acm International Collegiate Programming Contest 2018

Maratona de Programação da SBC 2018

Sub-Regional Brasil do ACM ICPC

15 de Setembro de 2018

Caderno de Problemas

Informações Gerais

Este caderno contém 13 problemas; as páginas estão numeradas de 1 a 22, não contando esta página de rosto. Verifique se o caderno está completo.

A) Sobre os nomes dos programas
1) Para soluções em C/C++ e Python, o nome do arquivo-fonte não é significativo, pode ser qualquer nome.
2) Se sua solução é em Java, ela deve ser chamada *codigo_de_problema*`.java` onde *codigo_de_problema* é a letra maiúscula que identifica o problema. Lembre que em Java o nome da classe principal deve ser igual ao nome do arquivo.
3) Se sua solução é em Kotlin, ela deve ser chamada *codigo_de_problema*`.kt` onde *codigo_de_problema* é a letra maiúscula que identifica o problema. Lembre que em Kotlin o nome da classe principal deve ser igual ao nome do arquivo.

B) Sobre a entrada
1) A entrada de seu programa deve ser lida da *entrada padrão*.
2) A entrada é composta de um único caso de teste, descrito em um número de linhas que depende do problema.
3) Quando uma linha da entrada contém vários valores, estes são separados por um único espaço em branco; a entrada não contém nenhum outro espaço em branco.
4) Cada linha, incluindo a última, contém exatamente um caractere final-de-linha.
5) O final da entrada coincide com o final do arquivo.

C) Sobre a saída
1) A saída de seu programa deve ser escrita na *saída padrão*.
2) Quando uma linha da saída contém vários valores, estes devem ser separados por um único espaço em branco; a saída não deve conter nenhum outro espaço em branco.
3) Cada linha, incluindo a última, deve conter exatamente um caractere final-de-linha.

Promoção:

SBC Sociedade Brasileira de Computação

v1.0

Problema A

Aventurando-se no Slackline

Beltrano recentemente se interessou por *slackline*. Slackline é um esporte de equilíbrio sobre uma fita elástica esticada entre dois pontos fixos, o que permite ao praticante andar e fazer manobras em cima da fita. Durante as férias tudo que Beltrano quer fazer é praticar, e para isso ele foi para a fazenda de um amigo, onde há uma plantação de eucaliptos.

A plantação é muito bem organizada. Os eucaliptos estão dispostos em N fileiras com M árvores em cada. Há um espaço de um metro entre cada fileira e as árvores nas diferentes fileiras estão todas perfeitamente alinhadas com um espaço de um metro entre elas.

Beltrano vai montar o slackline usando duas árvores. Ao montar o slackline Beltrano não gosta que a distância entre as duas árvores seja muito pequena, já que as melhores manobras exigem que a fita tenha pelo menos L metros. Também não é possível esticar demais a fita já que ela tem um comprimento máximo de R metros. Note que ao esticar a fita entre as duas árvores escolhidas não pode haver nenhuma outra árvore na linha formada, caso contrário não seria possível utilizar a fita toda para as manobras.

Beltrano gostaria de saber de quantas formas diferentes é possível montar o slackline usando as árvores da fazenda. Duas formas são consideradas diferentes se pelo menos uma das árvores onde a fita foi amarrada é diferente.

Entrada

A entrada consiste de uma única linha que contém quatro inteiros, N, M, L, R, representando respectivamente o número de linhas e colunas da plantação e os comprimentos mínimo e máximo do slackline ($1 \leq N, M \leq 10^5$; $1 \leq L \leq R \leq 10^5$).

Saída

Seu programa deve produzir uma única linha com um inteiro representando de quantas formas diferentes o slackline pode ser montado. Como o resultado pode ser grande, a resposta deve ser esse número módulo $10^9 + 7$.

Exemplo de entrada 1	**Exemplo de saída 1**
`2 2 1 1`	`4`

Exemplo de entrada 2	**Exemplo de saída 2**
`2 3 1 4`	`13`

Exemplo de entrada 3	**Exemplo de saída 3**
`3 4 1 4`	`49`

Problema B

Bolinhas de Gude

Usar bolinhas de gude como moeda não deu muito certo em Cubicônia. Na tentativa de se redimir com seus amigos, depois de roubar suas bolinhas de gude, o imperador decidiu convidar todos para uma noite de jogos em seu palácio.

