Maratona de Programação de 2011

Universidade de São Paulo

Caderno de Problemas

Departamento de Ciência da Computação IME-USP

Domingo, 21 de agosto de 2011.

Problema A: Cartas

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

Jorge é um aluno que tem uma memória assombrosa. Recentemente ele inventou uma demonstração de suas habilidades, utilizando um conjunto de N baralhos completos, ou seja, cada baralho é composto por 52 cartas de quatro naipes (Espadas, Copas, Paus e Ouros) e em cada naipe há treze cartas (com figuras de Ás a Rei).

No início da demonstração, os N baralhos são colocados sobre a mesa formando uma fila de N baralhos. Os baralhos são identificados por seus índices na fila de baralhos (de 0 a $N-1$). No início da demonstração cada baralho é ordenado da forma tradicional, primeiro por naipe (na ordem Espadas, Copas, Paus e Ouros do fundo para o topo) e então pela figura (figuras maiores mais perto do topo). Ou seja, em cada baralho os naipes do fundo para o topo são Espadas, Copas, Paus e Ouros, e as figuras do fundo para cima são Ás, Dois, Três, ..., Nove, Dez, Valete, Rainha e Rei, de forma que inicialmente em cada baralho a carta de baixo é o Ás de Espadas e a carta no topo é o Rei de Ouros. Na descrição abaixo, vamos considerar que a carta de baixo de um baralho tem índice 0 e a carta no topo do baralho tem índice 51, e que os valores das cartas são 1 para o Ás, x para o número x ($2 \leq x \leq 10$), 11 para o Valete, 12 para a Rainha e 13 para o Rei.

Para a demonstração, Jorge, de olhos vendados, planeja pedir a um voluntário da platéia que realize uma sequência das seguintes operações nos baralhos:

- *Embaralha*: o voluntário deve escolher quatro inteiros A, B, I e J e anunciá-los para a platéia e para Jorge. O voluntário deve então trocar a carta de índice I com a carta de índice J em cada baralho cujos índices estão entre A (inclusive) e B (exclusive).

- *Soma*: o voluntário deve escolher dois inteiros A e B, calcular a soma dos valores das figuras no topo dos baralhos cujos índices estão entre A (inclusive) e B (exclusive), escrever o resultado em uma placa e mostrá-lo para a platéia.

O voluntário irá realizar uma sequência qualquer das operações definidas e, para mostrar a sua habilidade de raciocínio e memorização, a cada operação de Soma, Jorge vai anunciar o resultado para que a platéia possa conferir sua resposta com o resultado calculado e mostrado pelo voluntário.

Como Jorge quer praticar a demonstração antes de apresentá-la a uma platéia real, ele quer que você escreva um programa que, dada uma sequência qualquer de operações Embaralha e Soma, imprime o resultado de cada uma das operações de Soma.

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 respectivamente o número de baralhos e o número de operações da sequência. Cada uma das M linhas seguintes contém a descrição de uma operação. Uma linha que descreve uma operação *Embaralha* contém a letra '`E`' e quatro inteiros A, B, I, e J, todos separados por espaços em branco. Uma linha que descreve a operação *Soma* contém a letra '`S`' e dois inteiros A e B, todos separados por espaços em branco.

A entrada deve ser lida da entrada padrão.

Saída

Para cada operação *Soma*, de cada instância, seu programa deve imprimir uma linha contendo o inteiro que Jorge deve anunciar para platéia.

A saída deve ser escrita na saída padrão.

Restrições

- $0 \leq N \leq 5 \times 10^4$.
- $0 \leq M \leq 10^6$.
- $0 \leq A \leq B \leq N$.
- $0 \leq I, J < 52$ e $I \neq J$.

