Seletiva para Maratona de Programação de 2013

Instituto de Matemática e Estatística

Universidade de São Paulo

Caderno de Problemas

Departamento de Ciência da Computação IME-USP

Sábado, 17 de agosto de 2013.

Problema A: Popularidade no Facebook

Arquivo: `facebook.[c|cpp|java]`

Hoje em dia todos estão conectados, participam do Facebook, publicam suas fotos no Instagram, seus vídeos no Youtube, e assim por diante. Até mesmo sistemas como GPS hoje se baseiam em redes sociais, tornando tudo mais divertido (e talvez mais difícil de entender, mas isso é outra conversa). Ser popular no Facebook é quase uma necessidade. Uma pessoa com menos de 700, 800 amigos pode ser considerado quase como um pária nessa nova realidade.

Talvez por isso algumas pessoas costumam exagerar quando dizem o número de amigos que possuem. Considere uma comunidade com N pessoas, e para cada uma delas, considere que sabemos o número de amigos que cada pessoa diz ter na comunidade. Sua tarefa neste problema é determinar se de fato é possível que todos os membros da comunidade estejam falando a verdade. Lembre que uma pessoa não pode ser amiga de si mesma, e duas pessoas não podem ser amigas várias vezes.

Entrada

A entrada é composta por diversas instâncias e termina com final de arquivo (`EOF`).

A primeira linha de cada instância contém um inteiro N. A segunda linha possui N inteiros, a_i, separados por um espaço em branco, correspondendo ao número de amigos que a pessoa i diz ter na comunidade.

A entrada deve ser lida da entrada padrão.

Saída

Para cada instância imprima, em uma única linha, `possivel` se é possível que todos os membros da comunidade estejam falando a verdade, ou `impossivel` caso contrário.

A saída deve ser escrita na saída padrão.

Restrições

- $1 \leq N \leq 10^5$
- $0 \leq a_i \leq 10^5$

Exemplos

Exemplo de entrada	**Saída para o exemplo de entrada**

`3`
`1 1 1`
`3`
`2 2 2`
	`impossivel`
	`possivel`

Problema B: Duelo de espiões

Arquivo: `duelo.[c|cpp|java]`

Alexey e Boris eram dois agentes da KGB que moravam em Ecaterimburgo nos anos 70. A cidade era um tanto parada, e, como nada acontecia, para não morrerem em tédio os dois agentes decidiram inventar um jogo de dados. Nele cada um deles começa com V_A e V_B pontos de vida respectivamente. Cada um têm a sua disposição um número de ataques possíveis, e eles se alternam atacando um ao outro. Cada ataque é descrito por uma quantidade de dados. Para saber o dano do ataque rodamos essa quantidade de dados e a soma dos valores é o dano causado.

Para jogar, eles têm disponível dados honestos com número de faces entre 1 e 12. Isso é, se um dado com L faces for jogado ele vai mostrar um valor inteiro entre 1 e L com igual probabilidade e de maneira independente de qualquer outro lançamento no jogo.

Ambos os jogadores conhecem todos os seus ataques e os do seu oponente e escolhem como atacar em cada turno de forma a maximizar a sua própria probabilidade de vitória.

Sua tarefa nesse problema é determinar qual a probabilidade de vitória de cada jogador.

Entrada

A entrada é composta por diversas instâncias e termina com final de arquivo (`EOF`).

A primeira linha de cada instância contém quatro inteiros, V_A, V_B, N_A e N_B. Cada uma das próximas N_A linhas descrevem um ataque do Alexey, elas começam com um inteiro D e são seguidas por outros D inteiros $L_1, \ldots, L_D$, indicando que nesse ataque Alexey lança D dados, com $L_1, L_2, \ldots, L_D$ faces. As próximas N_B linhas descrevem os ataques do Boris de maneira análoga.

A entrada deve ser lida da entrada padrão.