Naturalmente, os jogos utilizam bolinhas de gude, afinal agora o imperador precisa encontrar alguma utilidade para tantas bolinhas. N bolinhas de gude são espalhadas em um grande tabuleiro cujas linhas são numeradas de 0 a L e as colunas numeradas de 0 a C. Os jogadores alternam turnos e em cada turno o jogador da vez deve escolher uma das bolinhas de gude e movê-la. O primeiro jogador que mover uma bolinha para a posição $(0, 0)$ é o vencedor. Para que o jogo seja interessante, os movimentos são limitados; do contrário, o primeiro jogador sempre moveria a bolinha para a posição $(0, 0)$ e venceria. Um movimento consiste em escolher um inteiro u maior que 0 e uma bolinha, cuja localização denotaremos por (l, c), e movê-la para uma das seguintes posições, desde que a mesma não saia do tabuleiro:

- $(l - u, c)$;
- $(l, c - u)$; ou
- $(l - u, c - u)$.

Note que mais de uma bolinha de gude pode ocupar a mesma posição no tabuleiro.

Como o imperador não gosta de perder você deve ajudá-lo a determinar em quais partidas ele deve participar. Como é de se esperar, sempre que joga o imperador fica com o primeiro turno. Assumindo que todos jogam de forma ótima, seu programa deve analisar a distribuição inicial das bolinhas de gude no tabuleiro e informar se é possível ou não que o imperador vença caso ele jogue.

Entrada

A primeira linha contém um inteiro N ($1 \leq N \leq 1000$). Cada uma das N linhas seguintes contém dois inteiros l_i e c_i indicando em qual linha e coluna a i-ésima bolinha de gude se encontra no tabuleiro ($1 \leq l_i, c_i \leq 100$).

Saída

Seu programa deve produzir uma única linha contendo o caractere `Y` caso seja possível para o imperador ganhar o jogo ou `N` caso contrário.

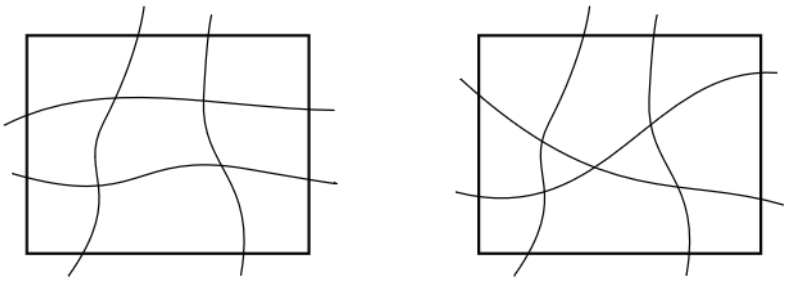
Exemplo de entrada 1	**Exemplo de saída 1**
2 1 3 2 3	Y

Exemplo de entrada 2	**Exemplo de saída 2**
1 1 2	N

Problema C

Cortador de Pizza

Vô Giuseppe ganhou de presente um cortador profissional de pizza, daqueles do tipo carretilha e, para comemorar, assou uma pizza retangular gigante para seus netos! Ele sempre dividiu suas pizzas em pedaços fazendo cortes ao longo de linhas contínuas, não necessariamente retilíneas, de dois tipos: algumas começam na borda esquerda da pizza, seguem monotonicamente para a direita e terminam na borda direita; outras começam na borda inferior, seguem monotonicamente para cima e terminam na borda superior. Mas Vô Giuseppe sempre seguia uma propriedade: dois cortes do mesmo tipo nunca podiam se interceptar. Veja um exemplo com 4 cortes, dois de cada tipo, na parte esquerda da figura, que dividem a pizza em 9 pedaços.



Acontece que Vô Giuseppe simplesmente ama geometria, topologia, combinatória e coisas assim; por isso, resolveu mostrar para as crianças que poderia obter mais pedaços, com o mesmo número de cortes, se cruzamentos de cortes de mesmo tipo fossem permitidos. A parte direita da figura mostra, por exemplo, que se os dois cortes do tipo dos que vão da esquerda para a direita puderem se interceptar, a pizza será dividida em 10 pedaços.

Vô Giuseppe descartou a propriedade, mas não vai fazer cortes aleatórios. Além de serem de um dos dois tipos, eles vão obedecer às seguintes restrições:

- Dois cortes têm no máximo um ponto de interseção e, se tiverem, é porque os cortes se cruzam naquele ponto;
- Três cortes não se interceptam num mesmo ponto;
- Dois cortes não se interceptam na borda da pizza;
- Um corte não intercepta um canto da pizza.

