CI097 - Prova 2

19 de Outubro de 2016

Servidor BOCA:
`http://maratona.c3sl.ufpr.br/`

Instruções Importantes

- A prova é composta por 5 problemas que valem 20 pontos cada, e um problema bônus que também vale 20 pontos.
- O problema F é um problema bônus, usado em uma seletiva mas que não foi resolvido. Este problema só deve ser considerado caso todos os demais forem resolvidos.
- Se sua solução para um problema for aceita, a nota integral do problema é garantida.
- Se sua solução não for aceita, o professor irá avaliar sua última (mas **apenas** a última) submissão para o problema e atribuir a ela uma nota parcial, considerando a corretude do algoritmo utilizado, a complexidade computacional do algoritmo, o uso correto de funções e estruturas de dados disponíveis utilizadas, e eventuais detalhes da solução.
- Em cada problema, cada arquivo de entrada contém *apenas um* caso de teste. Sua solução será executada com vários arquivos de entrada separadamente.
- Sua solução será compilada ou executada com a seguinte linha de comando:
  - C: `gcc -static -O2 -lm`
  - C++: `g++ -static -O2 -lm`
  - C++11: `g++ -std=c++11 -static -O2 -lm`
  - Java: `javac`
  - Pascal: `fpc -Xt -XS -O2`
  - Python: `python3`
- O tempo limite para a execução de sua solução é de 1 segundo para todos os seis problemas da prova.
- Todas as linhas, tanto na entrada quanto na saída, terminam com o caractere de fim-de-linha ($\backslash n$), mesmo quando houver apenas uma única linha no arquivo.
- Para submissões em **JAVA**, a classe deverá ter o mesmo nome que o *basename* do problema (leia a linha entre o título e o texto do problema).

A: Viagem para BH

`Arquivo: viagem.[c|cpp|java|pas|py]`

Está chegando a grande final do Campeonato Nlogonense de Surf Aquático, que este ano ocorrerá na cidade de Bonita Horeleninha (BH)! Você decidiu que irá viajar de sua cidade natal para BH para acompanhar a final.

Existem N cidades em Nlogônia, numeradas de 1 a N. Considere que cidade 1 é sua cidade natal, e a cidade N é BH.

Além disso, existem M trechos pelos quais é possível viajar. Cada trecho pode ser usado para ir de uma cidade para alguma outra do país. Alguns trechos são feitos de ônibus, enquanto os demais são feitos de avião. Para cada trecho, você conhece o preço, em reais, da passagem que deve pagar para poder utilizá-lo.

Para não tornar sua viagem muito cansativa com deslocamentos entre rodoviárias e aeroportos, você decidiu que irá utilizar *apenas um* meio de transporte em *toda* sua viagem, isto é, você quer ir para BH ou utilizando apenas ônibus, ou utilizando apenas aviões.

Sua tarefa é determinar o custo mínimo necessário, em reais, para viajar da sua cidade natal para BH, dada a restrição que o meio de transporte não deve ser alterado durante a viagem.

Entrada

A primeira linha contém dois inteiros N e M ($2 \leq N \leq 100, 1 \leq M \leq 2(N^2 - N)$), o número de cidades e de trechos, respectivamente. As próximas M linhas descreve um trecho cada. Cada linha contém quatro inteiros $A\ B\ T\ R$ ($1 \leq A, B \leq N, A \neq B, T = 0$ ou $1, 1 \leq R \leq 10^4$), indicando um trecho que sai da cidade A e chega na cidade B (nesta ordem), feito por ônibus se $T = 0$ ou por avião se $T = 1$, e cuja passagem custa R reais.

É garantido que existe ao menos um caminho de sua cidade para BH utilizando apenas um meio de transporte. Além disso, para cada par ordenado de cidades (A, B), existe no máximo um trecho de A para B para cada meio de transporte possível (mas note que pode haver um trecho de ônibus e outro de avião de A para B).

Saída

Imprima uma única linha contendo um inteiro indicando o custo mínimo necessário para fazer sua viagem, dadas as restrições acima.

