8o Contest Noturno

6 de Junho de 2014

Sevidor CD-MOJ:

`http://moj.naquadah.tk`

Problemas:

Bruno César Ribas
Mateus Dantas
Ricardo Tavares de Oliveira
Christian Bonilha

Lembretes:

- É permitido consultar livros, anotações ou qualquer outro material impresso durante a prova.
- A correção é automatizada, portanto, siga atentamente as exigências da tarefa quanto ao formato da entrada e saída de seu programa. Deve-se considerar entradas e saídas padrão.
- Procure resolver o problema de maneira eficiente. Se o tempo superar o limite pré-definido, a solução não é aceita. As soluções são testadas com outras entradas além das apresentadas como exemplo dos problemas.
- Teste seu programa antes de submetê-lo. A cada problema detectado (erro de compilação, erro em tempo de execução, solução incorreta, formatação imprecisa, tempo excedido ...), há penalização de 20 minutos. O tempo é critério de desempate entre duas ou mais equipes com a mesma quantidade de problemas resolvidos.
- Para sumissões em JAVA a classe deverá ser chamada de Main.
- **CLARIFICATIONS** deverão ser enviadas para o email *clarification@naquadah.tk*. Os juízes podem opcionalmente atendê-lo com respostas acessíveis a todos.

Problema A: Gravando CDs

Arquivo: *gravando.[c|cpp|pas] ou Main.java*

Mas que azar, o bairro onde você mora está sem internet por tempo indeterminado, e justo agora que uma banda local gravou um CD com músicas originais e você e seus amigos queriam escutar. Se houvesse internet, todos poderiam baixar as músicas online no mesmo dia, porém como não há vocês terão que se virar emprestando cópias do CD e gravando outras.

Inicialmente há uma cópia do CD. Há N pessoas no seu bairro, e cada uma tem um determinado estoque de CD's virgens para gravar novas cópias.

Digamos que uma pessoa receba a cópia do CD no tempo t. Ela começará imediatamente a produzir tantas cópias quanto possível, de acordo com seu estoque, e tais cópias estarão disponíveis nos tempos t+1, t+2, ..., t+M, onde M é o número de CD's em estoque dessa pessoa.

Dado o número de CD's que cada pessoa tem em estoque em casa, encontre uma estratégia e diga qual o menor tempo necessário para que todos tenham em casa uma cópia do CD.

Entrada

Haverá diversos casos de teste. Cada caso de teste inicia com um inteiro N ($1 \leq N \leq 10^6$), indicando o número de pessoas no bairro.

Em seguida haverá N inteiros Si ($0 \leq Si \leq N$), indicando o número de CD's que a $i-$ésima pessoa tem em estoque, para todo $1 \leq i \leq N$.

O último caso de teste é indicado quando $N = 0$, o qual não deverá ser processado.

Saída

Para cada caso de teste imprima uma linha, contendo um inteiro, indicando o menor tempo necessário para que todas as pessoas do bairro tenham um cópia do CD em casa. Caso não seja possível que todas as pessoas tenham uma cópia do CD, imprima -1.

Exemplos

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

Problema B: Processamento de Imagem

Arquivo: `imagem.[c|cpp|pas] ou Main.java`

Com a proximidade da copa do mundo, as empresas de televisão estão trabalhando a todo vapor para transmitir o campeonato com a maior qualidade possível. Poucos sabem, mas a computação e matemática estão fortemente relacionada as tecnologias responsáveis pela transmissão dos jogos.

Uma exemplificação disso é o uso do XOR (Ou exclusivo), ele é bastante importante no processamento de imagens. Seu uso é importante por exemplo na detecção de mudanças em imagens.

Uma das maquinas responsáveis por tratar imagens é bem antiga, e seu algoritmo é histórico e lento. Essa maquina é uma arvore com N nós cada um possuindo um valor que recebe diversos pedidos de mudança em um certo intervalo desses valores. Explicitamente ela funciona da seguinte forma:

- A maquina nada mais é que uma arvore com N nós enraizada no nó de valor 1
- Cada nó possui um valor V_i diferente ou não dos demais
- A maquina pode receber os seguintes tipos de operação:
- MUDA xA: Para todo nó y, onde y é x ou y está contido na subárvore de x, faça $V_y = V_y$ XOR A
- SOMA x: Retorna a soma de todos os valores da subárvore de x, incluindo o mesmo.

