Maratona de Programação de 2012

Instituto de Matemática e Estatística
Universidade de São Paulo

Caderno de Problemas

Departamento de Ciência da Computação IME-USP

Sábado, 18 de agosto de 2012.

Problema A: Festival das noites brancas

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

Todos os anos, na época das chamadas "noites brancas" em que o sol não se põe sobre a cidade de São Petersburgo ocorre o "festival de artes das noites brancas", que consiste de uma série de apresentações musicais, concertos, balés, e muito mais que atraem artistas de todo o mundo. É considerado uma das maiores manifestações populares de toda a Russia, uma vez que no auge das noites brancas, o festival costuma ter até um milhão de participantes circulando pelas ruas da cidade. O Teatro Mariinski recebe alguns dos melhores espetáculos e, uma vez que não tem ingressos suficientes para todos os que desejam assistir às performances, costuma utilizar um sistema curioso e divertido para sortear os que poderão entrar no teatro.

Cada pessoa que entra no teatro, interessado em assistir a uma apresentação escolhe uma fileira na qual gostaria de sentar e recebe um cartão com um número de 000 a 999 escrito nele. Este número é o código do sorteio daquela pessoa. Ao chegar à entrada o atendente verifica a situação da fila na qual a pessoa sentará. A fila é descrita por uma sequência de '1's e '0's, onde 1 indica cadeira livre e 0 indica cadeira ocupada. Essa sequência é então interpretada como a representação binária do número n. A pessoa entrará com seus acompanhantes se o n-ésimo número da sequência de Fibonacci terminar exatamente com o número escrito no seu cartão. Assim, por exemplo, se a descrição da fileira é 100 a pessoa só entrará se possuir o cartão com o número 003.

Entrada

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

Cada instância consiste em uma linha contendo uma descrição de fileira. A descrição de uma fileira é uma sequência de '1's e '0's, nunca começando com '0' (a primeira cadeira de todas as fileiras estão reservadas).

Saída

Para cada instância imprima os 3 dígitos que devem estar escrito no cartão para a pessoa entrar no teatro.

Restrições

Uma fileira não tem mais do que 10.000 (dez mil) cadeiras.

Exemplos

Exemplo de entrada	**Saída para o exemplo de entrada**
3 1 10 1010	001 001 055

Observação

Seja n um número inteiro positivo escrito na base decimal. O n-ésimo número, $f(n)$, da sequência de Fibonacci é definido da seguinte forma:

$$f(n) = \begin{cases} 1 & \text{se } n = 1 \text{ ou } n = 2 \\ f(n-1) + f(n-2) & \text{caso contrário} \end{cases}$$

Problema B: Pontes de São Petersburgo

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

Todos conhecem o famoso problema das pontes de Königsberg, cidade da Prússia que ficou famosa pelo problema resolvido por Euler ainda no século XVIII. Poucos conhecem, entretanto, o problema das pontes de São Petersburgo. A cidade de São Petersburgo localiza-se às margens do Rio Neva, e é cruzada por dezenas de pontes que ligavam as margens do rio às centenas de pequenas ilhas que o rio possui. Os moradores da cidade, conhecedores do famoso problema das pontes de Königsberg, criaram seu próprio problema. Os moradores sabem que existem K pontes na cidade, que são R regiões distintas na cidade e que cada ponte liga exatamente 2 regiões distintas da cidade. Os moradores querem saber se, para a cidade deles, é possível escolher algumas destas regiões tais que o número de pontes que incide em todas elas é igual a K. Note que, se duas destas regiões escolhidas tiverem uma ponte entre elas, esta ponte será contada duas vezes.

Entrada

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

A primeira linha de cada caso de teste contém dois números, R e K, o número de regiões e pontes da cidade, respectivamente. Por efeito de simplificação, as regiões são enumeradas de 1 até R, inclusive. A seguir temos K linhas, cada uma delas contendo dois números A e B, informando que existe uma ponte ligando as regiões A e B da cidade.

Saída

Para cada caso de teste imprima uma linha apenas com "S" (aspas apenas para evidenciar), se é possível escolhermos as regiões da maneira descrita anteriormente, ou "N" (idem), se não for possível.