Saída

Para cada instância, imprima uma linha com um único ponto flutuante arredondado para 3 casas decimais, indicando a probabilidade que o Alexey vença o duelo, sendo que ele que começa atacando.

A saída deve ser escrita na saída padrão.

Restrições

- $1 \leq V_A, V_B \leq 300$
- $1 \leq N_A, N_B \leq 10$
- $1 \leq D \leq 3$
- $1 \leq L_i \leq 12$

Exemplos

Exemplo de entrada	Saída para o exemplo de entrada
2 12 2 1 1 12 3 4 4 5 2 1 1 5 5 1 2 1 6 2 3 5 2 1 6	0.083 0.534

Problema C: As pirâmides de Ecaterimburgo

Arquivo: `piramides.[c|cpp|java]`

As pirâmides são estruturas muito frequentes em civilizações antigas de todo o mundo. As mais famosas, as do Egito, foram construídas mais de 2000 anos antes da passagem de Cristo pela Terra. Outras pirâmides famosas são as encontradas no México e América Central, ligadas às civilizações astecas e maias.

Poucos conhecidas, porém, são as pirâmides construídas nos montes Urais, perto da cidade de Ecaterimburgo. Tais pirâmides remontam ao início da era cristã, e acredita-se que foram construídas por povos mongois, que invadiram a Europa através dos montes Urais vindos da Ásia. Diferentemente das outras pirâmides conhecidas, estas tinham base triangular. Dessa forma tais pirâmides tinham 4 faces triangulares. Muitas dessas pirâmides eram construídas sobre colunas, o que permitia construções não paralelas ao solo e até mesmo que a base "de baixo" ficasse exposta.

Isso era muito importante, pois as faces da pirâmide eram pintadas com figuras que representavam deuses antigos, figuras mitológicas, planetas e outros corpos celestes, e assim por diante. Dessa forma, em algum ponto da cidade, um cidadão poderia visualizar uma ou mais das faces da pirâmide. Isso era importante na religião local, e encontrar uma casa de cuja janela se vislumbrasse as melhores faces das pirâmides era muito valorizado naquela época.

Sua tarefa é, dadas as posições no espaço dos vértices de uma pirâmide, e a posição no espaço de um observador, determinar quais das faces da pirâmide são visíveis a ele, considerando que não existe nenhum obstáculo entre o observador e a pirâmide.

Entrada

A entrada é composta por diversas instâncias e termina com final de arquivo (`EOF`).

Cada instância consiste de 5 linhas. Cada linha contém três inteiros, separados por espaços e representam as coordenadas dos pontos A, B, C, D e X respectivamente, onde X é a posição do observador e os demais pontos são vértices da pirâmide.

A entrada deve ser lida da entrada padrão.

Saída

Para cada instância, imprima uma linha contendo 4 caracteres. O primeiro caractere deve ser `S` se o observador enxerga a face da pirâmide oposta ao vértice A e `N` caso contrário. Analogamente para o segundo, terceiro e quarto caracteres, mas levando em conta as faces opostas aos vértices B, C e D respectivamente.

A saída deve ser escrita na saída padrão.

Restrições

- Todos os pontos fornecidos têm coordenadas inteiras entre -100 e 100.
- Os pontos A, B, C e D não são coplanares.
- Todos os pontos fornecidos são distintos.
- O ponto X não pertence nem ao interior nem a nenhuma face da pirâmide.

Exemplos

Exemplo de entrada	Saída para o exemplo de entrada
0 0 0 1 0 0 0 1 0 0 0 1 1 1 1 0 0 0 1 0 0 0 1 0 0 0 1 -1 -1 -1 0 0 0 1 0 0 0 1 0 0 0 1 1 1 0	SNNN NSSS SNNN

Problema E[1] : Poker

Arquivo: `poker.[c|cpp|java]`

Poker é jogado com um baralho tradicional de 52 cartas (13 valores com 4 naipes). Os valores das cartas, em ordem crescente, são: `2`, `3`,. . ., `10`, `Valete`, `Dama`, `Rei`, `Ás`. Dada uma mesa de poker, com dois jogadores, sua tarefa é determinar quem ganhou.