Exemplo de entrada	**Exemplo de saída**
5 6 1 2 0 200 1 3 1 400 2 4 0 300 3 4 1 300 2 5 0 700 4 5 1 100	800

B: Pizza antes de BH

`Arquivo: pizza.[c|cpp|java|pas|py]`

Está chegando a grande final do Campeonato Nlogonense de Surf Aquático, que este ano ocorrerá na cidade de Bonita Horeleninha (BH)! Antes de viajar para BH, você e seus $N-1$ amigos decidiram combinar algum dia para ir a uma pizzaria, para relaxar e descontrair (e, naturalmente, comer!).

Neste momento está sendo escolhida a data do evento. Para que todas as peossas possam participar, foi decidido que o encontro na pizzaria ocorrerá em um data tal que *todas* as N pessoas podem comparecer à pizzaria nesta data. Portanto, nem toda data pode ser escolhida, pois algumas pessoas podem ter outros compromissos já marcados em alguns dias.

Dada a lista de datas consideradas para o evento e a informações de quais pessoas podem comparecer em quais datas, determine se o evento poderá ocorrer e, em caso positivo, sua data. Caso mais de uma data seja possível, o evento deve ocorrer o mais cedo possível.

Entrada

A primeira linha contém os inteiros N e D ($1 \leq N, D \leq 50$), o número de pessoas e o número de datas consideradas, respectivamente. As pessoas são numeradas de 1 a N. As próximas D linhas descrevem uma data considerada. Cada linha começa com a data na forma $dia/mes/ano$. A linha é seguida de N inteiros $p_1, p_2, ..., p_N$. O inteiro p_i é 1 se a pessoa i pode comparecer na data considerada, ou 0 caso contrário. É garantido que as datas são sempre válidas, e não há zeros à esquerda. Além disso, as datas são dadas em ordem, do dia mais cedo para o dia mais tarde.

Saída

Imprima uma linha contendo a data que o evento deve ocorrer, na forma $dia/mes/ano$, de maneira idêntica à da entrada. Caso não seja possível realizar o evento, imprima "`Pizza antes de FdI`" (sem aspas).

Exemplo de entrada	**Exemplo de saída**

`4 4`
`1/6/2016 0 0 1 0`
`12/7/2016 1 1 1 0`
`5/10/2016 1 1 1 1`
`25/12/2016 0 0 0 0`
	`5/10/2016`

Exemplo de entrada	**Exemplo de saída**

`5 3`
`20/9/2016 0 1 1 1 1`
`30/9/2016 1 0 1 1 1`
`1/10/2016 1 1 0 1 1`
	`Pizza antes de FdI`

C: Paodequeijosweeper

`Arquivo: paodequeijo.[c|cpp|java|pas|py]`

Está chegando a grande final do Campeonato Nlogonense de Surf Aquático, que este ano ocorrerá na cidade de Bonita Horeleninha (BH)! Nesta cidade, o jogo *PãodequeijoSweeper* é bastante popular!

O tabuleiro do jogo consiste em uma matriz de N linhas e M colunas. Cada célula da matriz contém um pão de queijo ou o número de pães de queijo que existem nas celulas adjacentes a ela. Uma célula é adjacente a outra se estiver imediatamente à esquerda, à direita, acima ou abaixo da célula. Note que, se não contiver um pão de queijo, uma célula deve obrigatoriamente conter um número entre 0 e 4, inclusive.

Dadas as posições dos pães de queijo, determine o tabuleiro do jogo!

Entrada

A primeira linha contém os inteiros N e M ($1 \leq N, M \leq 100$). As próximas N linhas contém M inteiros cada, separados por espaços, descrevendo os pães de queijo no tabuleiro. O $j-$ésimo inteiro da $i-$ésima linha é 1 se existe um pão de queijo na linha i e coluna j do tabuleiro, ou 0 caso contrário.

Saída

Imprima N linhas com M inteiros cada, *não separados por espaços*, descrevendo a configuração do tabuleiro. Se uma posição contém um pão de queijo, imprima `9` para ela; caso contrário, imprima o número cuja posição deve conter.

Exemplo de entrada	**Exemplo de saída**
4 4 0 0 1 1 0 1 0 1 0 0 1 0 1 1 0 1	0299 1949 1393 9939

Exemplo de entrada	**Exemplo de saída**
1 2 0 1	19

D: Passeio em FdI

`Arquivo: passeio.[c|cpp|java|pas|py]`

Está chegando a grande final do Campeonato Nlogonense de Surf Aquático. Ano que vem, a final ocorrerá na cidade de Foça do Iguachim (FdI)! A cidade é famosa por conter o Parque Nacional do Iguachim, que conta com várias atrações. Dentre elas, destacam-se as Cataratas do Iguachim, um dos pontos turisticos mais famosos de Nlogonia!