Restrições

- $2 \leq R \leq 100$
- $1 \leq K \leq \frac{R \times (R-1)}{2}$

Exemplos

Exemplo de entrada	**Saída para o exemplo de entrada**
2 1 1 2 3 3 1 2 1 3 3 2	S N

Problema C: Myachowski, o futebol russo

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

Muitas origens diferentes são atribuídas ao futebol. A atividade mais antiga que se assemelha ao futebol era praticada na China entre os séc. III e II a.C. e chamava-se ts'uh Kúh (cuju), e consistia em jogar uma pequena bola com os pés para uma rede. No Japão existe o kemari, praticado até hoje em eventos culturais. Em Roma jogava-se o harpastum, e na Grécia o epislcyros. Com a descoberta do novo mundo descobriu-se também um jogo maia muito semelhante ao futebol, o pok ta pok que teria mais de 3 000 anos de história. Na idade média jogava-se em Florença o calcio florentino, que muitos reputam ser o berço do futebol moderno. Até hoje índios do Amazonas jogam um jogo muito semelhante em que uma bola é empurrada usando apenas a cabeça em direção às metas inimigas. Seja como for, é quase impossível dizer qual o jogo que deu origem ao futebol hoje jogado, cujas regras foram formalmente estabelecidas pelos ingleses no final do século XIX.

Pouco se tem notícia de um jogo russo, também ancestral do futebol e com regras bastante claras (como diria o Arnaldo). É o Myachowski, também conhecido como Otskok. O nome vem provavelmente de мяч (lê-se myach) que significa "bola" em russo. No jogo um jogador entra em um campo que é uma elipse fechada e deve acertar um buraco localizado na parede do campo. Porém, o ponto só é computado se a bola entra no buraco após ser chutada contra as paredes do campo, sendo desviada para dentro do buraco.

Dadas a posição inicial da bola, a direção na qual ela está se movimentando e a descrição do campo, sua tarefa é determinar os próximos dois pontos de contato da bola com a parede do campo. Considere que o centro do campo é a posição $(0, 0)$.

Entrada

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

Cada instância consiste em uma linha contendo 6 inteiros, X, Y, D_x, D_y, A e B. A posição inicial da bola é dada pelo ponto (X, Y) dentro da elipse e a direção pelo vetor (D_x, D_y). O campo tem o formato de uma elipse descrita pela equação

$$\frac{x^2}{A^2} + \frac{y^2}{B^2} = 1.$$

Saída

Para cada instância imprima duas linhas. A primeira linha deve conter o primeiro ponto de contato da bola com a parede do campo e a segunda linha deve conter o segundo ponto de contato. Um ponto de contato deve ser impresso como dois números racionais separados por um espaço. Imprima os números com exatamente 3 casas decimais.

