Seletiva para Maratona de Programação - 2007

Universidade de São Paulo

Caderno de Problemas

Departamento de Ciência da Computação IME-USP

Domingo, 19 de agosto de 2007.

Problema A: Recuperação

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

A nossa grandiosa Professora Cris no último aquecimento ficou conhecida como a grande maquiavélica do IME. Para quem não está a par do assunto, a digníssima professora exigiu que os alunos formassem uma fila em ordem lexicográfica (pelo nome) com no máximo k permutações. Isto fez com que muitos alunos nem sequer entrassem na sala para fazer a prova. No entanto, nesta seletiva ela resolveu se redimir perante seus alunos, e resolveu aplicar um probleminha para recuperação.

Sua tarefa, mesmo não tendo sido reprovado, é dada uma sequência de n inteiros $a_1, a_2, .., a_n$, onde $-30 \leq a_j \leq 30$ para $j = 1, 2, .., n$, imprima, se existir, um inteiro a_k tal que $a_k = \sum_{i=1}^{k-1} a_i$. Se houver mais de um inteiro que satisfaça esta condição, imprima o que aparece primeiro na seqüência.

Cris: *"Meninos, lembrem-se que a soma de nenhum número é zero! Tá?"*

Entrada

A entrada é composta de diversas instâncias. A primeira linha de cada instância consiste em um inteiro n ($1 \leq n \leq 100$) indicando o número de inteiros na linha seguinte devem ser processados.

A entrada termina com final de arquivo.

Saída

Para cada instância, você deverá imprimir um identificador `Instancia k`, onde k é o número da instância atual. Na linha seguinte imprima o inteiro que satisfaça a restrição descrita acima. Caso não exista tal inteiro imprima `nao achei`.

Após cada instância imprima uma linha em branco.

Exemplo de entrada

```
1
0
7
1 2 3 4 5 6 7
```

Exemplo de saída

```
Instancia 1
0

Instancia 2
3
```

Problema B: Quem vai ser reprovado?

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

Prof. Wallywow da Universidade da Columbia Britânica está muito preocupado com a queda do nível de atenção de seus estudantes. Ele já tentou várias técnicas mundialmente conhecidas para incentivar os alunos a prestar atenção nas suas aulas e fazer as tarefas que ele passa para a turma: deu nota para os alunos mais participativos, ofereceu chocolates aos alunos, levou seu karaokê e cantava nas aulas etc. Como tais medidas não levaram a uma melhora no comparecimento às aulas (a idéia do karaokê, inclusive, mostrou-se bastante infeliz... na segunda aula com karaokê a turma reduziu-se a um aluno – que tinha problemas auditivos) ele teve uma brilhante idéia: faria uma competição entre os alunos.

Prof. Wallywow passou um conjunto de problemas aos alunos, e deu um mês para que eles os resolvessem. No final do mês os alunos mandaram o número de problemas resolvidos corretamente. A promessa do brilhante didata era reprovar sumariamente o último colocado da competição. Os alunos seriam ordenados conforme o número de problemas resolvidos, com empates resolvidos de acordo com a ordem alfabética dos nomes (não há homônimos na turma). Isso fez com que alunos com nomes iniciados nas últimas letras do alfabeto se esforçassem muito nas tarefas, e não compartilhassem suas soluções com colegas (especialmente aqueles cujos nomes começassem com letras anteriores). Sua tarefa neste problema é escrever um programa que lê os resultados dos alunos do Prof. Wallywow e imprime o nome do infeliz reprovado.

Qualquer semelhança entre o Prof. Wallywow e o Prof. Carlinhos é mera coincidência.

Entrada

A entrada é composta de diversas instâncias. A primeira linha de cada instância consiste em um inteiro n ($1 \leq n \leq 100$) indicando o número de alunos na competição. Cada uma das n linhas seguintes contém o nome do aluno e o número de problemas resolvidos por ele. O nome consiste em uma seqüência de letras `[a-z]` com no máximo 20 letras e cada time resolve entre 0 à 10 problemas.

A entrada termina com final de arquivo.