Exemplos

Exemplo de entrada	Saída para o exemplo de entrada
10 6 S 0 10 S 5 5 E 0 5 51 0 S 0 10 E 5 10 51 1 S 0 10 8 3 E 2 4 51 48 E 0 6 51 49 S 1 3	130 0 70 15 22

Problema B: Corrida Aleatória

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

A cada dois anos realiza-se a conferência "Random Structures and Algorithms", que reúne os principais pesquisadores do mundo na área. Já é uma tradição a realização da "Random Run" (veja http://rsa2011.amu.edu.pl/run.php para a 15a. edição do evento, realizado em Atlanta em 2011) uma corrida que é um evento aleatório realizado da seguinte forma: inicialmente é jogado um dado, que determina o número de voltas que devem ser corridas em uma pista de atletismo. Os corredores saem em disparada e, assim que cada um dos competidores completa o número de voltas definido no primeiro sorteio o dado é jogado novamente, definindo o número de voltas que cada corredor deve completar para finalizar a prova.

A 16a. conferência está prevista para realizar-se na Polônia em 2013, próximo da data da final do ICPC. Neste ano os organizadores pretendem fazer um evento mais, digamos, cultural, mantendo, sempre, o aspecto aleatório e divertido da competição. Serão escolhidos N bares da cidade de Varsóvia (internacionalmente conhecida por seus bares animados e cerveja de primeira). Inicialmente a organização monta uma rede entre estes bares, definindo que dois bares são vizinhos se eles compartilham algum tipo de característica (ter as melhores cervejas, o ambiente mais agradável, as garçonetes mais simpáticas, etc). Os participantes iniciam, então a corrida no bar B e podem continuar no mesmo bar ou ir para um dos seus vizinhos. Se o bar tem D vizinhos, a probabilidade do corredor ficar em B é $\frac{1}{D+1}$ e a probabilidade de seguir para cada um dos vizinhos é também igual a $\frac{1}{D+1}$. Ganha a corrida o participante que retornar primeiro ao bar de saída. Note, inclusive que um corredor pode ganhar a corrida sem dar um só passo! Os organizadores imaginam que isso será um evento de grande interesse, e poderá gerar muita diversão entre os participantes, ainda que a chance de muitos não completarem a prova pode crescer muito.

Como todos são matemáticos, os organizadores estão preocupados com a possibilidade de algum corredor demorar muito, ou mesmo nunca terminar a corrida... Dessa forma, eles pediram a vocês que façam um programa que calcule o número esperado de visitas a bares que um corredor fará até terminar a corrida.

Deve-se contar repetições e não contar o bar de partida. Por exemplo, suponha que um competidor iniciou a corrida no bar B, depois foi para o bar A, permaneceu no bar A e em seguida voltou para o bar B. Então o número de visitas foi 3, pois contamos as duas visitas ao bar A e a visita ao bar B que encerrou a corrida.

Entrada

A entrada é composta por diversas instâncias, que devem ser lidas até o fim do arquivo (EOF).

A primeira linha de cada instância contém dois inteiros, N e M. Que representam respectivamente a quantidade de bares e a quantidade de relações de vizinhaça entre bares. Seguem-se M linhas, cada uma com dois inteiros, A e B, representando que os bares A e B, $A \neq B$ são vizinhos. Os bares são numerados de 1 até N. Não haverá repetições da mesma ligação no arquivo de entrada.

A entrada deve ser lida da entrada padrão.

Saída

Para cada instância, imprima N linhas, a i-ésima linha deve conter dois inteiros positivos A_i e B_i separados por espaço. Ela indica que o valor esperado pedido para um competidor que inicie a corrida no bar i é dado por $\frac{A_i}{B_i}$. A_i e B_i não podem ter um divisor comum maior que 1.

A saída deve ser escrita na saída padrão.

Restrições

- $1 \leq N \leq 1000000$
- $1 \leq M \leq 10000000$

Exemplos

Exemplo de entrada	Saída para o exemplo de entrada
2 1 1 2 5 3 1 2 3 4 4 5 3 3 1 2 2 3 3 1	2 1 2 1 2 1 2 1 7 2 7 3 7 2 3 1 3 1 3 1

Problema C: Mina de sal

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

A cidade de Wieliczka na Polônia é conhecida por sua imponente mina de sal. Reza a lenda que Santa Conegunda, filha de um rei húngaro, foi oferecida em casamento ao rei da Polônia, e seu pai quis oferecer-lhe vários tesouros como dote. A santa recusou, dizendo que preferia receber o seu dote em sal, um bem primordial para a existência humana. O pai lhe deu, então, uma mina de sal na Romênia, onde a santa jogou seu anel de ouro em agradecimento. Anos mais tarde, ao passar pela cidade de Wieliczka, a santa apontou um lugar no solo e pediu que se cavasse ali. Para espanto de todos descobriu-se no local uma mina de sal, e, mais surpreendente, o anel da santa estava ali.

