CI097 - Prova 1

5 de Setembro 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 na 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 os problemas A, B, C, D e E, e de 2 segundos para o problema F.
- 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: Transfira para o Professor

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

Analógimôn Go é um jogo bastante popular. Em sua jornada, o jogador percorre diversas cidades capturando pequenos monstrinhos virtuais, chamados *analógimôns*.

Você é um experiente jogador e já capturou N analógimôns, numerados de 1 a N. Você capturou tantos monstrinhos que já está difícil levar todos com você em sua jornada. Por isso, você pode se livrar de alguns de seus monstrinhos transferindo-os para o Professor.

Ao transferir o analógimôn i (para $1 \leq i \leq N$) para o Professor, você ganha D_i doces do Professor em troca do monstrinho. Como os doces são itens muito importantes no jogo, você quer transferir quais e quantos analógimôns forem necessários para ter a maior quantidade possível de doces!

Entretanto, o analógimôn i (para $1 \leq i \leq N$) pesa P_i kg, e, devido a uma limitação de espaço no laboratório do Professor, ele não pode receber analógimôns cuja soma total dos pesos é maior que K kg.

Sua tarefa é determinar a quantidade máxima de doces que você pode obter transferindo seus monstrinhos, respeitando a limitação de espaço do laboratório do Professor.

Entrada

A primeira linha contém os inteiros N e K ($1 \leq N \leq 100, 1 \leq K \leq 10^4$), o número de analógimôns que você capturou e a capacidade do laboratório do Professor, em kg, respectivamente. A segunda linha contém N inteiros D_1, ..., D_N ($1 \leq D_i \leq 10^4$), indicando quantos doces você ganhará pela transferência de cada analógimôn. A terceira linha contém N inteiros P_1, ..., P_N ($1 \leq P_i \leq 10^4$) indicando o peso de cada analógimôn, em kg.

Saída

Imprima uma única linha contendo a quantidade máxima de doces que você pode obter.

Exemplo de entrada	**Exemplo de saída**

`4 52`
`1 8 14 22`
`4 12 20 30`
	`36`

Exemplo de entrada	**Exemplo de saída**

`3 2`
`9 5 2`
`12 8 42`
	`0`

B: Impeachment do Líder

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

Analógimôn Go é um jogo bastante popular. Os jogadores de *Analógimôn Go* são divididos em três grandes times: Time *Valor*, Time *Instinto* e Time *Místico*, que são liderados pelos seus líderes Kandera, Esparky e Blanque, respectivamente. Naturalmente, você faz parte de um desses times!

O líder do seu time está sendo acusado de infringir as regras do jogo por gerenciar incorretamente os doces recebidos do Professor que são destinados ao time. Isto criou uma grande polêmica dentro da equipe: alguns jogadores defendem que o líder realmente agiu incorretamente e deve sofrer um *impeachment* e ser afastado de seu cargo, enquanto outros defendem que ele não infringiu as regras, que a acusação é inverídica e que ele deve continuar no cargo.

Para resolver a situação, uma votação será realizada entre todos os N jogadores do seu time. Cada jogador deverá votar se o *impeachment* deve ou não ocorrer. Se o número de votos favoráveis ao *impeachment* foi maior ou igual a 2/3 (dois terços) do total de jogadores, o líder será afastado. Caso contrário, a acusação é arquivada e ele continuará no cargo.

Dados os votos de todos os jogadores, determine o resultado da votação.

Entrada

A primeira linha contém o inteiro N ($1 \leq N \leq 10^5$), o número de jogadores em seu time. A próxima linha contém N inteiros v_1, ..., v_N ($v_i = 0$ ou 1), indicando os votos dos jogadores. O valor 1 indica um voto favorável ao *impeachment*, enquanto o valor 0 indica um voto contrário ao mesmo.

Saída

Imprima uma linha contendo a palavra `impeachment` se o líder deve ser afastado de seu cargo, ou `acusacao arquivada` caso contrário.

Exemplo de entrada	**Exemplo de saída**
`6` `1 0 1 1 0 1`	`impeachment`

Exemplo de entrada	**Exemplo de saída**
`5` `0 1 1 1 0`	`acusacao arquivada`

C: Ginásio

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

Analógimôn Go é um jogo bastante popular. Em sua jornada, o jogador percorre diversas cidades capturando pequenos monstrinhos virtuais, chamados *analógimôns*. As cidades contém localidades especiais chamadas de *ginásios*. Ao chegar a um ginásio, um jogador pode *tentar* colocar um de seus analógimôns dentro dele.

Cada anlógimôn tem dois inteiros associados a ele: seu *Poder de Combate* (PC) e seu *Número de Ataques* (NA). Além disso, um ginásio tem associado a ele um *Intervalo de Poder* (IP). Ao tentar colocar um analógimôn em um ginásio, o jogo verifica quantos são os analógimôns já presentes no ginásio cuja diferença do seu PC para o PC do analógimôn sendo colocado é de no máximo IP. Se esta quantidade for menor ou igual ao NA do analógimôn sendo colocado, o monstrinho é inserido no ginásio com sucesso. Caso contrário, ele não é colocado no ginásio. Em ambos os casos, os analógimôns que já estavam no ginásio continuam no ginásio. Como exemplo, considere um ginásio com IP=3 com analógimôns de PC iguais a 5, 8, 13 e 20. Se um jogador tenta colocar um analógimôn de PC=10 e NA=4, o jogo contará quantos analógimôns há no ginásio com PC entre $10 - 3 = 7$ e $10 + 3 = 13$, inclusive. Como há dois analógimôns neste caso, o monstrinho é colocado com sucesso no ginásio, pois $2 \leq 4$. O ginásio passa a conter analógimôns de PC iguais a 5, 8, 10, 13 e 20.