Até então a maquina funcionava perfeitamente, porém com o avanço da tecnologia ela se tornou lenta e não consegue mais acompanhar em tempo real o processamento de imagem dos jogos. Sabendo disso, a Rede Bobo de Televisão convidou você como um programador nato para reprogramar a máquina tal que ela continue funcionando e da forma mais eficiente possível.

Entrada

A primeira linha da entrada contém dois inteiros N, M ($1 \leq N, M \leq 10^5$) representando a quantidade de nós na arvore e a quantidade de *queries* respectivamente.

A segunda linha da entrada contém N inteiros A_i ($1 \leq A_i \leq 10^8$) onde A_i é o valor inicial presente no nó i da arvore.

As próximas $N - 1$ linhas contem um par de inteiros A, B cada ($1 \leq A, B \leq N$) representando que existe uma ligação os nós A e B.

As próximas M linhas contém uma *query* cada, podendo ser:

- MUDA x A
- SOMA x

É garantido que o nó x é um nó da arvore e que o valor de A não excede 10^8.

Saída

Para cada *query* do tipo SOMA x, você deverá imprimir uma linha contendo soma de todos os valores no momento da subárvore de x, incluindo o mesmo.

Exemplo

Exemplo de entrada	Saída para o exemplo de entrada
4 3 1 2 3 4 2 1 3 1 4 3 SOMA 1 MUDA 3 10 SOMA 1	10 26

Problema C: Número Proibido

Arquivo: `proibido.[c|cpp|pas] ou Main.java`

Os números proibidos são números que possuem alguma representação problemática, por exemplo, número do azar, de algo ruim, e até números que são senhas do governo.

O número proibido mais conhecido é um número primo[1] que foi descoberto em 2001 e representa o arquivo binário da versão compactada do código C que implementa o algoritmo DeCSS, que pode ser utilizado para lograr o sistema de proteção do DVD.

Luan, um rapaz que tem muito receio de ser procurado por agências espiãs internacionais coletou um conjunto de números ilegais e está filtrando esses números de todos os seus arquivos no computador.

Infelizmente, Luan ainda não sabe programar muito bem e pediu a sua ajuda para implementar um programa que receba um conjunto de números ilegais e responda se um outro conjunto de números fazem parte dos números ilegais.

Entrada

A entrada é composta por um único caso teste que possui diversas linhas. A primeira linha possui um número N ($1 \leq N \leq 140000$) que representa a quantidade de números proibidos existentes. A segunda linha do caso de teste possui N números P_i ($0 \leq P_i \leq 2^{31}$) representando os números proibidos.

Depois existirão diversas linhas contendo um único número que se quer saber se é proibido ou não.

A entrada termina em EOF.

Saída

Para cada número da consulta deve-se imprimir uma única linha contendo a palavra `sim` se o número for proibido, ou `nao` caso o número não seja proibido.

Exemplo

Exemplo de entrada	**Saída para o exemplo de entrada**
7 10 0 50 25 121 0 3000 1 2 3 10 0	nao nao nao sim sim

[1] http://en.wikipedia.org/wiki/Illegal_prime

Problema D: Sanidade

Arquivo: `sanidade.[c|cpp|pas] ou Main.java`

Os Alunos de Algoritmos e Estruturas de Dados I, geralmente aprendem o conceito de listas encadeadas e durante o processo de aprendizado os professores, geralmente, pedem para que os alunos implementem as suas bibliotecas de listas encadeadas.

Um problema clássico nas listas é identificado quando os alunos implementam as listas duplamente encadeadas. Estas listas possuem apontadores para o próximo elemento e para o anterior. Geralmente as implementações dos alunos possuem alguns bugs e após algumas inserções e remoções a lista deixar de ficar sana, isto é, o caminho percorrido de um elemento qualquer PTR1 para outro elemento PTR2 deixa de ser o inverso do percurso de PTR2 para PTR1.