Saída

Para cada instância, você deverá imprimir um identificador `Instancia k`, onde k é o número da instância atual. Na linha seguinte imprima o nome do infeliz reprovado.

Após cada instância imprima uma linha em branco.

Exemplo de entrada

```
4
cardonha 9
infelizreprovado 3
marcel 9
infelizaprovado 3
```

Exemplo de saída

```
Instancia 1
infelizreprovado
```

Problema C: Não é Mais um Joguinho Canadense!

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

O Canadá é um país muito frio. Em 8 meses por ano as temperaturas praticamente impedem que as ruas sejam ocupadas por vida inteligente, restando apenas criaturas resistentes ao frio como alces, ursos e canadenses[1]. Nestes longos meses de inverno famílias buscam diversão em frente de suas lareiras (ou, para as mais corajosas, ao redor de suas fogueiras). A família Smith, de Banff, inventou o jogo que descrevemos a seguir.

A brincadeira começa com uma das crianças desenhando um diagrama com estados (representados por bolinhas) ligados por transições (flechas ligando os estados). Cada transição tem uma letra e um número associados. Podemos fazer diversos passeios neste diagrama, partindo de um estado inicio caminhando por suas transições e terminando em um estado final. Um passeio forma uma palavra (obtida da concatenação das letras das transições percorridas) e tem um custo (que é dado pelo produto dos números destas transições).

Exemplo. Considere o diagrama abaixo.

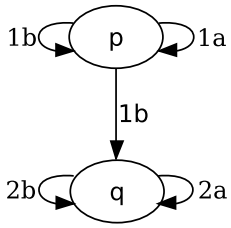

Figura 1: Diagrama

Todos os passeios iniciam no estado p e terminam em q.

O passeio que segue pelas transições `(p,1a), (p,1a), (p,1b)` e termina no estado q forma a palavra `aab` concatenando as letras de cada transição tem custo 1 (produto dos números destas transições).

O passeio que segue pelas transições `(p,1a), (p,1a), (p,1b), (q,2b)` e termina no estado q forma a palavra `aabb` e tem custo 2.

O jogo inventado pelo papai Smith era o seguinte. Depois de desenhar um diagrama como esse, um dos membros da família falava uma palavra, e os outros deveriam descobrir a soma dos custos de todos os passeios no diagrama que formam a palavra dada tais que iniciam no estado p e terminam no estado q. No caso do exemplo do diagrama acima, se o Sr. Smith pedisse a palavra `aba` a resposta deveria ser 2.

Entrada

A entrada é composta de diversas palavras (o diagrama é sempre o da figura). Cada instância é dada por uma linha contendo uma palavra. Uma palavra é uma seqüência de letras `[a,b]` com no máximo 60 letras.

A entrada termina com final de arquivo.

Saída

Para cada instância, você deverá imprimir um identificador `Palavra k`, onde k é o número da instância atual. Na linha seguinte imprima a soma dos custos.

Após cada instância imprima uma linha em branco.

[1] He he he, brincadeirinha...

Exemplo de entrada

```
a
b
ab
ba
aaaa
bbbb
aabb
abbb
```

Exemplo de saída

```
Palavra 1
0

Palavra 2
1

Palavra 3
1

Palavra 4
2

Palavra 5
0

Palavra 6
15

Palavra 7
3

Palavra 8
7
```

Problema D: Cardápio da Sra. Montagny!

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

Sra. Montagny é uma socialite de Quebec, que passa as férias em Banff, na sua mansão à beira do Lake Louise. Seus jantares são famosos porque ela com antecedência passa um questionário aos convidados onde os mesmos participam da escolha do cardápio. No questionário, a famosa magnata lista todos os pratos que poderá fazer no jantar, oferecendo uma coluna para o convidado selecionar o prato e outra para vetá-lo. É permitido fazer apenas duas escolhas no questionário, ou seja, cada convidado pode selecionar um prato e vetar outro, vetar dois pratos ou selecionar dois pratos. A Sra. Montagny garante que todos os convidados terão pelo menos um de seus desejos atendidos.