Restrições

- $1 \leq A, B \leq 500$
- $-1000 \leq D_x, D_y \leq 1000$

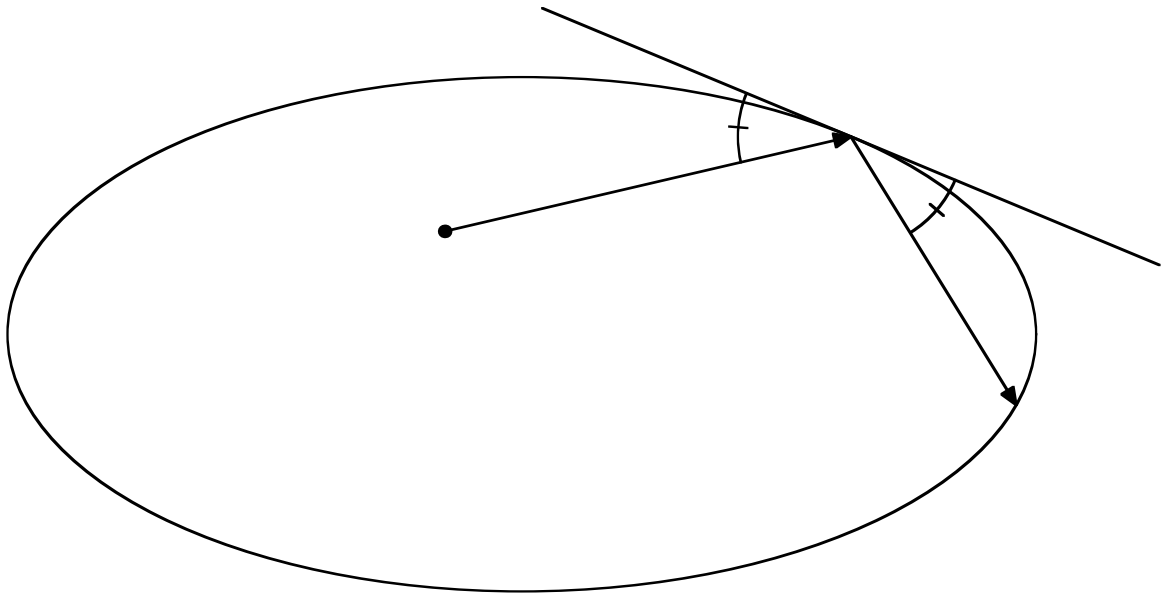

Figura 1: Ângulo de entrada é igual ao ângulo de saída

Exemplos

Exemplo de entrada	**Saída para o exemplo de entrada**
0 0 1 1 1 1 0 0 1 0 2 3 -30 40 30 7 200 100	0.707 0.707 -0.707 -0.707 2.000 0.000 -2.000 0.000 127.955 76.856 192.310 -27.464

Reflexão da bola

Suponha para esse problema que a bola é refletida perfeitamente quando atinge a parede do campo. Isso é, o ângulo que o vetor de entrada faz com a reta tangente à parede do campo no ponto de contato é o mesmo que ângulo de saída. A Figura 1 exemplifica esse comportamento.

Problema D: Cerco a Leningrado

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

A cidade de São Petersburgo mudou de nome depois da revolução russa em 1914 para Petrogrado. Após a morte de Lênin, em homenagem ao grande líder o nome da cidade mudou novamente para Leningrado em 1924, e assim permaneceu até o fim da União Soviética. Em 1991, a cidade voltou a ter o nome antigo. Durante a segunda guerra mundial a cidade de Leningrado sofreu um cerco das tropas alemãs que durou cerca de 900 dias. Foi uma época terrível, de muita fome e perdas humanas, que terminou em 27 de janeiro de 1944 com a vitória dos soviéticos. É considerada uma das vitórias mais custosas da história em termos de vidas humanas perdidas.

No auge da ofensiva alemã, no ano de 1942, vários atiradores de elite foram espalhados pela cidade, inclusive, em alguns pontos estratégicos da cidade mais de um atirador aguardavam soldados inimigos. A espionagem russa tinha informações detalhadas das habilidades desses atiradores, mas seus esconderijos eram excelentes, tornando a tarefa de um soldado soviético que desejasse cruzar a cidade extremamente difícil. Os soldados soviéticos eram bem treinados, mas com o passar do tempo e a continuação do cerco à cidade, os melhores soldados foram sendo dizimados, uma vez que se errassem o alvo na primeira tentativa certamente eram mortos pelos soldados alemães na tocaia.

Sabendo a probabilidade de um soldado em matar um atirador alemão e sabendo também o número de balas que ele tinha à sua disposição, desejamos saber a probabilidade desse soldado conseguir chegar a um ponto estratégico de destino, partindo de um ponto estratégico de origem. O soldado, sendo muito experiente, sempre usava um caminho que maximizava a probabilidade de sucesso. Note que o soldado deve matar todos os atiradores presentes no caminho usado, inclusive os que estiverem nos pontos estratégicos de origem e destino.

Entrada

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

A primeira linha de cada instância contém 3 inteiros, N, M, e K e a probabilidade P do soldado matar um atirador. Os inteiros N, M, e K representam respectivamente os números de pontos estratégicos, estradas ligando pontos estratégicos e balas carregadas pelo soldado soviético. Os pontos estratégicos são numerados de 1 a N.

Cada uma das próximas M linhas contém um par de inteiros i e j indicando que existe uma estrada ligando o ponto i ao j. Em seguida tem uma linha contendo um inteiro A, correspondendo ao número de atiradores na cidade, seguido por A inteiros indicando a posição de cada atirador.