A extração de sal mineral deve respeitar restrições importantes, uma vez que o material deve ser retirado de forma que as paredes e teto da mina permaneçam estáveis e não caiam sobre as cabeças dos mineiros. Estudos de engenheiros de minas da Universidade da Cracóvia mostram que pode-se modelar a produção de sal da mina através de um polinômio $A(x)$ (onde x é o volume total de material retirado da mina). Sabe-se que depois de um período de trabalho, em que foi retirado um volume x_0 de material da mina a produção passa a ser dada por $q(x)$ e existe um desperdício r onde $A(x) = q(x)(x - x_0) + r$.

Sua tarefa neste problema é dados $A(x)$ e um inteiro x_0, determinar o desperdício r e o novo polinômio relacionado à capacidade de produção da mina $q(x)$.

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 x_0 que representam o grau do polinômio e o valor retirado da mina de sal, respectivamente. A linha seguinte contém $n+1$ inteiros que indicam os coeficientes do polinômio $A(x)$. Isto é, o i-ésimo inteiro da linha seguinte é o coeficiente de x^{i-1} de $A(x)$.

A entrada deve ser lida da entrada padrão.

Saída

Para cada instância seu programa deve imprimir duas linhas. A primeira linha deve ser composta de um inteiro indicando o desperdício r. A segunda linha deve conter n inteiros, onde o i-ésimo inteiro é o coeficiente de x^{i-1} de $q(x)$.

A saída deve ser escrita na saída padrão.

Restrições

- $1 \leq n \leq 15$.
- $1 \leq x_0 \leq 50$.
- O coeficiente de cada termo do polinômio $A(x)$ é um inteiro entre 0 e 100.

Exemplos

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

`2 3`
`3 2 4`
`3 7`
`1 10 5 7`
	`45`
	`14 4`
	`2717`
	`388 54 7`

Problema D: Flores de fogo

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

Nos dias atuais uma flor de fogo não é algo considerado estranho para muitos jovens. Isso porque um famoso jogo de videogame popularizou esse tipo de flor. Nesse jogo o protagonista ganhava superpoderes ao tocar em uma flor de fogo, passando a atirar pequenas bolas de fogo para derrotar seus inimigos.

No entanto, já se falava sobre flores de fogo há muito tempo atrás. Na mitologia polonesa, flores de fogo são flores místicas de grande poder guardadas por espíritos malignos. Ela possuía esse nome porque brilhava tanto que era impossível olhá-la diretamente. Quem possuísse uma flor dessas ganharia a habilidade de ler a mente de outras pessoas, encontrar tesouros escondidos e repelir todos os males.

Para obter uma flor de fogo, a pessoa deveria procurá-la em uma floresta antes da meia-noite na véspera do *Noc Kupały*. Exatamente à meia-noite ela floresceria. Para colhê-la seria preciso desenhar um círculo em volta dela. Parece uma tarefa fácil, no entanto, os espíritos malignos que guardam a flor tentariam de tudo para distrair qualquer um tentando colher a flor. Se a pessoa falhasse ao tentar desenhar um círculo em volta da flor, teria sua vida sacrificada.

Dados dois círculo, um desenhado por um ambicioso caçador de flores de fogo e outro representando a área da flor, sua tarefa é determinar se o caçador morre ou fica rico com sua conquista.

Entrada

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

Cada instância consiste em uma linha com seis inteiros, R_1, X_1, Y_1, R_2, X_2, Y_2. O círculo desenhado pelo caçador possui raio R_1 e centro $(X_1,\ Y_1)$. O círculo representando a área da flor possui raio R_2 e centro $(X_2,\ Y_2)$.

A entrada deve ser lida da entrada padrão.

Saída

Para cada instância imprima uma única linha contendo `MORTO`, se o caçador morre, ou `RICO` se o caçador consegue colher a flor.

A saída deve ser escrita na saída padrão.