Cada jogador possui duas cartas próprias e há cinco cartas comuns aos dois jogadores na mesa. Ganha quem conseguir uma mão de cinco cartas mais valiosa, entre as suas duas cartas próprias e as cinco cartas da mesa. Uma carta da mesa pode ser usada pelos dois jogadores ao mesmo tempo e a mão mais valiosa de um ou dos dois jogadores pode ser obtida ignorando as suas duas cartas próprias e usando as cinco comuns.

Para comparar duas mãos de cinco cartas, verifica-se em quais tipos da lista abaixo elas se encaixam. Se uma mão se encaixar em mais de um tipo, escolhe-se o mais valioso. Se as duas mãos se encaixarem num mesmo tipo, se aplica uma regra de desempate específica para este tipo.

A lista de tipos de mãos, ordenados do menos valioso para o mais valioso, e seus respectivos critérios de desempate é:

- **Carta mais alta**: qualquer mão que não se enquadre em nenhum dos demais tipos. No desempate, as cinco cartas são comparadas uma a uma, da mais valiosa para a menos, até uma mão apresentar uma carta com valor maior que o da outra.
- **Um par**: duas cartas de mesmo valor. O desempate é análogo ao da carta mais alta comparando primeiro o valor do par e depois as demais cartas;
- **Dois pares**: dois pares. O desempate é análogo ao da carta mais alta comparando primeiro o valor do par mais valioso, depois o valor do par menos valioso e por fim a carta restante;
- **Trinca**: três cartas de mesmo valor. O desempate é análogo ao do par;
- **Straight**: sequência de cinco cartas de valores consecutivos. Neste caso o `Ás` pode tomar o valor tanto de carta mais baixa (antes do `2`) ou de mais alta (depois do `Rei`). O desempate é feito pela carta de maior valor, sendo que excepcionalmente o `Ás` tem o menor valor se aparecer antes do `2`;
- **Flush**: cinco cartas do mesmo naipe. O desempate é feito pelo critério da carta mais alta;
- **Full House**: uma trinca e um par. No desempate é comparado primeiro o valor da trinca. Persistindo o empate, é comparado o valor do par;
- **Quadra**: quatro cartas com um mesmo valor. No desempate compara-se o valor da quadra e depois a carta restante;
- **Straight Flush**: *straight* e *flush* simultâneamente. O desempate é feito como no *straight*.

Note que é possível persistir o empate mesmo depois de aplicadas as regras de desempate. Os naipes das cartas só são considerados para definir um *flush*, não sendo considerados em nenhuma regra de desempate.

Entrada

A entrada é composta por diversas instâncias e termina com final de arquivo (`EOF`).

Cada instância é composta por três linhas. As duas primeiras linhas contêm a descrição de duas cartas cada uma, separadas por espaço. A primeira linha corresponde às cartas do primeiro jogador enquanto que a segunda corresponde às cartas do segundo jogador. A terceira linha contém a descrição das cinco cartas na mesa, também separadas por espaço.

Caractere	Carta
'2' - '9'	`2 - 9`
'T'	`10`
'J'	`Valete`
'Q'	`Dama`
'K'	`Rei`
'A'	`Ás`
'e'	Espadas
'c'	Copas
'o'	Ouros
'p'	Paus

Tabela 1: Codificação das cartas do baralho

A descrição de uma carta é dada por 2 caracteres, o primeiro indica o valor da carta e o segundo o seu naipe, conforme a tabela abaixo:

A entrada deve ser lida da entrada padrão.

Saída

Para cada instância, imprima uma linha contendo um número inteiro. Imprima 1 se o primeiro jogador ganha essa instância, imprima 2 caso o segundo ganhe e imprima 0 se houver um empate, mesmo depois de aplicadas as regras de desempate.

A saída deve ser escrita na saída padrão.