A última linha de cada instância contém dois inteiros indicando o ponto de partida e o destino do soldado.

Saída

Para cada instância imprima uma linha contendo a probabilidade de sucesso do soldado soviético. A probabilidade deve ser impressa com 3 casas decimais.

Restrições

- $2 \leq N \leq 1000$
- $0 \leq K \leq 1000$
- $0 \leq A \leq 2000$
- $0 \leq P \leq 1$

Exemplos

Exemplo de entrada	Saída para o exemplo de entrada
3 2 10 0.1 1 2 2 3 10 1 1 3 3 1 3 1 1 3 3 1 3 5 5 10 0.3 1 2 2 4 2 5 4 5 5 3 6 3 3 3 3 3 3 1 3	0.000 0.001

Problema E: Desafio de São Petersburgo

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

A Russia sempre foi berço de grandes mestres de xadrez. Poucos sabem, mas a FIDE (Federação Internacional de Xadrez), que é o órgão máximo regulador do jogo de xadrez foi fundada em 1924, a partir de um movimento iniciado 10 anos antes no campeonato mundial da modalidade que ocorreu em São Petersburgo em 1914. Hoje, entre os 10 melhores jogadores do mundo, segundo a FIDE, três são russos.

O torneio de São Petersburgo ficou também conhecido pelas tentativas dos grandes mestres de popularização do jogo. Na época os maiores mestres (como Capablanca) foram às ruas propor desafios para as pessoas com o objetivo de interessá-las a praticar o jogo. Um desses desafios ficou conhecido como o desafio de São Petersburgo. O grande mestre montava uma situação em que as peças brancas tinham apenas o rei, e o objetivo era que a pessoa dissesse se o rei branco estava ou não em xeque mate.

Na situação descrita acima, o rei branco está em xeque mate se ele está sendo atacado e qualquer movimento que ele faça o leva para uma casa que também está sendo atacada.

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 indicando o número de peças pretas. A linha seguinte contém a descrição das posições das N peças pretas separadas por um espaço. A terceira linha contém a descrição do rei branco.

Uma descrição de uma peça de consiste em 3 caracteres. O primeiro indica se a peça é um peão (P), torre (T), bispo (B), rainha (R) ou rei (W). Note que o grande mestre não usava cavalos para facilitar para que ainda estava começando a aprender o jogo. O segundo caracter, entre 'a' e 'h', indica a coluna na qual a peça está e o terceiro, de '1' a '8' indica a linha.

Em nenhuma das instâncias ocorre a situação na qual o rei branco e o rei preto são adjacentes.

Saída

Para cada instância, imprima uma linha com a palavra `SIM`, se o rei branco está em xeque mate, ou a palavra `NAO`, caso contrário.

Restrições

- $2 \leq N \leq 10$

Exemplos

Exemplo de entrada	**Saída para o exemplo de entrada**
4 Tc3 Te4 Bf4 Wa8 Wd1 5 Wb5 Pf3 Rh7 Te1 Tg1 Wh1 4 Wa8 Ta1 Ta3 Rd2 We2	NAO SIM NAO

O que você precisa saber sobre xadrez

Considere que inicialmente as peças do jogador preto ficam nas linhas 7 e 8 enquanto as do jogador branco iniciam nas linhas 1 e 2. Não pode haver duas peças na mesma casa. As peças consideradas no problema (peão, torre, bispo, rainha e rei) não podem passar por cima de outras peças, ou seja, se durante sua movimentação existir alguma peça no seu caminho você deve parar antes ou atacar a peça (se ela for do oponente), tomando o seu lugar. A movimentação e o ataque de cada tipo de peça são da seguinte forma:

- Peão: anda apenas uma casa para frente (em direção a linha 1) podendo atacar em qualquer uma das duas diagonais imediatamente a sua frente;
- Torre: anda/ataca quantas casas quiser ou na horizontal, ou na vertical;
- Bispo: anda/ataca quantas casas quiser na diagonal;
- Rainha: anda/ataca quantas casas quiser ou na horizontal, ou na vertical, ou na diagonal;
- Rei: anda/ataca apenas uma casa ou na horizontal, ou na vertical, ou na diagonal.