O Parque conta com N atrações, numeradas de 1 a N. As atrações são dispostas em uma linha reta no Parque. Desta forma, o Parque pode ser descrito como uma rua contendo entradas para as atrações 1, 2, ..., N, onde a atração 1 é a mais próxima da entrada do Parque, enquanto a atração N é a mais próxima da saída do Parque. Para não tumultoar o Parque, é exigido que as atrações sejam visitadas *em ordem* da entrada para a saída, isto é, se você visitar a atração i, você não pode voltar e visitar as atrações 1, 2, ..., $i - 1$.

Além disso, existem dois tipos de *tickets* no Parque: os *tickets* verdes e os *tickets* amarelos. Cada uma das N atrações exigem, como pagamento por sua entrada, uma certa quantidade de *tickets* de exatamente um tipo. Ao entrar em uma atração, o Parque pode lhe presentear com uma certa quantidade de *tickets* do outro tipo, isto é, uma atração que cobra *tickets* verdes como entrada pode lhe dar *tickets* amarelos como presente, ou vice-versa. Você pode não poder entrar em uma atração se não tiver *tickets* suficientes para ela, mas também pode optar não entrar nela se quiser, mesmo se tiver *tickets* suficientes.

Entretanto, você quer aproveitar o Parque o máximo possível! Dada a quantidade inicial de *tickets* de cada tipo que você possui e a descrição das atrações do Parque, determine o número máximo de atrações que podem ser visitadas.

Entrada

A primeira linha contém o inteiro N ($1 \leq N \leq 40$). A segunda linha contém dois inteiros V e A ($0 \leq V, A \leq 20$), o número de *tickets* verdes e amarelos que você possui inicialmente. As próximas N linhas descrevem as atrações do Parque, na ordem da entrada para a saída do Parque. Cada linha contém dois inteiros V_i e A_i ($-20 \leq V_i, A_i \leq 20$, $V_i \times A_i < 0$). Se $V_i < 0$, a atração cobra $|V_i|$ *tickets* verdes como entrada, e, se visitada, lhe presenteia com A_i *tickets* amarelos. Caso contrário, ela cobra $|A_i|$ *tickets* amarelos, e, se visitada, lhe presenteia com V_i *tickets* verdes.

Saída

Imprima uma linha contendo a quantidade máxima de atrações que podem ser visitadas.

Exemplo de entrada	**Exemplo de saída**
3 10 0 -10 5 -5 4 20 -5	2

E: Compras em FdI

`Arquivo: compras.[c|cpp|java|pas|py]`

Está chegando a grande final do Campeonato Nlogonense de Surf Aquático. Ano que vem, a final ocorrerá na cidade de Foça do Iguachim (FdI)! A região de FdI e das cidades próximas é famosa por seu comércio, composto por diversas lojas que costumam vender diversos produtos a preços mais atraentes que no restante do país. Você quer aproveitar a viagem para FdI para comprar o novo celular *Aifôni (R)*![1]

Existem N lojas na região, numeradas de 1 a N. Todas as lojas vendem o celular, embora o preço do aparelho pode ser diferente em cada loja. Para não tornar sua viagem cansativa, você pode considerar não visitar todas as N lojas, mas sim visitar apenas as lojas entre duas dadas lojas i e j, inclusive. Você está interessado na *maior diferença de preços* do aparelho entre as lojas visitadas. A diferença é dada por $|M - m|$, onde M é o maior preço dentre as lojas visitadas, e m é o menor.

Além disso, as lojas podem alterar o preço do celular como desejarem! Sua tarefa é determinar, *para várias consultas*, a maior diferença de preços nas lojas entre duas dadas lojas, considerando também eventuais alterações de preços nas lojas.