Dados os pontos de começo e término de cada corte, seu programa deve computar o número de pedaços resultantes dos cortes do Vô Giuseppe.

Entrada

A primeira linha da entrada contém dois inteiros X e Y, $(1 \leq X, Y \leq 10^9)$, representando as coordenadas (X, Y) do canto superior direito da pizza. O canto inferior esquerdo tem sempre coordenadas $(0, 0)$. A segunda linha contém dois inteiros H e V, $(1 \leq H, V \leq 10^5)$, indicando, respectivamente, o número de cortes que vão da esquerda para a direita, e o número de cortes que vão de baixo para cima. Cada uma das H linhas seguintes contém dois inteiros Y_1 e Y_2 definindo as ordenadas de encontro dos lados verticais da pizza com um corte que vai do lado esquerdo, na ordenada Y_1, para o lado direito, na ordenada Y_2. Cada uma das V linhas seguintes contém dois inteiros X_1 e X_2 definindo as abscissas de encontro dos lados horizontais da pizza com um corte que vai do lado inferior, na abscissa X_1, para o lado superior, na abscissa X_2.

Saída

Imprima uma linha contendo um inteiro representando o número de pedaços resultantes.

Exemplo de entrada 1	**Exemplo de saída 1**
3 4 3 2 1 2 2 1 3 3 1 1 2 2	13

Exemplo de entrada 2	**Exemplo de saída 2**
5 5 3 3 2 1 3 2 1 3 3 4 4 3 2 2	19

Exemplo de entrada 3	**Exemplo de saída 3**
10000 10000 1 2 321 3455 10 2347 543 8765	6

Problema D

Desvendando Monty Hall

No palco de um programa de auditório há três portas fechadas: porta 1, porta 2 e porta 3. Atrás de uma dessas portas há um carro, atrás de cada uma das outras duas portas há um bode. A produção do programa sorteia aleatoriamente a porta onde vai estar o carro, sem trapaça. Somente o apresentador do programa sabe onde está o carro. Ele pede para o jogador escolher uma das portas. Veja que agora, como só há um carro, atrás de pelo menos uma entre as duas portas que o jogador não escolheu, tem que haver um bode!

Portanto, o apresentador sempre pode fazer o seguinte: entre as duas portas que o jogador não escolheu, ele abre uma que tenha um bode, de modo que o jogador e os espectadores possam ver o bode. O apresentador, agora, pergunta ao jogador: "você quer trocar sua porta pela outra porta que ainda está fechada?". É vantajoso trocar ou não? O jogador quer ficar com a porta que tem o carro, claro!

Paulinho viu uma demonstração rigorosa de que a probabilidade de o carro estar atrás da porta que o jogador escolheu inicialmente é 1/3 e a probabilidade de o carro estar atrás da outra porta, que ainda está fechada e que o jogador não escolheu inicialmente, é 2/3 e, portanto, a troca é vantajosa. Paulinho não se conforma, sua intuição lhe diz que tanto faz, que a probabilidade é 1/2 para ambas as portas ainda fechadas...

Neste problema, para acabar com a dúvida do Paulinho, vamos simular esse jogo milhares de vezes e contar quantas vezes o jogador ganhou o carro. Vamos supor que:

- O jogador sempre escolhe inicialmente a porta 1;
- O jogador sempre troca de porta, depois que o apresentador revela um bode abrindo uma das duas portas que não foram escolhidas inicialmente.

Nessas condições, em um jogo, dado o número da porta que contém o carro, veja que podemos saber exatamente se o jogador vai ganhar ou não o carro.

Entrada

A primeira linha da entrada contém um inteiro N ($1 \leq N \leq 10^4$), indicando o número de jogos na simulação. Cada uma das N linhas seguintes contém um inteiro: 1, 2 ou 3; representando o número da porta que contém o carro naquele jogo.

Saída

Seu programa deve produzir uma única linha, contendo um inteiro representando o número de vezes que o jogador ganhou o carro nessa simulação, supondo que ele sempre escolhe inicialmente a porta 1 e sempre troca de porta depois que o apresentador revela um bode abrindo uma das duas portas que não foram escolhidas inicialmente.

Exemplo de entrada 1	**Exemplo de saída 1**
5 1 3 2 2 1	3