Antigamente ela mesma dava conta de montar o cardápio e atender o que prometia, mas com o crescimento de suas festas isso tem se tornado impossível. Assim, ela resolveu contratar vocês para fazer um programa que recebe os pedidos dos convidados e responde se é montar um possível cardápio para a festa.

Entrada

A entrada é composta de diversas instâncias. Cada instância começa com um inteiro n ($1 \leq n \leq 1000$), indicando a quantidade de questionários recebidos pela Sra. Montagny. Cada uma das próximas n linhas contém dois nomes de comida indicando a preferência de cada convidado. Um nome de comida é uma seqüência de letras `[a-z]` com no máximo 20 letras. Quando o nome de uma comida é iniciado por `‘‘!’’` significa que o convidado deseja vetar a comida, caso contrário ele deseja selecionar.

Saída

Para cada instância, você deverá imprimir um identificador `Instancia k`, onde k é o número da instância atual. Na linha seguinte você deve imprimir `sim` se for possível atender pelo menos um desejo de cada convidado e `nao` caso contrário.

Após cada instância, seu programa deve imprimir uma linha em branco.

Exemplo de entrada

```
2
!feijoada !file
rabada feijoada
4
arroz churrasco
!arroz !churrasco
arroz !churrasco
!arroz churrasco
```

Exemplo de saída

```
Instancia 1
sim

Instancia 2
nao
```

Problema E: Mesa da Sra. Montagny!

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

Já comentamos as festas da Sra. Montagny à beira do Lake Louise em Banff. Nas suas festas ela se compromete a resolver um outro problema que faz tremer organizadores de jantares em todo o mundo: onde sentar os convidados. A magnata simplifica bastante o problema pedindo aos convidados, no mesmo questionário já comentado, que anote na lista dos convidados aqueles que desejariam ter à sua frente na mesa do jantar. A idéia é ter seus amigos sempre à sua frente, para que a conversa possa fluir melhor. A habilidade da socialite é tamanha que ela foi contratada pelo Fairmont Banff Springs hotel (hotel em que vão ocorrer as finais mundiais do ICPC em 2008: *http://en.wikipedia.org/wiki/Banff_Springs_Hotel*) para trabalhar no arranjo de mesas de banquete.

Sua tarefa neste problema é auxiliar novamente a magnata. Dados os desejos dos convidados, seu programa deve decidir se é possível dispô-los numa mesa de forma que cada convidado tenha todos os seus amigos no lado oposto da mesa.

Entrada

A entrada é composta de diversas instâncias. A primeira linha de cada instância contém dois inteiros n ($1 \leq n \leq 100$) e m ($0 \leq m \leq \frac{n(n-1)}{2}$), onde n é o número de convidados e m é o número de relações de amizade. Cada uma das m linhas seguintes contém dois inteiros u e v indicando que u é amigo de v e v é amigo de u, onde $1 \leq u, v \leq n$.

A entrada termina com final de arquivo.

Saída

Para cada instância, você deverá imprimir um identificador `Instancia k`, onde k é o número da instância atual. Na linha seguinte imprima `sim` se é possível e `nao` caso contrario.

Após cada instância imprima uma linha em branco.

Exemplo de entrada

```
3 3
1 2
2 3
1 3
4 3
1 2
1 3
1 4
```

Exemplo de saída

```
Instancia 1
nao

Instancia 2
sim
```

Problema F: Sistema Cipoviário

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

Os pesquisadores do departamento de pesquisa operacional da Universidade da Columbia Britânica foram contratados para uma estranha tarefa. Vários países da África resolveram se unir e utilizar oficialmente o meio de transporte que ficou mundialmente conhecido nos filmes do Tarzan: o cipó. Há milhões de cipós na África e é surpreendente com que velocidade e eficiência uma pessoa pode se deslocar na selva utilizando esse meio de transporte. Só surgiu um pequeno problema. Os cipós são dominados por três grandes tribos: os makelelês, os malouhdás e os abedis. As tribos exigem ser pagas por cipó usado no sistema de transporte. Como eles ainda não sabem o significado de palavras como cartel, cada uma fez o seu preço, e divergiram bastante. Enquanto os makelelês exigem 1235 bongôs por cipó usado, os malouhdás exigem 8977 e os abedis 10923 (a Jane ainda está viva, e ajudou a intermediar a negociação para esta tribo).