Restrições

- Todas as cartas fornecidas serão válidas e distintas.

Exemplos

Exemplo de entrada	**Saída para o exemplo de entrada**

`Te Je`
`Tp Jp`
`Qe Qp Ke Kp Ae`
`Ae 7o`
`Ac 8e`
`Ap Ao 9e Jc Kp`
`Ae 7o`
`Ac 8e`
`Ap Ao 6e 3c Kp`
	`1`
	`0`
	`2`

[1]Se estiver sentindo falta de algo, deve ser o problema D. Ele não existe!

Problema F: Votação em Ecaterimburgo

Arquivo: `votacao.[c|cpp|java]`

Ecaterimburgo, Rússia, é uma cidade com um curioso sistema de votação. Em uma eleição em que haja V vagas para um cargo, cada eleitor tem direito a fazer V votos, ordenados em sua ordem de preferência. Assim, se, por exemplo, há 3 vagas de senador, cada eleitor vota em até 3 nomes. Serão eleitos os candidatos que tiverem o maior número de votos, sem importar em que posição da preferência do eleitor está o candidato. Apenas quando há empate no número de votos se torna relevante a ordem dada pelos eleitores. Ganha aquele candidato que tiver mais indicações em primeiro lugar. Se persistir o empate, em segundo lugar, e assim por diante. Caso dois ou mais candidatos que estejam em posição de serem eleitos tenham exatamente o mesmo número de indicações em todas as posições, todos são eleitos (podendo inclusive exceder o número de vagas). Candidatos com zero votos podem ser eleitos se ainda existir vagas disponíveis.

Entrada

A entrada é composta por diversas instâncias e termina com final de arquivo (`EOF`).

Cada instância começa com o número N de eleitores, o número K de candidatos e V de vagas. A seguir vêm N linhas com os votos de cada um dos eleitores. Em seu voto, o eleitor i indicará o número L_i de candidatos em quem votará, e os índices destes candidatos na sua ordem de preferência. Índices de candidatos fora do intervalo $[1, K]$ significam votos em branco apenas para a opção de preferência correspondente. Se indicar mais que V votos, os últimos serão desconsiderados. Um eleitor nunca indica o mesmo candidato mais de uma vez.

A entrada deve ser lida da entrada padrão.

Saída

Para cada instância da entrada seu programa deverá imprimir, em uma única linha, a lista de candidatos eleitos ordenada pela classificação dos candidatos na eleição. No caso de dois candidatos possuirem a mesma classificação, o de menor índice vem antes.

A saída deve ser escrita na saída padrão.

Restrições

- $1 \leq N \leq 10^5$
- $1 \leq V \leq K \leq 100$
- $1 \leq L_i \leq 100$

Exemplos

Exemplo de entrada	**Saída para o exemplo de entrada**
5 3 2 2 1 3 2 2 3 2 1 3 2 2 3 1 1 3 6 3 3 1 5 3 3 1 0 3 3 1 4 5	3 1 1 5 3

Problema G: Banho de sol no jardim

Arquivo: jardim.[c|cpp|java]

Uma empresa quer demolir um prédio para construir um jardim em uma das principais avenidas de Ecaterimburgo, mas antes quer fazer um estudo: descobrir o tempo de luz de sol que esse jardim receberia por dia, considerando que ela irá demolir o prédio escolhido e construir o jardim no lugar. Isso é importante, pois banho de sol é uma atividade muito popular entre os habitantes da cidade durante o verão. O jardim recebe luz do sol se pelo menos um pedaço dele estiver recebendo raios solares.

Para facilitar as coisas, a construtora escolheu um dia do ano para fazer as medições. Nesse dia o sol nasce às 5:30 e se põe às 21:30 (no verão russo os dias são longos). Considere que o sol tem velocidade angular constante no céu e que ele nasce no leste e se põe no oeste.

Todos os prédios são alinhados na direção leste-oeste, têm a mesma largura e o espaçamento entre eles é constante, o jardim terá largura igual à dos prédios.