Dadas as informações sobre um ginásio e as tentativas de colocar analógimôns dentro dele, determine quantos analógimôns ficarão no ginásio após todas as tentativas. Considere que o ginásio inicialmente não contém nenhum analógimôn.

Entrada

A primeira linha contém os inteiros IP e M ($1 \leq IP, M \leq 10^5$), o IP do ginásio e o número de tentativas, respectivamente. As próximas M linhas descrevem as tentativas de colocar um analógimôn no ginásio, na ordem em que são feitas. Cada linha contém dois inteiros PC e NA ($1 \leq PC, NA \leq 10^5$), indicando o PC e o NA do analógimôn, respectivamente. O Poder de Combate de todos analógimôns são distintos.

Saída

Imprima uma linha com um inteiro indicando quantos analógimôns ficarão no ginásio.

Exemplo de entrada	**Exemplo de saída**
3 7 5 2 13 1 8 1 20 5 6 1 11 1 10 4	5

D: Tipos de Espécies

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

Analógimôn Go é um jogo bastante popular. Em sua jornada, o jogador percorre diversas cidades capturando pequenos monstrinhos virtuais, chamados *analógimôns*. Existem várias espécies de analógimôns. Cada espécie é de (exatamente) um *tipo*, como *fogo*, *água*, *elétrico*, etc. Algumas espécies podem ser do mesmo tipo, enquanto outras podem se tipos diferentes.

No manual oficial do jogo consta que algumas espécies são do mesmo tipo. Entretanto, o manual pode não apresentar esta informação para todos os pares de espécies que são do mesmo tipo. Por exemplo, se o manual indica que uma espécie a é do mesmo tipo que uma espécie b, e que uma espécie b é do mesmo tipo que uma espécie c, então as espécies a e c certamente são do mesmo tipo, embora esta informação pode não constar no manual.

Você capturou um analógimôn de uma certa espécie. Sua tarefa é determinar o menor número possível de espécies que certamente são do mesmo tipo da espécie do seu analógimôn, de acordo com as informações contidas no manual.

Entrada

A primeira linha contém dois inteiros N e M ($1 \leq N \leq 1000, 0 \leq M \leq \frac{N \times (N-1)}{2}$), o número de espécies de analógimôns e o número de informações presentes no manual, respectivamente. As espécies são numeradas de 1 a N. Cada uma das próximas M linhas contém uma informação presente no manual. Cada linha contém dois inteiros a e b ($1 \leq a, b \leq N, a \neq b$), indicando que as espécies a e b são do mesmo tipo. A última linha contém um inteiro E ($1 \leq E \leq N$), indicando a espécie de seu analógimôn.

Saída

Imprima uma linha com um inteiro indicando a menor quantidade de espécies de analógimôns que certamente são do mesmo tipo da espécie do seu analógimôn, de acordo com o manual. Note que a espécie do seu analógimôn também deve ser contada.

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

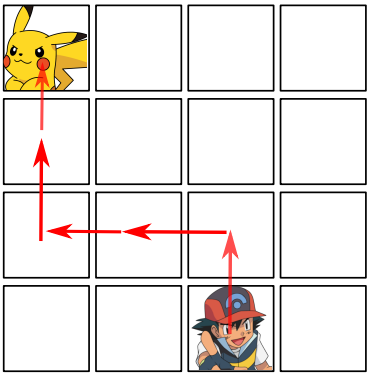
Exemplo de entrada	**Exemplo de saída**
3 1 1 2 3	1

E: O Último Analógimôn

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

Analógimôn Go é um jogo bastante popular. Em sua jornada, o jogador percorre diversas cidades capturando pequenos monstrinhos virtuais, chamados *analógimôns*. Você acabou de chegar em uma cidade que contém o último analógimôn que falta para sua coleção!

A cidade pode ser descrita como um *grid* de N linhas e M colunas. Você está em uma dada posição da cidade, enquanto o último analógimôn está em outra posição da mesma cidade. A cada segundo, você pode se mover (exatamente) uma posição ao norte, ao sul, a leste ou a oeste. Considerando que o analógimôn não se move, sua tarefa é determinar o menor tempo necessário para ir até a posição do monstrinho. A figura descreve o exemplo da entrada, e apresenta um caminho percorrido em 5 segundos. Outros caminhos percorridos no mesmo tempo são possíveis, mas não há outro caminho que pode ser percorrido em um tempo menor.



Entrada

A primeira linha contém dois inteiros N e M ($2 \leq N, M \leq 100$), o número de linhas e de colunas na cidade, respectivamente. As próximas N linhas contém M inteiros cada, descrevendo a cidade. O inteiro 0 indica uma posição em branco; o inteiro 1 indica a sua posição na cidade; o inteiro 2 indica a posição do analógimôn na cidade. É garantido que haverá exatamente um inteiro 1 e exatamente um inteiro 2 na descrição da cidade, e que os demais inteiros serão iguais a 0.

Saída

Imprima uma linha contendo o menor tempo necessário para ir até o monstrinho, em segundos.

Exemplo de entrada	**Exemplo de saída**
4 4 2 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0	5