Problema F: Os benefícios da vodka

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

São Peterburgo é conhecida como a capital da cerveja russa e abriga diversas cervejarias importantes. Dizem que a qualidade da água da cidade é responsável por uma cerveja de excelente qualidade. Além de fábricas tradicionais, como a Heineken, algumas marcas locais são destacadas, como a Tinkoff e a Baltika. Também na cidade são produzidas algumas das melhores vodkas do mundo. A mais antiga, chamada Liviz, data de 1897. Esta destilaria produz vodkas de excelente qualidade, medida por padrões internacionais. Curiosamente, alguns tipos de vodkas, quando consumidos juntos, acabam tendo, segundo os especialistas, sabor muito melhor. Dessa forma, alguns tipos de vodka são reunidos em categorias que, quando compradas totalmente pelo consumidor, trazem um benefício agregado medido segundo padrões internacionais de qualidade. Cada uma das vodkas tem um preço associado, e sua tarefa é encontrar uma compra que maximize o benefício total menos o custo das vodkas adquiridas.

Reescrevendo, cada vodka tem um custo C_j e existem M categorias diferentes, cada qual com um benefício B_i. Um benefício só é computado se todos os tipos de vodka que compõem a categoria são adquiridos. Uma mesma garrafa de vodka pode participar de mais de uma categoria para computar o benefício. Sua tarefa é determinar quais tipos de vodka comprar de forma a maximizar a soma dos benefícios adquiridos menos o custo dos itens comprados. Você pode supor que foi à Russia com dinheiro suficiente para comprar todos os tipos de vodka produzidos pela Liviz (oba!! :D)

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 representando, respectivamente, a quantidade de tipos diferentes de vodka a venda e o número de categorias existentes. Os tipos de vodka são identificados por números de 1 a N e as categorias por números de 1 a M.

A linha seguinte contém N inteiros, C_j, separados por espaço, correspondendo ao custo da vodka j. Na próxima linha existem M inteiros, P_i, separados por espaço, indicando quantos tipos diferentes de vodkas compõe a categoria i.

Cada uma das M linhas seguintes descreve uma categoria começando com um inteiro, B_i, indicando seu benefício, seguido pelos tipos de vodka que a compõe, separados por espaços.

Saída

Para cada instância imprima, em uma única linha, o maior valor que pode ser obtido da soma dos benefícios das categorias adquiridas menos o custo dos tipos de vodkas compradas.

Restrições

- $1 \leq N \leq 600$
- $1 \leq M \leq 400$
- $1 \leq C_j \leq 1000$ para $1 \leq j \leq N$
- $1 \leq P_i \leq N$ para $1 \leq i \leq M$
- $1 \leq B_i \leq 1000$ para $1 \leq i \leq M$

Exemplos

Exemplo de entrada	Saída para o exemplo de entrada
2 3 80 80 1 2 1 90 1 50 1 2 25 2 4 3 50 200 50 130 2 2 2 70 1 2 260 2 3 120 3 4	10 30

Observações

No primeiro exemplo o valor máximo pode ser obtido comprando apenas a vodka 1 e adquirindo a categoria 1. No segundo exemplo o valor máximo pode ser obtido comprando as vodkas 1, 2 e 3 e adquirindo as categoria 1 e 2.

Problema G: As dinastias de São Petersburgo

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

São Petersburgo foi fundada no dia 27 de maio de 1703 pelo czar Pedro, o Grande, e foi capital imperial da Rússia por um período curto logo após (de 1713 a 1728) e depois por quase dois séculos, de 1732 a 1918. Neste último período o trono imperial russo acabou sendo ocupado por diversos imperadores, muitas vezes de linhas de dinastia diferentes. Na tradição imperial russa chama-se de текущий (lê-se текущий [1]) uma sequência de descendentes dentro de uma dinastia, ou seja, um elemento, seu filho, seu neto, e assim por diante. A determinação destas текущий é fundamental quando se deseja determinar o sucessor do atual imperador, uma vez que o próximo imperador é o elemento vivo de uma текущий que esteja mais próxima do atual imperador. É claro que uma árvore genealógica pode ser dividida em текущий de várias formas diferentes. O interessante é encontrar uma partição que minimize o número de текущий necessário para cobrir todos os elementos da dinastia.