Entrada

A entrada é composta por diversas instâncias e termina com final de arquivo (`EOF`).

Cada instância consiste de 2 linhas. A primeira linha contém 3 inteiros N, L, D, o número de prédios, a largura de cada um deles, e o espaçamentos entre eles, respectivamente. A segunda linha contém N inteiros h_i, as alturas dos prédios. Uma altura de 0 indica que o jardim será construído nessa posição e aparecerá exatamente uma vez. Todas as medidas são em metros.

A entrada deve ser lida da entrada padrão.

Saída

Para cada caso de teste imprima uma linha contendo o tempo, em minutos, de luz solar que o jardim receberá no dia escolhido, arredondado para 2 casas decimais.

A saída deve ser escrita na saída padrão.

Restrições

- $2 \leq N \leq 100$
- $1 \leq L, D \leq 100$
- $0 \leq h_i \leq 50$

Exemplos

Exemplo de entrada	**Saída para o exemplo de entrada**

`3 2 1`
`10 0 5`
`4 5 3`
`0 2 4 20`
	`254.20`
	`747.70`

Problema H: Escalonamento de salas de aula

Arquivo: `salas.[c|cpp|java]`

Os professores da Universidade de Ecaterimburgo não gostam de deslocar-se por longas distâncias. Cada docente deseja que as salas em que ele vai dar aula estejam em posições adjacentes. No início de cada semestre cabe ao responsável pela Comissão de Graduação determinar as salas de aula em que os docentes deverão dar aula. Cada docente sabe que turma de alunos deverá assistir às suas aulas como, por exemplo, alunos do terceiro período de Engenharia Mecânica, ou alunos do primeiro período de Computação, etc. Os alunos de cada turma permanecem na mesma sala em todas as aulas. O importante é que todas as salas em que um docente dá aulas fiquem em posições adjacentes. Nem sempre é possível satisfazer os requisitos dos docentes. Se, por exemplo, um docente dá aulas para o terceiro semestre de Matemática e primeiro semestre de Computação, um segundo dá aulas para o primeiro semestre de Computação e segundo período de Engenharia Elétrica e um terceiro professor dá aulas para os alunos do segundo período de Engenharia Elétrica e terceiro semestre de Matemática, claramente não é possível satisfazer os três professores. Sua tarefa é ajudar o responsável pela alocação das salas, e determinar se é possível satisfazer todos os requisitos dos docentes.

Entrada

A entrada é composta por diversas instâncias e termina com final de arquivo (`EOF`).

Na primeira linha de cada instância é dado o número de turmas T, numeradas de 1 a T, e o número de docentes D. Nas D linhas seguintes são dados o número K de turmas em que o docente correspondente dá aulas seguido pelas identificações destas turmas em ordem crescente.

A entrada deve ser lida da entrada padrão.

Saída

Para cada instância o seu programa deverá imprimir uma permutação das turmas que atenda os requisitos de todos os docentes, ou seja, todas as turmas em que um docente dá aula estejam adjacentes. Caso não exista uma tal permutação seu programa deverá imprimir `impossivel`. Se existir mais de uma permutação possível, qualquer uma será aceita.

A saída deve ser escrita na saída padrão.

Restrições

- $1 \leq T \leq 10^3$
- $1 \leq D \leq 10^3$
- $0 \leq K \leq T$

Exemplos

Exemplo de entrada	Saída para o exemplo de entrada
5 4 3 2 4 5 2 2 5 2 1 5 2 1 3 3 3 2 1 2 2 2 3 2 1 3	4 2 5 1 3 impossivel

Problema I: Entendendo o sorobov

Arquivo: `sorobov.[c|cpp|java]`

Instrumentos de ajuda a calcular existem há séculos. Muito antes do surgimento das máquinas de calcular no século XVII, chineses e japoneses faziam uso de ábacos com os quais podem fazer operações matemáticas sofisticadas em velocidade estonteante. Um instrumento semelhante foi recentemente descoberto em escavações nas imediações da cidade de Ecaterimburgo. Acredita-se ser um ábaco similar ao japonês, chamado, em russo, sorobov (соробов).