Os pesquisadores foram contratados para escolher os cipós que comporão o primeiro sistema cipoviário do mundo. Os contratantes construíram milhões de "pontos de cipó" pela selva africana e desejam que os cipós sejam escolhidos de tal forma que seja possível ir de qualquer ponto a qualquer outro usando os cipós contratados (você pode ter de trocar de cipó algumas vezes, como fazia o Tarzan). Você deve dizer qual o custo de um sistema que atenda estes requisitos e seja o mais barato possível.

Você pode supor que existam cipós suficientes na selva para que sempre exista um sistema cipoviário que atenda os requisitos.

Entrada

A entrada é composta de diversas instâncias. A primeira linha de cada instância contém dois inteiros n ($1 \leq n \leq 1000$) e m ($1 \leq m \leq 2000000$), onde n é o número de "pontos de cipó" e m é o número de cipós. Cada uma das m linhas seguintes contém três inteiros u, v e c indicando que existe um cipó que vai do ponto u e até o ponto v com custo c, onde $1 \leq u, v \leq n$ e $c \in \{1235, 8977, 10923\}$.

A entrada termina com final de arquivo.

Saída

Para cada instância, você deverá imprimir um identificador `Instancia k`, onde k é o número da instância atual. Na linha seguinte imprima o custo de um sistema que atenda os requisitos descritos acima.

Após cada instância imprima uma linha em branco.

Exemplo de entrada

```
3 3
1 2 10923
1 3 1235
2 3 1235
3 2
1 2 1235
2 3 10923
```

Exemplo de saída

```
Instancia 1
2470

Instancia 2
12158
```

Problema G: Números de Dinostratus

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

Descobertas arqueológicas recentes de pesquisadores da Universidade de Alberta, no Canadá, mostraram que uma estranha seqüência de números eram encontrados nas paredes das pirâmides do Egito, nas ruínas de Macchu Picchu e nas pedras de Stonehenge. Intrigados com a aparente coincidência os pesquisadores acionaram o Departamento de Matemática para decifrar o que aquela seqüência ou aqueles números tinham de especial.

A descoberta foi estarrecedora. Todos os números eram gerados por matrizes de Dinostratus. Dinostratus foi um famoso matemático grego que viveu de 390 à 320 a.C. e trabalhou em importantes problemas de geometria como a quadratura do círculo. Dinostratus estudava matrizes M de dimensão 3×3 formada por 9 inteiros distintos com a propriedade que para toda posição $(i, j), i = 1, \ldots, 3, j = 1, \ldots, 3$ da matriz o elemento $m_{i,j}$ é múltiplo dos seus vizinhos $m_{i-1,j}$, $m_{i-1,j-1}$ e $m_{i,j-1}$ (quando existirem). Em sua homenagem, dizemos que n é um **número de Dinostratus** se existir uma matriz M com a propriedade acima em que $m_{3,3} = n$.

Veja um exemplo com $n = 36$.

1	2	4
3	6	12
9	18	36

A relação entre os números de Dinostratus, as pirâmides do Egito, as pedras do Stonehenge e as ruínas de Macchu Picchu ainda permanece um grande mistério. Mas, os pesquisadores de Alberta estão dispostos a estudar estes números mágicos. Sua tarefa é fazer um programa que recebe um inteiro n e verifica se este é um número de Dinostratus.

Entrada

A entrada é composta de diversas instâncias. Cada instância é dada por uma linha contendo um inteiro n ($1 \leq n \leq 1048576$).

A entrada termina com final de arquivo.

Saída

Para cada instância, você deverá imprimir um identificador `Instancia k`, onde k é o número da instância atual. Na linha seguinte imprima `sim` se n é um número de Dinostratus, caso contrário imprima `nao`.

Após cada instância imprima uma linha em branco.