Restrições

- $1 \leq R_1,\ R_2 \leq 1000$
- $|X_1|,\ |Y_1|,\ |X_2|,\ |Y_2| \leq 1000$

Exemplos

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

`6 -8 2 3 0 0`
`7 3 4 2 4 5`
`3 0 0 4 0 0`
`5 4 7 1 8 7`
	`MORTO`
	`RICO`
	`MORTO`
	`RICO`

Problema E: Fortaleza polonesa

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

O castelo de Malbork é o maior castelo do mundo construído com tijolos. Foi edificado em 1274 em homenagem à Virgem Maria, e fica à beira do rio Nogat, na Polônia. É patrimônio mundial da UNESCO desde 1997. Durante a segunda guerra mundial o castelo foi totalmente destruído. Nesta época descobriu-se que o castelo na verdade foi construído em diversas fases, e o conjunto foi cercado por uma fortaleza e um muro apenas alguns séculos após o início de sua construção. Naquela época a construção de muros de proteção era uma tarefa muito elaborada. Táticas de guerra apontavam para a conveniência de construção de nichos nos muros, que são reentrâncias nos mesmos. Nesses nichos um soldado poderia ficar em relativa segurança, uma vez que podia enxergar a área externa delimitada pelos segmentos do muro conectados ao nicho sem precisar girar a cabeça (considerava-se que os soldados tinham campo de visão de 180 graus). Observa-se na fortaleza do castelo de Malbork a existência de vários destes nichos em que eram colocados os soldados que faziam a proteção do castelo. Quanto mais destes nichos um muro tinha, mais segura era considerada a fortaleza.

Sua tarefa neste problema é dados N pontos, com os cantos de uma fortaleza, determinar quantos deles formam nichos como descrito acima, em que soldados podem ser colocados para a proteção do castelo.

Entrada

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

Cada instância começa com um inteiro N indicando o número de cantos da fortaleza. Cada uma das N linhas seguintes contém um par de inteiros X e Y, representando a coordenada de um canto da fortaleza. Os cantos são dados em sequência, podendo ser tanto no sentido horário quanto anti-horário. Não existem três cantos consecutivos que sejam colineares e o muro da fortaleza não possui cruzamentos.

A entrada deve ser lida da entrada padrão.

Saída

Para cada instância imprima uma única linha contendo o número de nichos da fortaleza.

A saída deve ser escrita na saída padrão.

Restrições

- $3 \leq N \leq 100000$
- $|X|, |Y| \leq 18000$

Exemplo

Exemplo de entrada	Saída para o exemplo de entrada
13 0 0 700 100 400 500 1300 600 700 400 1400 200 2200 1200 1500 2000 1500 1700 400 1700 400 2200 0 1900 10 1000 3 -32 45 2 19 -100 -3	5 0

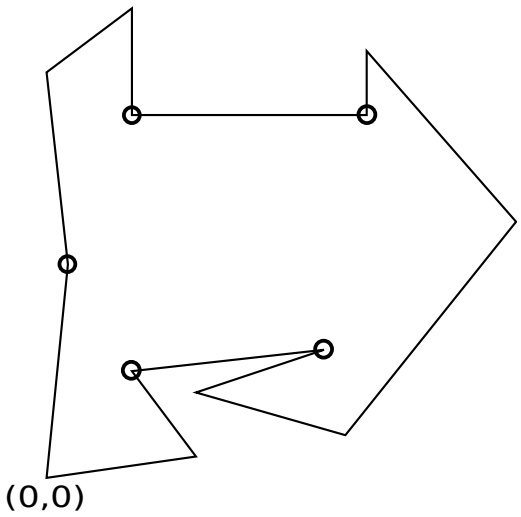
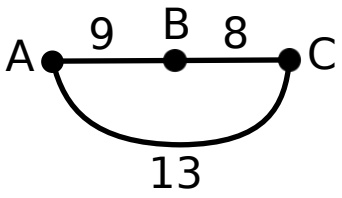


Figura 1: Ilustração do primeiro exemplo.