O sorobov tem nove colunas, onde cada coluna corresponde a um dígito. A coluna mais à direita representa a unidade, a segunda mais à direita representa as dezenas e assim por diante. Existem 7 linhas, sendo as duas primeiras separadas, por uma barra, das 5 últimas. Na parte de cima (duas primeiras linhas) cada coluna possui uma única pedra, que se encostada na barra separadora soma 5 ao valor do dígito correspondente. Na parte de baixo cada coluna possui 4 pedras e um espaço vazio e a quantidade de pedras entre a barra separadora e o espaço vazio é somado ao valor do dígito correspondente. Dessa forma, dizemos que as pedras de cima valem 5 e as de baixo valem 1.

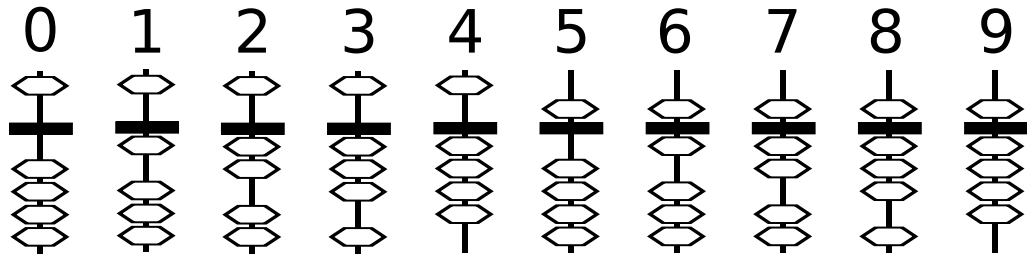

Figura 1: Ilustração de como representar os números de 0 a 9.

Sua tarefa neste problema será, dado um número N imprimir uma representação da configuração do sorobov correspondente ao número.

Entrada

A entrada é composta por diversas instâncias e termina com final de arquivo (`EOF`).

Cada instância corresponde a uma única linha contendo o número N a ser inserido no sorobov.

A entrada deve ser lida da entrada padrão.

Saída

Para cada instância imprima a representação do número N no sorobov com o seguinte formato. As duas primeiras linhas correspondem às pedras que valem 5, na linha seguinte imprima `---------` (nove hífens) e as próximas cinco linhas correpondem às pedras que valem 1. Cada linha de pedras deve conter nove caracteres, onde '0' corresponde a um espaço vazio e '1' a uma pedra.

Imprima uma linha em branco ao final de cada instância **(inclusive a última)**.

A saída deve ser escrita na saída padrão.

Restrições

- $0 \leq N < 10^9$

Exemplos

Exemplo de entrada	Saída para o exemplo de entrada
2 16	111111111 000000000 --------- 000000001 111111111 111111110 111111111 111111111 111111110 000000001 --------- 000000011 111111100 111111111 111111111 111111111

Problema J: Turismo em Ecaterimburgo

Arquivo: `turismo.[c|cpp|java]`

Muitos podem pensar: "O que vou fazer em Ecaterimburgo? Essa cidade é no fim do mundo!!!". Entretanto, muitas coisas interessantes ocorreram na cidade, possuindo vários monumentos e locais históricos. Para citar alguns, Ecaterimburgo tem um monumento que é um grande teclado de computador localizado na beira do rio Izet; um monumento a Michael Jackson (!!); na mansão Ipatiev foram assassinados os Romanovs (o czar Nicolau, sua esposa, quatro filhas e filho); lá houve um vazamento de antraz em 1979; um piloto de U2 americano foi capturado e condenado por espionagem; entre outros. Ou seja, existe muito a fazer nos dias em que a cidade for visitada.