Sua tarefa neste problema é determinar, dada a árvore genealógica da família imperial russa, o menor número de текущий que particionam toda a família imperial, isso é, todos os imperadores tem que pertencer à exatamente uma текущий e essas têm que ser o menor número possível.

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 representando, respectivamente, a quantidade de imperadores e o número de relações de filiação naquela instância. Os imperadores são indentificados por números de 1 à N. Cada uma das próximas M linhas contém dois inteiros P_i e F_i, indicando que P_i é pai de F_i. Uma particularidade da da árvore genealógica dada é que em caso de dúvidas de paternidade, todos os possíveis pais eram indicados, ou seja, uma pessoa pode ter qualquer número de pais.

Saída

Para cada instância imprima uma linha contendo um único número inteiro, que é o número mínimo de текущий nescessários para particionar todos os imperadores daquela instância.

Restrições

- $1 \leq N \leq 1.000$
- $0 \leq M \leq 10.000$
- $1 \leq P_i < F_i \leq N$

[1] Russo é uma língua fonética ;)

Exemplos

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

Problema H: Festas de São Petersburgo

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

São Petersburgo tornou-se após o fim da cortina de ferro, no início dos anos 90, uma das principais cidades da cena alternativa em todo o mundo. Grupos de punks, diversas bandas de hardcore e outros representantes da cena alternativa mudaram-se para a cidade, atraídas pela grande quantidade de jovens. Com o surgimento das comunidades virtuais, alguns anos mais tarde, notou-se o enorme potencial do uso destas comunidades para combinar encontros, festas, raves, etc. Nestas festas de São Petersburgo é sempre muito importante que cada um dos participantes tenha pelo menos um certo número de amigos na rede social. E, ao mesmo tempo, desejamos convidar o maior número possível de pessoas de São Petersburgo desde que a restrição com relação ao número de amigos seja satisfeita. Tal restrição diz que, para ser convidada a festa, a pessoa precisa ter pelo menos um número K de amigos na lista de convidados.

Sua tarefa neste problema é, dado o conjunto de pessoas da comunidade e a lista de suas relações, determinar quais devem ser chamadas para que a festa tenha a maior quantidade possível de participantes satisfazendo a restrição.

Entrada

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

A primeira linha de cada instância contém três inteiros N, M e K representando respectivamente o número de pessoas na comunidade, o número de relações de amizade nessa comunidade e o número mínimo de amigos convidados uma pessoa precisa ter para ser convidada. Cada pessoa da comunidade é identificada por números de 1 a N. Cada uma das próximas M linhas contém um par de pessoas indicando que elas são amigas na rede social.

Saída

Para cada instância imprima uma única linha contendo a lista das pessoas a serem convidadas separadas por um espaço em branco. A lista deve estar ordenada em ordem crescente. Caso ninguém possa ser convidado, imprima o número 0.

Restrições

- $1 \leq N \leq 1000$
- $0 \leq K \leq N$

Exemplos

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

Problema I: Produção ótima de ótima vodka

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

A produção de vodka da cidade de São Petersburgo é famosa em todo o mundo. Conta a lenda que a vodka produzida é distribuída diretamente na casa de alguns dos funcionários mais graduados da empresa através do sistema de abastecimento de água. Ou seja, basta abrir a torneira e a vodka jorra geladinha (afinal os canos estão correndo a uma temperatura negativa na maior parte do ano) do cano. Isso causa diversos problemas de segurança, afinal as pessoas escavam as ruas procurando os supostos canos de vodka saindo da empresa.

Este não é o único problema enfrentado na produção de vodka da cidade. Para garantir o padrão de qualidade exigido da bebida, ela é produzida em apenas um destilador, que tem uma vida útil bem definida, de M anos. Sua manutenção varia dependendo da idade do equipamento. O custo de manutenção é C_i, onde i é a idade do destilador, e deve ser pago todo ano, até mesmo para destiladores novos. Estes destiladores têm um preço P quando comprados novos (idade 0) e os destiladores usados em fábricas russas são disputados por destilarias de todo o mundo, onde são usados ainda por muitos anos, e por museus. O preço de venda de um destilador com idade i é V_i. Note que um destilador com idade M não pode mais ser usado e deve ser vendido.