Problema F: Pregão de serviços

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

Está cada vez mais comum o uso de estratégias baseadas nas redes sociais para fazer compras, contratar serviços, combinar encontros, etc. Tais estratégias muitas vezes envolvem conhecimentos matemáticos interessantes, como, por exemplo, de teoria dos jogos. Uma nova febre está chegando, que é o uso das redes para a realizações de pregões virtuais de contratação de serviços. As diversas prestadoras de serviço têm acordos entre si que permitem que o serviço a ser contratado custe um determinado valor. Os preços que uma mesma empresa cobra quando em colaboração com cada um de seus parceiros pode ser muito diferente, uma vez que diversos fatores como localização, parcerias anteriores, tradição e outros influem na conveniência de uma determinada parceria. Assim, quando alguém deseja contratar um determinado serviço pode verificar quanto é o melhor preço existente na rede social que interliga duas empresas em que ele confia. Note que o melhor preço pode ser fornecido por um par de empresas diferente deste par inicial dado, desde que seja possível a partir das empresas iniciais construir uma rede social de parcerias que atinja as duas empresas que serão contratadas. No exemplo acima, apesar da parceria direta entre A e C custar 13, o melhor preço na



rede para contratá-las é 9, através da parceria indireta que usa B. Note que o preço de uma parceria indireta é o preço da parceria mais cara da corrente de parcerias usadas.

Sua tarefa é fazer um programa de apoio a esses pregões virtuais. A pessoa interessada em contratar um determinado serviço escolhe duas empresas de sua confiança, e contratará o serviço pelo melhor preço solicitado por uma parceria de empresas que esteja na rede social destas duas empresas inicialmente escolhidas.

Entrada

A entrada é composta por diversas instâncias. A primeira linha da entrada contém um inteiro T indicando o número de instâncias.

A primeira linha de cada instância contém inteiros N e M indicando, respectivamente, o número de empresas e de parcerias. Em seguida existem M linhas com três inteiros A, B e C, representando a parceria entre A e B com custo C. As empresas são numeradas de 0 a $N-1$. A próxima linha contém um inteiro Q. As próximas Q linhas contém o par de empresas cujo preço da parceria deve ser consultado.

A entrada deve ser lida da entrada padrão.

Saída

Para cada instância imprima uma linha com o melhor preço de parceria para cada consulta.

A saída deve ser escrita na saída padrão.

Restrições

- $1 \leq N \leq 500$
- $1 \leq M \leq \frac{N*(N-1)}{2}$
- $1 \leq C \leq 1000000$
- $1 \leq Q \leq 10000$

Exemplos

Exemplo de entrada	Saída para o exemplo de entrada
2 4 4 0 1 6 0 2 1 1 3 2 2 3 2 2 0 1 0 3 4 4 0 1 1 1 2 7 1 3 3 2 3 4 4 0 2 0 3 2 1 3 1	2 2 4 3 4 3

Problema G: 7168 – SMOK

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

O dragão da colina de Wawel habitava uma caverna nesta colina localizada na Cracóvia, Polônia. Narrativas da existência deste dragão remontam ao século XII. Em 1970 foi construída uma escultura em bronze no local onde teria morado o dragão que solta fogo pela boca de tempos em tempos, ou quando alguém manda um SMS com a palavra "SMOK"(dragão em polonês) para o número 7168.

O dragão foi morto através de um método inventado por Skuba, um aprendiz de sapateiro na época do Rei Krak o fundador, segundo a lenda, na cidade de Cracóvia. O Rei Krak prometia àquele que matasse o dragão a mão de sua jovem filha, a princesa. Vários dos mais bravos cavaleiros tentaram sem sucesso matar a besta, e vários acabaram morrendo tentando. Em busca de um bravo guerreiro que resolvesse o problema o Rei Krak enviou mensageiros a várias cidades polonesas. O rei Krak era muito religioso, e acreditava no misticismo dos números. Ele queria que seus mensageiros espalhassem a notícia da procura de um matador de dragões por todos os vilarejos, mas os mensageiros deveriam passar por exatamente K estradas entre a cidade de origem e a final. Ele não impedia, entretanto, que o mensageiro passasse por uma mesma cidade ou estrada várias vezes.