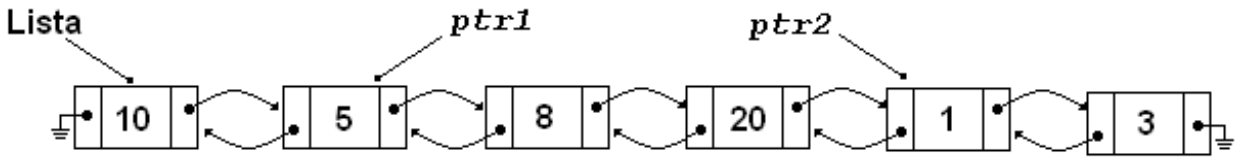


Figura 1: Exemplo de uma lista duplamente encadeada sana

Então para ajudar os alunos o professor definiu o problema da sanidade:

Definição 1 *Dado dois ponteiros PTR1 e PTR2 pertencentes a uma lista duplamente encadeada, que:*

- *Estes ponteiros existem e não serão passados valores nulos;*
- *Podem estar em qualquer posição da lista;*
- *PTR1 vem "antes" na lista, de PTR2;*
- *Pode existir qualquer quantidade de elementos antes de PTR1, depois de PTR2 e entre PTR1 e PTR2.*

Um subconjunto de uma lista duplamente encadeada é dita sana quando o caminho de PTR1 a PTR2 é o inverso do caminho de PTR2 a PTR1.

Para ajudar a identificar listas insanas, o professor fez um *patch* no *Valgrind* fazendo um *dump* da memória pertinente a lista encadeada, ou seja, com que todos os elementos da lista encadeada fossem impressos na tela contendo o endereço daquele elemento e os ponteiros para o próximo e anterior.

Infelizmente o professor já gastou muito tempo implementando o *patch* e precisa de sua ajuda para implementar um programa que leia o *dump* da lista duplamente encadeada e diga se a lista está sana ou não.

Entrada

A entrada é composta por um único caso de teste, contendo o *dump* da lista encadeada de um aluno. A entrada possui diversas linhas. Cada linha possui 3 números inteiros em formato hexadecimal representando, respectivamente, o endereço do elemento, o endereço do elemento anterior e o endereço do próximo elemento. O endereço 0 representa $NULL$.

As duas primeiras linhas do caso de teste representam os ponteiros de PTR1 e PTR2.

Nem todos os ponteiros fornecidos na entrada precisam, necessariamente, fazer parte do caminho PTR1->PTR2.

Para representar os ponteiros utilize inteiro sem sinal de 64bits.

Saída

A saída deverá conter uma única linha contendo a palavra `sana`, quando o caminho de PTR1 para PTR2 for o inverso de PTR2 para PTR1 e `insana` quando não.

Exemplos

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

`a 0 b`
`f e 0`
`e d f`
`b a c`
`c b d`
`d c e`
	`sana`

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

`facada dead babaca`
`dead 0 facada`
`babaca facada 0`
	`insana`

Problema E: Soma simples

Arquivo: `soma.[c|cpp|pas] ou Main.java`

Quando um número é muito grande, torna-se cansativo realizar toda uma soma. Cabe a você apenas dizer, então, se o resultado da soma de dois números é par ou ímpar.

Entrada

Haverá diversos casos de teste. A primeira linha contém um inteiro N, indicando o número de casos de teste que virão a seguir. Cada caso de teste conterá dois valores A e B ($1 \leq A, B \leq 10^{100}$), separados por um espaço, indicando os valores a serem somados.

Saída

Para cada caso de teste, deverá ser impressa uma linha, dizendo se o resultado da soma é par ou ímpar.

Exemplos

Exemplo de entrada	**Saída para o exemplo de entrada**
4 10 20 15 17 2 3 100000000000000000000000 10000000000000000000000	Par Par Impar Par

Problema F: Torcidas de satebol

Arquivo: `torcidas.[c|cpp|pas] ou Main.java`

O satebol é um esporte recém-criado por coreanos, russos e brasileiros que está conquistando seus primeiros praticantes e fãs pelo mundo. Para alavancar a popularização desse esporte em Nlogônia, A CNB (Confederação Nlognonense de Satebol) decidiu elaborar o primeiro campeonato nlogonense de satebol. Diversas equipes de satebol foram formadas no país, e irão disputar o campeonato em poucos meses.