A central de turismo de Ecaterimburgo, construiu um mapa com as principais atrações turísticas da cidade, assim como os belos passeios ligando esses caminhos. Esse mapa também mostra um nível de dificuldade de cada passeio (relacionado a duração, pavimentação da via, relevo etc.) e o sentido no qual ele deve ser feito. Eles desejavam construir uma rota que passasse por todas as atrações turísticas e os passeios. Foi idealizado, então, um concurso que visava fazer esta rota e, ao mesmo tempo, homenageava uma das cidades irmãs de Ecaterimburgo: Caliningrado, cujo nome até o final da segunda guerra mundial era Königsberg. A ideia era construir uma rota em que se partisse de uma das atrações, e passando por todos os passeios se retornasse ao ponto de partida. Sabemos que, como no caso das pontes de Königsberg, nem sempre é possível construir uma rota assim. Por isso a central permitiu que, se necessário, os passeios poderiam ser feitos mais de uma vez. No entanto, ela exigiu que a dificuldade total da rota (soma das dificuldades de cada passeio multiplicado pelo número de vezes que ele é feito) fosse mínima. Dessa forma, o concurso consistia de propor, a partir de uma rota inicial, quais passeios deveriam ser percorridos mais de uma vez e quandas vezes, para se obter uma rota como a desejada pela central.

Entrada

A entrada é composta por diversas instâncias e termina com final de arquivo (`EOF`).

A primeira linha de cada instância contém dois inteiros N e M, o número de atrações da cidade e o número de passeios respectivamente. As próximas M linhas contém três inteiros, a_i, b_i, d_i indicando que o passeio i começa em a_i, termina em b_i e tem dificuldade d_i.

A entrada deve ser lida da entrada padrão.

Saída

Para cada instância imprima a dificuldade total mínima da rota desejada. Se for impossível obter uma rota da forma desejada, imprima `impossivel`.

A saída deve ser escrita na saída padrão.

Restrições

- $2 \leq N \leq 50$
- $0 \leq M \leq N^2 + 10^3$
- $1 \leq a_i, b_i \leq N$
- $1 \leq d_i \leq 3 \times 10^4$

Exemplos

Exemplo de entrada	Saída para o exemplo de entrada
2 2 1 2 10000 2 1 30000 4 7 1 2 1 2 1 2 2 3 4 2 3 4 3 2 3 3 4 10 4 3 100 3 2 1 2 1000 2 3 1000	40000 127 impossivel

Problema K: O incidente de Sverdlovsk

Arquivo: `antraz.[c|cpp|java]`

Durante os anos da União Soviética o nome da cidade de Ecaterimburgo era Sverdlovsk, em homenagem ao bolchevique Iakov Sverdlov, filho de um artesão judeu que era excelente orador e foi um dos principais protagonistas ao lado de Lenin na revolução de outubro de 1905. Era considerado honesto, enérgico e trabalhador e respeitado por todos os setores do partido. Faleceu aos 34 anos. A cidade retomou o nome original em 1991 por iniciativa de Boris Yeltsin primeiro presidente da Rússia, nascido na cidade.

Em 2 de abril de 1979, quando a cidade ainda se chamava Sverdlovsk houve um vazamento de antraz de uma fábrica militar na cidade. Este incidente é muitas vezes chamado de "Chernobyl biológico", e causou aproximadamente 100 mortes, apesar de que o número exato de vítimas e contaminados seja desconhecido. A União Soviética negou por anos as reais causas do acidente e todos os registros das vítimas desapareceram, pois poderiam revelar sérias violações da Convenção de Armas Biológicas.

As autoridades soviéticas tiveram de recorrer a procedimentos altamente sofisticados de descontaminação, especialmente das áreas rurais. Cada área retangular de dimensões N por M metros era dividida em $N \times M$ setores quadrados de um metro quadrado. Estes setores eram identificados pelas coordenadas de seus centros, numeradas de oeste para leste e de sul para norte a partir de $(1, 1)$.