Depois de anos de tentativas infrutíferas o próprio Rei Krak decidiu que mataria o dragão, mas foi interrompido por Skuba que lhe disse que teria inventado uma forma de matar a besta. Skuba pega um carneiro bem gordo que mata e recheia com alcatrão e enxofre. Este carneiro é deixado durante a noite (quando o dragão está dormindo) na entrada da caverna. Pela manhã a cidade é acordada por uma violenta explosão: o dragão engole o carneiro, mas tem uma sede terrível. De tanto tomar água no rio da cidade sua barriga explode e pedaços do dragão cobrem toda a cidade. Como prometido a mão da princesa é oferecida a Skuba, casam-se e vivem felizes para sempre.

Sua tarefa neste problema é dada uma lista de cidades de origem e final e um inteiro K, determinar quantos mensageiros deveriam ser enviados de cada cidade de origem para a cidade final correspondente de forma que todos os passeios que passam por K estradas sejam percorridos.

Entrada

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

A primeira linha de cada instância contém os inteiros N e K representado, respectivamente, o número de cidades no reino e estradas que devem ser percorridas pelos mensageiros. As cidades são numeradas de 1 a N. Cada uma das N linhas seguintes descreve o mapa do reino. A i-ésima linha, $2 \leq i \leq N + 1$, corresponde a cidade i e contém um inteiro indicando o número de cidades vizinhas seguido da lista dessas cidades. A $(N + 2)$-ésima linha da instância contém um inteiro P. As próximas P linhas contém o par de cidade de origem e cidade de final, que são sempre distintas.

A entrada deve ser lida da entrada padrão.

Saída

Para cada instância imprima P linhas com os números de mensageiros a serem enviados da cidade de origem para a final, considerando a mesma ordem apresentada na entrada. Caso não seja possível chegar a cidade final passando por exatamente K estradas, imprima -1. Imprima uma linha em branco após cada instância.

Como o número de mensageiros pode ser muito grande, imprima-os módulo 30203.

A saída deve ser escrita na saída padrão.

Restrições

- $2 \leq N \leq 100$
- $1 \leq K \leq 1000$
- $1 \leq P \leq N * (N - 1)$

Exemplos

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

Problema H: Vestibular de Varsóvia

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

A Universidade de Varsóvia é uma das instituições de ensino mais tradicionais da Europa, com quase 200 anos de idade. Lá estudaram personalidades ilustres como Chopin (músico), Menachen Begin e Itzhak Shamir (primeiros-ministros de Israel) e Leonid Hurwicz (matemático ganhador do Nobel de Economia de 2007). A universidade se orgulha por manter suas tradições, e, desde sua fundação, o ingresso permanece funcionando da mesma forma. A universidade mantém M cursos em todas as áreas do conhecimento, cada um dos quais oferece um certo número de vagas, e recebe N candidatos que devem escolher quantos cursos são do seu interesse, e deve estabelecer uma ordem de preferência, começando no curso que mais deseja cursar até o último que lhe interessa. Os cursos que não são listados por um candidato não são considerados por ele, mesmo que haja sobra de vagas. Então, os candidatos são submetidos a uma série de provas comuns a todos os cursos oferecidos, e cada candidato recebe uma pontuação. Esta pontuação será usada na escolha dos cursos de cada candidato. Os candidatos são considerados em ordem decrescente de sua pontuação e serão alocado no primeiro curso de sua preferência que tenha vaga . Os empates são resolvidos de acordo com a preferência. Se um candidato empatou com outro, e ambos querem um mesmo curso, mas o primeiro o quer em sua terceira opção enquanto o segundo na décima, leva a vaga o primeiro candidato. Caso o empate permaneça, a vaga é dada para quem se inscreveu primeiro.

Sua tarefa neste problema é escrever um sistema de computação para fazer a seleção dos candidatos à Universidade de Varsóvia.

Entrada

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

A primeira linha de cada instância possui dois inteiros N e M, indicando o número de candidatos e o número de cursos, respectivamente. Os cursos são identificados pelos números de 1 até M. A segunda linha contém inteiros $V_1, V_2, \ldots, V_M$, onde V_i representa o número de vagas no curso i. Nas N linhas seguintes segue as informações dos candidados na ordem em que se inscreveram. Cada uma dessas linhas contém, nesta ordem, um inteiro P indicando a pontuação do candidato, um inteiro Q indicando a quantidade de cursos que lhe interessa seguida pela lista desses cursos na ordem de sua preferência (do mais desejado ou menos).

