul C reprezentând cerința 1, 2 sau 3.

Dacă C este 2 atunci pe linia a șasea se află două numere p și q separate printr-un spațiu.

Date de ieșire

Fișierul de ieșire `seqstr.out` va conține o singură linie pe care va fi scris numărul cerut, conform cerinței.

Restricții

- Rezultatele cerute vor fi calculate și afișate modulo $10^9 + 7$
- Prin subșir T de lungime p al lui A se înțelege un șir

$$A_{t_1}, A_{t_2}, A_{t_3}, \dots, A_{t_p}$$

determinat de pozițiile

$$1 \leq t_1 < t_2 < t_3 < \dots < t_p \leq n$$

- Orice subsecvență a lui B este determinată de două poziții

$$1 \leq p \leq q \leq m$$

și este egală cu șirul

$$B_p, B_{p+1}, \dots, B_q.$$

Lungimea secvenței este egală cu $q - p + 1$.

- Două subsecvențe din B , determinate respectiv de perechile de poziții p_1, q_1 și p_2, q_2 sunt **distincte** dacă au lungimi diferite sau dacă au lungimi egale și există k astfel încât $p_1 \leq p_1 + k \leq q_1$ și $p_2 \leq p_2 + k \leq q_2$ și $B_{p_1+k} \neq B_{p_2+k}$.
- $1 \leq m \leq n$

#	Punctaj	Restricții
1	7	$C = 1$ și $n \leq 20$
2	9	$C = 1$ și $21 \leq n \leq 500$
3	19	$C = 1$ și $501 \leq n \leq 5000$
4	3	$C = 2$ și $n \leq 20$
5	9	$C = 2$ și $21 \leq n \leq 100$
6	15	$C = 2$ și $101 \leq n \leq 5000$
7	9	$C = 3$ și $n \leq 20$
8	9	$C = 3$ și $21 \leq n \leq 100$
9	20	$C = 3$ și $101 \leq n \leq 500$

Exemple

seqstr.in	seqstr.out	Explicații
5 1 1 0 0 1 3 0 1 1 1	4	Avem de rezolvat cerința 1. Subsecvențele distincte din șirul B pentru care există subșir în A sunt: 0, 1, 0 1 și 1 1. Pentru subsecvența 0 1 1 din B nu există subșir în A.
5 1 1 0 0 1 3 0 1 1 2 2 3	3	Avem cerința 2. Subșirurile din A egale cu subsecvența 1 1 din B sunt cele determinate de subșirurile de poziții: 1 2, 1 5 și 2 5
5 1 1 0 0 1 3 0 1 1 3	10	Avem cerința 3. Vor fi analizate numai subșirurile de lungime 1, 2 sau 3. Nu avem subșiruri de lungime 3 în A egale cu subsecvențe din B, iar din cele de lungime 1 sau 2 sunt numărate cele egale cu 0, 1, 0 1 și 1 1.

Problema Teze

Fișier de intrare `teze.in`
Fișier de ieșire `teze.out`

Profesorul de informatică trebuie să corecteze tezele a m elevi. Elevii au avut de rezolvat n probleme în teză, numerotate de la 1 la n . Fiecare elev a rezolvat toate problemele, deci profesorul are de corectat în total $m \times n$ probleme. El observă că luând lucrările la rând și corectând toate problemele din prima lucrare, apoi toate problemele din a doua lucrare, ș.a.m.d., nu obține întotdeauna cel mai mic timp total de corectare a tezelor. Profesorul are următoarea metodă de corectură:

- Toate tezele se vor corecta în f faze ($1 \leq f \leq n$).
- În fiecare fază i se va decide asupra unei submulțimi de probleme care nu au fost deja corectate, S_i . Aceste probleme se vor corecta în toate tezele înainte să se revină la prima teză.
- Formal, alegem $S_i \subseteq \{1, \dots, n\}$ astfel încât:
 - $S_i \cap S_j = \emptyset, \forall i, j \in \{1, \dots, f\}, i \neq j$
 - $S_1 \cup \dots \cup S_f = \{1, \dots, n\}$

La începerea corectării fiecărei teze, trebuie identificat numele elevului, proces care durează exact p secunde de fiecare dată, chiar dacă se revine la aceeași teză de mai multe ori.

După începerea corectării unei teze, căutarea fiecărei probleme durează k secunde. Corectarea primei probleme din submulțimea aleasă durează t_1 secunde, corectarea celei de-a doua probleme durează t_2 secunde ș.a.m.d. Se garantează că $t_1 < t_2 < \dots < t_n$. De fiecare dată când se revine la o anumită teză și se reîncepe corectarea ei cu o altă submulțime de probleme, corectarea primei probleme din submulțime va dura din nou t_1 secunde.

Valoarea t_1 va fi dată în fișierul de intrare împreună cu valorile $d_0, \dots, d_{q-1}$, din care se vor obține celelalte valori din șirul t prin formula: $t_i = t_{i-1} + d_{i \bmod q}, \forall i \in \{2, \dots, n\}$, unde $x \bmod y$ semnifică restul împărțirii lui x la y .

Cerință

Să se determine timpul minim în care pot fi corectate cele m lucrări.

Date de intrare

Pe prima linie a fișierului de intrare `teze.in` se află numerele n, m, p și k , despărțite prin câte un spațiu. Pe a doua linie se află două valori t_1 și q , despărțite printr-un spațiu. Pe următoarele q linii se află valorile $d_0, \dots, d_{q-1}$.

Date de ieșire

În fișierul de ieșire `teze.out` se va scrie un singur număr, reprezentând numărul de secunde în care se pot corecta tezele mod $10^9 + 7$.

Restricții

- $1 \leq n \leq 1\,500\,000\,000$, unde n este numărul problemelor din teze
- $1 \leq m, q, k \leq 1\,000$, unde m este numărul elevilor, q este lungimea șirului d , iar k reprezintă numărul de secunde necesar căutării unei probleme
- $1 \leq p \leq 10^{10}$, unde p este numărul de secunde necesar identificării numelui unui elev
- $1 \leq t_1, d_i \leq 10$

#	Punctaj	Restricții
1	5	$n \leq 25$
2	10	$n \leq 50$
3	20	$n \leq 10\,000$
4	30	$n \leq 10^7$
5	7	$q = 1$
6	8	$q = 2$
7	20	fără alte restricții suplimentare

Exemple

teze.in	teze.out	Explicații
2 10 5 2 1 1 2	130	Avem două probleme în teză, iar $t_1 = 1$, $t_2 = t_1 + d_0 = 1 + 2 = 3$. Dacă le corectăm împreună, timpul de corectare pentru fiecare teză va fi: 5 secunde pentru găsirea numelui elevului, 2 secunde pentru găsirea primei probleme, 1 secundă pentru corectarea primei probleme, 2 secunde pentru găsirea problemei a doua și 3 secunde pentru corectarea problemei a doua, deci în total $5 + 2 + 1 + 2 + 3 = 13$ secunde pentru fiecare dintre cele 10 teze, adică 130 de secunde în total. O altă posibilitate este să corectăm prima dată problema 1 în toate tezele, iar pe urmă să corectăm problema 2 în toate tezele. Pentru corectarea primei probleme avem: 5 secunde începerea corectării, 2 secunde găsirea problemei și 1 secundă corectarea problemei, în total $5 + 2 + 1 = 8$ pentru o teză, în total 80 de secunde pentru toate tezele. Pentru corectarea problemei 2 avem exact același calcul, rezultă în total 160 de secunde, deci prima metodă a fost mai eficientă și posibilitate mai bună nu există.