Cada setor seria considerado descontaminado se ele for coberto por pelo menos K agentes de saúde. Cada agente era capaz de cobrir uma área circular. O raio dessa área variava de acordo com os equipamentos usados e com a experiência do agente de saúde. Sua tarefa é determinar quantos desses setores são considerados descontaminados, isso é, cobertos por pelo menos K agentes. Consideramos que um setor é coberto se seu centro está numa área coberta por um agente de saúde.

Entrada

A entrada é composta por diversas instâncias e termina com final de arquivo (`EOF`).

A primeira linha de cada instância contém dois inteiros, N e M, indicando a dimensão da área retangular falada no enunciado. A segunda linha contém o número de agentes, C, e o número K. As C linhas seguintes têm a descrição dos agentes X_c , Y_c e R_c, onde (X_c, Y_c) é o centro da área circular de raio R_c que o agente cobre.

A entrada deve ser lida da entrada padrão.

Saída

Para cada instância imprima o número de setores que são cobertos por pelo menos K agentes.

A saída deve ser escrita na saída padrão.

Restrições

- $1 \leq N \leq 10^3$
- $1 \leq M \leq 10^5$
- $1 \leq K \leq C \leq 10^3$
- $1 \leq X_c \leq N$
- $1 \leq Y_c \leq M$
- $0 \leq R_c \leq 10^8$

Exemplos

Exemplo de entrada	Saída para o exemplo de entrada
10 10 2 1 3 3 2 8 8 2 15 15 6 2 4 4 2 5 5 1 6 6 3 7 7 2 10 10 0 11 10 1	26 20

Observação

Mais precisamente, um setor com centro (x_s, y_s) é coberto por um agente posicionado em (x_a, y_a) capaz de descontaminar uma área com raio r se

$$(x_s - x_a)^2 + (y_s - y_a)^2 \leq r^2.$$

Problema L: Produção em Ecaterimburgo

Arquivo: `producao.[c|cpp|java]`

Ecaterimburgo é uma cidade russa localizada na fronteira entre a Europa e a Ásia, nos montes Urais. É a quarta maior cidade da Rússia com mais de 1,4 milhões de habitantes. A principal atividade econômica da cidade está relacionada com a produção de máquinas industriais. As fábricas da cidade produzem boa parte de todas as máquinas usadas na Rússia e exportada para diversos países de todo o mundo. Em especial a produção de ferramentas industriais é famosa no país. As ferramentas são produzidas por máquinas altamente especializadas, e, para cada ferramenta a ser produzida as máquinas gastam um tempo pré-estabelecido para sua produção.

Uma das fábrica possui apenas uma dessas máquinas e seu gerente precisa da sua ajuda para melhorar sua produtividade. Os pedidos de ferramentas chegam na fábrica de forma contínua, isto é, no início do dia nem todos os pedidos podem ser processados, pois estes estarão disponíveis ao longo do dia. O gerente acha que os funcionários não estão escolhendo bem a ordem na qual os pedidos são atendidos e quer analisar as sequências de pedidos de dias anteriores. Dessa forma, ele pede que você determine, para um dado dia, o menor instante possível em que todos os pedidos estariam finalizados.

Entrada

A entrada é composta por diversas instâncias e termina com final de arquivo (`EOF`).

Cada instância começa com o número N de tarefas que serão processadas no dia. As N linhas seguintes têm o tempo d_i em que a tarefa estará disponível e o tempo p_i de processamento da tarefa na máquina. O início do processamento se dá no instante 1.

A entrada deve ser lida da entrada padrão.

Saída

Para cada instância seu programa deverá imprimir o menor instante em que a tarefa que for processada por último terminará seu processamento.

A saída deve ser escrita na saída padrão.

Restrições

- $1 \leq N \leq 10^5$
- $1 \leq d_i, p_i \leq 10^4$

Exemplos

Exemplo de entrada	**Saída para o exemplo de entrada**
5 10 2 6 1 4 3 1 2 1 4 4 1 1 1 3 5 2 5 3	13 10