A entrada deve ser lida da entrada padrão.

Saída

Para cada instância imprima N linhas, cada uma contendo o curso no qual cada aluno foi alocado. A ordem na qual os alunos são considerados é a mesma da entrada. Se o aluno não conseguiu vaga em nenhum dos seus cursos de interesse imprima -1.

Imprima uma linha em branco após cada instância.

A saída deve ser escrita na saída padrão.

Restrições

- $1 \leq N \leq 1000$
- $1 \leq M \leq 1000$
- $1 \leq V_i \leq 1000$
- $0 \leq P \leq 100$

Exemplos

Exemplo de entrada	Saída para o exemplo de entrada
4 2 5 2 87 1 2 89 2 2 1 88 2 2 1 40 2 1 2 3 2 1 1 99 2 1 2 100 1 1 99 2 2 1 4 3 1 2 1 76 3 1 2 3 76 3 1 2 3 76 3 1 2 3 76 3 1 2 3	-1 2 2 1 -1 1 2 1 2 2 3

Problema I: Solidariedade polonesa

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

A cidade de Gdansk ficou internacionalmente conhecida no início dos anos 80 por conta da atividade do sindicato Solidarność (Solidariedade), liderado por Lech Wałęsa, que viria anos depois a ser presidente da Polônia. Wałęsa enfrentou na época o regime comunista e iniciou um processo que contribuiu, no fim da década de 80 do século passado, com o fim do comunismo na Europa oriental.

Wałęsa era um líder presente e que nunca se atrasava. Ele era conhecido por frequentar inúmeras reuniões do sindicato, sem nunca sair durante uma reunião. No entanto, ele exigia que tivessem hora exata de início e término, para que pudesse preparar com antecedência sua programação. Cada um dos comitês divulgava o horário de início e término de sua reunião e Wałęsa escolhia as reuniões em que iria aparecer naquele dia de forma que ele pudesse participar do maior número possível de reuniões. No caso de haver mais de uma escolha na qual pudesse participar do maior número possível de reuniões, Wałęsa dava preferência para a escolha na qual as primeiras reuniões terminavam antes, na esperança de ter mais tempo para se preparar para a próxima, desempatando pela ordem da lista.

Sua tarefa neste problema é fazer um programa que recebe uma lista de reuniões e escolhe um subconjunto das reuniões de forma que elas não se sobreponham e este conjunto tenha cardinalidade máxima e seja o preferido de Wałęsa.

Entrada

A entrada é composta por diversas instâncias. A primeira linha da entrada contém um inteiro T indicando o número de instâncias.

A primeira linha de cada instância contém o número N de reuniões do sindicato. As reuniões são identificadas pelos números de 1 a N. As N linhas seguintes contêm, nesta ordem, os horários de início e fim da reunião no formato $HH : MM$. Todas as reuniões começam e terminam no mesmo dia. A $i + 1$-ésima linha corresponde à reunião i.

A entrada deve ser lida da entrada padrão.

Saída

Para cada instância imprima uma linha com a lista das reuniões escolhidas por Wałęsa. As reuniões deves ser impressas na ordem do seu horário de início.

A saída deve ser escrita na saída padrão.

Restrições

- $1 \leq N \leq 10000$
- $00 \leq HH \leq 23$
- $00 \leq MM \leq 59$

Exemplos

Exemplo de entrada	Saída para o exemplo de entrada
3 3 08:15 08:45 08:00 08:30 08:30 09:00 5 11:00 12:00 12:20 13:15 10:10 12:15 12:30 13:00 10:30 12:16 3 15:23 16:02 15:00 16:02 15:27 16:02	2 3 1 4 1

Problema J: RPG otimizado

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

Todo fim de semana, Pedro e seus amigos se reúnem para jogar RPG (do inglês *Role Playing Game*). Um jogo de RPG consiste em um dos amigos, denominado o mestre do jogo, contar uma história com a qual os outros participantes interagem de acordo com certas regras predefinidas e resultados de rolagem de dados.

Pedro, sendo o amigo mais dedicado à atividade, é sempre o mestre do jogo. Ele sempre planeja os jogos com antecedência de forma a entreter ao máximo seus amigos.