A população de Nlognônia está bastante ansiosa para o início do campeonato. Entretanto, ela enfrenta um problema: como o esporte foi récem-criado, as equipes de satebol são muito novas, e nenhum nlognonense decidiu ainda para qual equipe torcer.

Toda pessoa nlognense aceita torcer para qualquer equipe do país, desde que:

- Cada pessoa deve torcer para *exatamente* uma equipe;
- Se duas pessoas distintas são amigas, então ambas devem *necessariamente* torcer para a *mesma* equipe;
- Se duas pessoas distintas não são amigas, então ambas devem *necessariamente* torcer para equipes *diferentes*.

Sua tarefa é, considerando o cenário acima, determinar se é possível associar uma equipe para cada pessoa torcer. Em caso positivo, determine o maior número possível de equipes para as quais há pelo menos um torcedor.

Considere que o número de equipes formadas para o campeonato é suficientemente grande, de forma que, se a criação de torcidas é possível, então, para cada pessoa, há garantidamente uma equipe para a qual ela pode torcer.

Entrada

A entrada consiste de vários casos de teste. Cada caso de teste inicia com um inteiro N ($1 \leq N \leq 10^3$), indicando a população de Nlognônia. As pessoas nlognenses são numeradas de 1 a N. As próximas N linhas contém N caracteres cada, não separados. O $i-$ésimo caractere da linha j é 1 se a pessoa i é amiga da pessoa j, e 0 caso contrário. É garantido que o $i-$ésimo caractere da linha j é igual ao $j-$ésimo caractere da linha i, e que o $i-$ésimo caractere da linha i é 0.

O último caso de teste é seguido de uma linha contendo um zero.

Saída

Para cada caso de teste, imprima o maior número possível de equipes para as quais há pelo menos um torcedor. Caso a criação de torcidas não seja possível, imprima `impossivel`.

Exemplos

Exemplo de entrada	Saída para o exemplo de entrada
6 001001 000110 100001 010010 010100 101000 4 0100 1010 0101 0010 0	2 impossivel

Problema G: Último a saber

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

Na cidade onde você mora as notícias se espalham rápido, ainda mais depois que todos começaram a utilizar redes sociais. Todo mundo lá gosta de conversar sobre as notícias, mas se tem uma coisa que eles não gostam é de serem os últimos a saber de algo. Geralmente, quando algo digno de comentários acontece, há um número K de pessoas presentes, as quais irão começar a contar os detalhes a todos os seus amigos. A notícia então se espalha da seguinte maneira: no primeiro dia há K pessoas que sabem da notícia; no segundo dia, todas as pessoas que são amigas de pelo menos uma das pessoas que sabia da notícia no dia passado ficará sabendo da notícia, caso ainda não saibam; nos dias seguintes, o processo se repete como no segundo dia. Todas as pessoas que souberam da notícia no último dia em que se foi falado sobre a notícia consideram-se as últimas a saber, e não gostam nada disso. Aqueles que nem ao menos ficaram sabendo da notícia não se sentem tão ofendidos. Sua tarefa é descobrir quem foram os últimos a saber.

Entrada

Haverá diversos casos de teste. Cada caso de teste inicia com três inteiros N, K e M ($2 \leq N \leq 10^4, 1 \leq K < N, 1 \leq M \leq 10^5$), indicando o número de pessoas na cidade, o número de pessoas que sabiam da notícia no primeiro dia e o número de relacionamentos de amizade na cidade, respectivamente. Em seguida haverá K inteiros distintos ki ($1 \leq ki \leq N$), indicando o índice das pessoas que sabiam da notícia no primeiro dia. Em seguida haverá M linhas, contendo dois inteiros A e B ($1 \leq A, B \leq N$) cada, indicando que as pessoas com índice A e B são amigas. O último caso de teste é indicado quando $N = K = M = 0$, o qual não deverá ser processado.