Entrada

A primeira linha contém o inteiro N ($1 \leq N \leq 10^5$). A segunda linha contém N inteiros $p_1, p_2, ..., p_N$ ($1 \leq p_i \leq 10^5$). O inteiro p_i indica o preço inicial do celular na loja i. A terceira linha contém um inteiro Q ($1 \leq Q \leq 10^5$), o número de operações. As próximas Q linhas descrevem uma operação cada. Cada operação pode ser descrita de duas formas:

- 1 i p ($1 \leq i \leq N, 1 \leq p \leq 10^5$), indicando que o preço do celular foi alterado para p na loja i.
- 2 i j ($1 \leq i \leq j \leq N$), indicando uma consulta.

Saída

Para cada consulta, imprima uma linha contendo a maior diferença de preços das lojas entre as lojas i e j, inclusive.

Exemplo de entrada	**Exemplo de saída**
4 100 150 90 170 3 2 1 3 1 2 50 2 2 4	60 120

[1] Na verdade, você queria um *Sãosunga (R)*, mas este celular é um verdadeiro estouro!

F: Desafio das Moedas Prateadas

`Arquivo: desafio.[c|cpp|java|pas|py]`

O *Desafio das Moedas Prateadas* é um esporte individual popular no reino de Diddykongolândia. O campo e as regras do jogo são descritos a seguir.

O campo do jogo consiste em *locais* e *trechos*. Há N locais no campo. Um desses locais é o local de *largada*, no qual o jogador inicia o jogo.

Há M trechos no campo. Cada trecho é uma via unidirecional que liga um local do campo a outro local distinto. Os trechos são os únicos meios pelos quais o jogador pode se mover entre os locais do campo. Os trechos do campo são dados de tal forma que:

- é possível ir do local de largada a qualquer outro local usando os trechos;
- é possível ir de qualquer local do campo ao local de largada usando os trechos;
- é impossível sair de um local l e voltar para o mesmo local l sem passar pelo local de largada.

Há também K *moedas prateadas* no jogo. Cada moeda está em um local distinto do campo. Se o jogador chegar a um local que contém uma moeda, o jogador pode coletá-la. Não há moeda no local de largada.

O jogador começa o jogo no local de largada. Uma *volta* é completada pelo jogador quando ele retorna ao local de largada após passar por outro(s) local(is).

O jogador deve completar exatamente três voltas. Se o jogador conseguir coletar todas as moedas de prata antes de completar a última volta, ele vence. Caso contrário, ele perde.

Após analizar o campo de jogo, você descobriu quanto tempo leva para atravessar cada trecho. Agora, você deve descobrir se é possível vencer no campo dado e, em caso positivo, qual o tempo mínimo que você deve levar para vencer. Considere instantâneo o tempo para coletar uma moeda e para atravessar um local (sair de um trecho e entrar em outro adjacente).

Entrada

A entrada inicia com uma linha contendo três inteiros N, M e K ($2 \leq N \leq 1000, 2 \leq M \leq (N^2 + N - 2)/2, 1 \leq K \leq min\{12, N - 1\}$), indicando o número de locais, de trechos e de moedas no campo. Os locais são numerados de 1 a N. O local 1 é o local de largada.

As próximas M linhas descrevem os trechos. Cada trecho é descrito por três inteiros l_a, l_b e t ($1 \leq l_a, l_b \leq N, l_a \neq l_b, 1 \leq t \leq 10^4$), indicando que há um trecho que leva do local l_a para o local l_b que é atravessado em t segundos.

A última linha contém K inteiros distintos k_i ($2 \leq k_i \leq N$ para $1 \leq i \leq K$) indicando os locais das moedas prateadas.

Saída

Se não é possível vencer o jogo, imprima uma linha contendo "`impossivel`" (sem aspas). Caso contrário, imprima uma linha contendo o menor tempo necessário, em segundos, para vencer o jogo.

Exemplo de entrada	**Exemplo de saída**
4 6 3 1 2 1 1 3 1 1 4 1 2 1 1 3 1 1 4 1 1 2 3 4	6

Exemplo de entrada	**Exemplo de saída**
5 7 2 1 2 1 1 3 2 2 3 3 3 4 7 3 5 3 4 5 2 5 1 4 2 4	35

Exemplo de entrada	**Exemplo de saída**
5 8 4 1 2 3 1 3 2 1 4 10 1 5 7 2 1 5 3 1 3 4 1 1 5 1 12 4 5 3 2	impossivel