No próximo fim de semana, Pedro irá realizar a continuação do jogo do fim de semana anterior. Nesta continuação, N monstros numerados de 1 até N irão aparecer, um após o outro. A dificuldade de um monstro é determinada por um conjunto de M atributos (força, velocidade, inteligência, etc). Tudo o que Pedro deve decidir é contra quais destes monstros seus amigos deverão lutar. Uma sequência de monstros é válida se seus índices estão em ordem crescente. Os amigos de Pedro são muito exigentes e se sentirão chateados se lutarem contra um monstro que não é mais difícil do que todos os monstros contra os quais já lutaram. Um monstro é dito mais difícil que outro se ele tiver um valor maior em cada atributo do que o outro.

Pedro gostaria que o jogo durasse o máximo possível e então pediu para você, que participa da Maratona de Programação, ajudar a escolher um subconjunto de monstros com o maior número possível de elementos para seus amigos enfrentarem satisfazendo as restrições acima.

Entrada

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

A primeira linha de cada instância contém de dois números inteiros M e N, o número de atributos e o número de monstros. As próximas M linhas irão conter N inteiros cada. O j-ésimo inteiro na i-ésima destas linhas será o valor do i-ésimo atributo do j-ésimo monstro.

A entrada deve ser lida da entrada padrão.

Saída

Para cada instância seu programa deve imprimir uma linha contendo os índices dos monstros, separados por um espaço, em ordem de dificuldade. Se houver mais de uma resposta possível, qualquer uma será aceita.

A saída deve ser escrita na saída padrão.

Restrições

- $1 \leq M \leq 10$.
- $1 \leq N \leq 10^5$.
- Se $M > 2$ então $N \leq 1000$.
- O valor de cada atributo de cada monstro está entre 0 e 10^9.

Exemplos

Exemplo de entrada	Saída para o exemplo de entrada
3 4 1 2 9 7 1 3 4 9 1 9 5 6 4 4 0 3 3 4 0 1 2 4 0 2 1 4 0 3 3 4	1 3 1 2 4

Problema K: Trabalho do papa

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

Karol Wojtyła nasceu em Wadowice em 1920, foi eleito papa João Paulo II em 1978 e morreu em 2 de abril de 2005 após completar 26 anos de pontificado. Foi um dos poloneses mais importantes da História. Antes de se tornar religioso, durante a segunda grande guerra, Wojtyła trabalhou em uma fábrica de produtos químicos. Uma das tarefas de Wojtyła era acomodar os pacotes de produtos químicos em pilhas para transportá-los. O problema surgia do fato de que alguns destes pacotes eram muito pesados, e outros tinham pouca resistência a ter peso sobre eles. Os gerentes da fábrica sempre desejavam pilhas grandes, com o maior número possível de caixas. Todas as caixas tinham associados dois números: seu peso em quilos e o peso, também em quilos, que era capaz de suportar (incluindo o peso da caixa). Assim, uma caixa de 25 quilos e resistência de 60 quilos era capaz de suportar no máximo 35 quilos em cima dela.

Sua tarefa é, dada uma lista de caixas, com seus pesos e resistências, determinar o número máximo de caixas que podem ser empilhadas sem ultrapassar a resistência de nenhuma caixa da pilha.

Entrada

A entrada é composta por diversas instâncias. A primeira linha da entrada contém um inteiro T indicando o número de instâncias.

A primeira linha de cada instância contém um inteiro N, indicando o número de caixas. Cada uma das N linhas seguintes contém inteiros P e R, correspondendo, respectivamente, ao peso e resistência de uma caixa.

A entrada deve ser lida da entrada padrão.

Saída

Para cada instância imprima, em uma única linha, o maior número de caixas que podem ser empilhadas nas condições do enunciado.

A saída deve ser escrita na saída padrão.

Restrições

- $1 \leq N \leq 1000$
- $1 \leq P \leq 1000$
- $1 \leq R \leq 1000000$

Exemplos

Exemplo de entrada	**Saída para o exemplo de entrada**
2 4 13 30 11 15 7 28 6 6 3 43 60 47 100 3 2	3 2