Sua tarefa neste problema é decidir em quais instantes a empresa deverá trocar o destilador de forma a minimizar o custo de produção ao final de N anos (a partir do ano 1). Considere que a troca de destiladores só pode ser feita no início do ano.

Entrada

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

A primeira linha de cada instância possui 4 inteiros, N, I, M e P representando, respectivamente, o período de produção, a idade inicial do destilador, a idade máxima do destilador e o preço de um destilador novo.

A linha seguinte contém M inteiros, separados por espaços, correspondendo ao custo de manutenção C_i, para $i = 0, 1, 2, \ldots, M-1$. A próxima e última linha contém M inteiros, separados por espaços, correspondendo ao valor de venda V_i, para $i = 1, 2, \ldots, M$.

Saída

Para cada instância a saída deve conter duas linhas. Na primeira, imprima o custo mínimo para o período dado. Na segunda, uma sequência crescente de inteiros, separados por espaços, indicando os anos nos quais são trocadas as máquinas. Se a máquina nunca é trocada, então imprima apenas um 0. Caso exista mais de uma sequência possível, escolha aquela na qual as máquinas são trocadas o mais cedo possível e sempre que possível (por exemplo, entre as sequências "1 4 7" e "1 2 8 10 14" escolha a segunda).

Restrições

- $1 \leq N \leq 2.000$
- $1 \leq M \leq 2.000$
- $1 \leq I \leq M$
- $1 \leq P \leq 1.000$
- $1 \leq C_i \leq 1.000$
- $1 \leq V_i \leq P$

Exemplos

Exemplo de entrada	Saída para o exemplo de entrada
4 2 6 100 30 50 65 80 100 120 60 50 40 30 20 10 5 5 6 200 1 100 100 100 100 200 50 100 100 100 100 100	260 1 3 501 1

Problema J: Lista telefônica econômica

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

Devido ao grande número de reclamações, a companhia telefônica de São Petersburgo está sendo obrigada a investir pesado na melhora de seus serviços. Para isso a companhia decidiu diminuir o orçamento de alguns setores para aumentar o de outros mais essenciais. Um dos setores que terá seu orçamento reduzido é o de impressão de listas telefônicas.

Com um orçamento reduzido, o setor de impressão de listas telefônicas não consegue comprar toner suficiente para imprimir as listas completas. Como os números de telefone são impressos alinhados na vertical, foi sugerida a seguinte solução: a partir do segundo número de telefone impresso, os dígitos iniciais do próximo número a ser impresso que coincidirem com os do número acima são omitidos, ficando apenas um espaço em branco. Por exemplo, para os números 535456, 535488, 536566 e 835456 a impressão é a seguinte:

```
5 3 5 4 5 6
        8 8
    6 5 6 6
8 3 5 4 5 6
```

Note que esta impressão economizou a impressão de 6 caracteres. A companhia telefonica cogitou também não imprimir os sufixos repetidos, mas nos testes feitos viram que a resposta não foi boa para o usuário e decidiram, portanto, fazer apenas a eliminação em prefixos. Para saber se a economia será suficiente, o setor de impressão quer saber o número máximo de caracteres que podem ser omitidos. No entanto, como em qualquer cidade grande, são vários os números telefônicos e eles não querem gastar homens-hora para calcular manualmente este valor. Então cabe a você, novo empregado da companhia, automatizar a economia feita pelo toner, no número de caracteres.

Entrada

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

Cada caso de teste contém um inteiro N, que informa o número de telefones na lista. As próximas N linhas possuem, cada uma delas, um telefone X_i, de até 20 caracteres. Para um mesmo caso de teste os números de telefone têm a mesma quantidade de caracteres. Um número de telefone pode começar com o caracter '0'.

Saída

Para cada caso de teste imprima uma linha informando o maior número possível de caracteres economizados por este processo.

Restrições

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

Exemplos

Exemplo de entrada	Saída para o exemplo de entrada
2 12345 12354 3 535456 535488 835456	3 4