Saída

Para cada caso de teste imprima uma linha, contendo R inteiros separados por um espaço em branco, onde cada um dos inteiros representa o índice de uma das R pessoas que foram as últimas a saber da notícia. Imprima os índices em ordem crescente. Note que não deverá haver um espaço em branco após o último inteiro.

Exemplos

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

Problema H: Onde está Zelda?

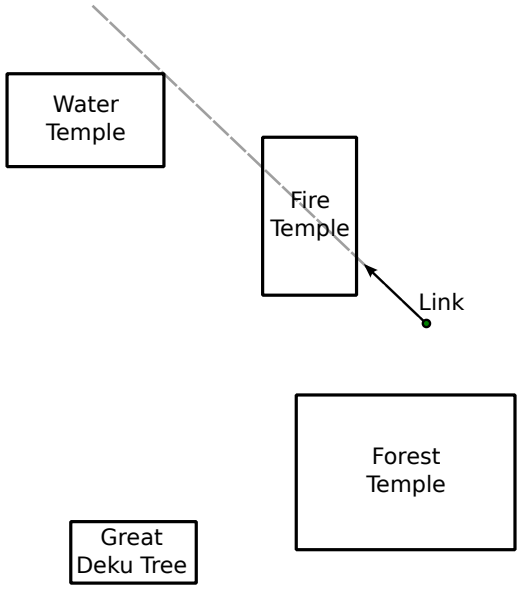
Arquivo: `zelda.[c|cpp|pas] ou Main.java`

O mundo de Hyrule está em perigo mais uma vez, e Link, novamente, está tentando salvá-lo. Para tal, Link precisa se encontrar com a princesa Zelda, para que ela possa lhe ajudar com seus grandes poderes místicos. Quando se encontrarem, Link e Zelda, juntos, irão derrotar o mal de uma vez por todas.

Link sabe que Zelda está sendo mantida refém dentro de algum dos N *templos* de Hyrule. Se Hyrule for projetada em um plano cartesiano, um templo consiste em um retângulo cujos lados são paralelos aos eixos.

Infelizmente, Link não sabe em qual templo Zelda está sendo mantida. Entretanto, sua espada tem uma habilidade especial (chamada *Dowsing*) que permite a Link saber a *direção* e o *sentido* em que Zelda está, a partir de sua própria localização.

A figura abaixo, que representa o segundo exemplo da entrada, exemplifica a situação. A espada de Link indica que Zelda está na direção e sentido do vetor destacado em preto. Logo, Link sabe que Zelda está em algum lugar sobre o segmento pontilhado.



Sua tarefa é fornecer a Link a lista dos templos nos quais Zelda pode estar. Note que, se Zelda estiver sobre alguma borda (aresta ou vértice) de um templo, então considera-se que ela está dentro dele (note que, no exemplo, Zelda pode estar na *Water Temple*).

Entrada

Cada caso de teste começa com uma linha contendo um inteiro N ($1 \leq N \leq 100$), o número de templos em Hyrule. A segunda linha contém dois inteiros x_L e y_L ($0 \leq x_L, y_L \leq 10^3$), as coordenadas da localização de Link. A terceira linha contém dois inteiros x_D e y_D ($-10^3 \leq x_D, y_D \leq 10^3$) indicando o vetor da direção indicada pela espada de Link.

As próximas N linhas contém, cada uma, a descrição de um templo. Cada templo é descrito com uma linha contendo *nome*, x_1, y_1, x_2 e y_2. *nome* é uma *string* com 1 a 20 letras maiúsculas ou minusculas, indicando o nome do templo. O par (x_1, y_1) e o par (x_2, y_2) indicam as coordenadas de vértices opostos do templo ($0 \leq x_1 < x_2 \leq 10^3$ e $0 \leq y_1 < y_2 \leq 10^3$).

É garantido que:

- O vetor de direção não é nulo (isto é, $(x_D, y_D) \neq (0, 0)$);
- A lista de templos em que Zelda pode estar não é vazia;
- Link não está dentro de nenhum templo;
- Não há interseção entre nenhum par de templos.

A entrada termina com $N = 0$.

Saída

Para cada caso de teste, imprima uma linha com os nomes dos templos nos quais Zelda pode estar. Imprima os nomes na ordem em que são apresentados na entrada, separados por espaços (mas não imprima um espaço depois do último nome).

Exemplos

Exemplo de entrada	**Saída para o exemplo de entrada**
2 4 6 3 0 DodongoCavern 2 2 5 4 JabuJabusBelly 7 4 10 8 4 14 10 -1 1 WaterTemple 1 15 6 18 ForestTemple 10 3 17 8 FireTemple 9 11 12 16 GreatDekuTree 3 2 7 4 0	JabuJabusBelly WaterTemple FireTemple

Problema I: Julgamento por Combate

Arquivo: `julgamento.[c|cpp|pas] ou Main.java`

Tyrion Lannister está sendo acusado de um crime terrível e exigiu que seu julgamento seja realizado *por combate*.

Tyrion vive em Westeros. De acordo com as leis de Westeros, em um julgamento por combate, ambos réu e acusador escolhem guerreiros para combaterem entre si. Se os guerreiros escolhidos pelo réu vencerem o combate, o réu é absolvido; entretanto, se os guerreiros escolhidos pelo acusador vencerem, então o réu é condenado.

Pela lei de Westeros[2], o réu Tyrion pode escolher qualquer subconjunto $S \subseteq \{g_1, g_2, ..., g_N\}$ dos N guerreiros disponíveis em Westeros para lutar por si. Cada guerreiro g_i $(1 \leq i \leq N)$ tem dois inteiros associados a ele: sua *força natural* F_i e seu *fator motivacional* M_i. Se um dado subconjunto S de guerreiros é escolhido, então um guerreiro $g_i \in S$ entrará no campo de batalha com uma força igual a $F_i + |\{g_j \in S : F_j > F_i\}| \times M_i$. Em outras palavras, sua força natural será acrescida de seu fator motivacional multiplicado pelo número de guerreiros com força natural maior que a sua no conjunto S, já que o guerreiro pode se motivar ao lutar ao lado de guerreiros melhores.

Assim, por exemplo, se um guerreiro $g_i \in S$ tem força natural 10 e fator motivacional 2, sua força no campo de batalha será igual a 14 se houver outros 2 guerreiros com força natural maior que 10 lutando a seu lado em S. Note que é possível que o fator motivacional de um guerreiro seja negativo, caso no qual o guerreiro se desmotiva ao lado de guerreiros maiores. Assim, se o mesmo guerreiro tiver fator motivacional -2, sua força em batalha será igual a 6. Note que é possível, inclusive, que um guerreiro entre em batalha com força negativa.

A *força total* de um conjunto de guerreiros S é dada pela soma das forças dos guerreiros de S em batalha. Tyrion pediu sua ajuda para determinar qual subconjunto $S \subseteq \{g_1, g_2, ..., g_N\}$ ele deve escolher de forma a maximizar sua força total, aumentando assim suas chances de vitória. Note que Tyrion pode, se desejar, não escolher nenhum guerreiro, caso no qual a força total dos guerreiros escolhidos é igual a 0 (neste caso, ele mesmo irá lutar).

Entrada

A entrada contém vários casos de teste. Cada caso de teste inicia com uma linha contendo um inteiro N $(1 \leq N \leq 10^3)$ indicando o número de guerreiros disponíveis. A segunda linha contém N inteiros *distintos* F_i $(0 \leq F_i \leq 10^6)$ separados por espaço, indicando a força natural dos guerreiros $g_1, g_2, ..., g_N$. Por fim, a terceira linha contem outros N inteiros M_i $(-10^3 \leq M_i \leq 10^3, M_i \neq 0)$ indicando o fator motivacional dos guerreiros.

A entrada termina com $N = 0$.

[2]na verdade, pela lei deste problema

Saída

Para cada caso de teste, imprima uma única linha contendo a força total do subconjunto que deve ser escolhido por Tyrion.

Exemplos

Exemplo de entrada	Saída para o exemplo de entrada
2 7 10 2 4 3 10 20 30 -10 -50 100 0	19 30