Exemplo de entrada

```
36
37
38
```

Exemplo de saída

```
Instancia 1
sim

Instancia 2
nao

Instancia 3
nao
```

Problema H: Final Mundial de 2008

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

Preocupado com a atual situação de crise no transporte aéreo, o diretor regional do concurso do ICPC no Brasil já iniciou seus preparativos para fazer as reservas das passagens aéreas para as finais mundiais de Banff em 2008. O primeiro passo foi estudar a malha aérea disponível, em que cada vôo tem um certo preço e liga duas cidades (estamos, na verdade, chamando de vôo apenas um trecho *non stop* de um vôo comercial). O objetivo do diretor é fazer várias consultas nesta malha de vôos.

Em geral desejamos fazer vôos sem escalas, mas estes podem ser muito caros. Para contornar este fato o diretor deseja permitir algumas escalas possíveis. Assim, ele ordenou as várias cidades da malha em sua ordem de preferência para fazer escala. Ou seja, a cidade de índice 1 é a que ele prefere fazer escala, seguida pela cidade 2, e assim por diante.

As consultas que o diretor fará são, então do seguinte tipo. É dada a cidade de partida e de chegada e um número t de cidades em que o diretor permite que sejam feitas escalas. Seu programa deverá encontrar o custo de um vôo de custo mínimo entre as cidades que faça, no máximo, escalas nestas cidades. Por exemplo, se $t = 1$ você deverá encontrar o custo de um vôo de custo mínimo entre as duas cidades que seja, ou *non stop* ou que faça uma escala na primeira cidade.

Entrada

A entrada é composta de diversas instâncias. A primeira linha de cada instância consiste em dois inteiros n ($1 \leq n \leq 100$) e m ($1 \leq m \leq 100000$), indicando o número de cidades e o número de escalas. Nas m linhas seguintes temos três inteiros u, v e w ($1 \leq u, v \leq n$ e $0 \leq w \leq 100$) indicando que existe uma escala que vai de u para v com custo w. Em seguida um inteiro c ($1 \leq c \leq 10000$) indicando o número de consultas, e nas c linhas seguintes temos três inteiros o, d e t ($1 \leq o, d \leq n$ e $1 \leq t \leq n$) onde o é a cidade de origem, d é a cidade de destino e t indica que as cidades $1, 2, .., t$ podem ser usadas para escalas.

A entrada termina com final de arquivo.

Saída

Para cada instância, você deverá imprimir um identificador `Instancia k`, onde k é o número da instância atual. Para cada consulta, na ordem da entrada, você deve imprimir o custo mínimo ou `-1` caso não exista caminho entre as duas cidades.

Após cada instância imprima uma linha em branco.

Exemplo de entrada

```
4 7
4 1 0
2 1 3
1 4 20
2 3 15
4 2 1
3 1 21
1 2 0
3
2 1 0
4 2 2
4 3 1
5 10
4 5 2
2 1 4
1 2 7
2 4 7
5 2 1
4 1 2
4 5 12
5 4 4
5 3 7
3 5 9
4
2 5 0
3 4 5
4 5 1
2 3 2
```

Exemplo de saída

```
Instancia 1
3
0
-1

Instancia 2
-1
13
2
-1
```

Problema I: Construtores de Totens

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

Várias civilizações pré-colombianas habitaram a região de Alberta, Canadá. Pouco restou destas culturas, dizimadas pelo frio intenso, pelos ursos e, finalmente, pelos invasores ingleses e franceses que chegaram depois do descobrimento. Hoje, pesquisadores da Universidade de Alberta tentam desvendar os mistérios destes povos, estudando os totens produzidos na região.

Estudos do departamento de artes da universidade mostraram que os construtores de totens gostavam de marcá-los com várias cópias de suas assinaturas. A assinatura do artista era feita com sulcos verticais "|" e horizontais "_". Estudiosos imaginam que o símbolo era uma representação do mal na cultura primitiva, e portanto, tabu) construindo um padrão quadrado. Já foram identificadas várias assinaturas de artistas daquela época.

Sua tarefa neste problema é fazer um programa que recebe o desenho de um totem, através de um padrão quadrado de tamanho $n \times n$ de sulcos verticais e horizontais, e uma assinatura quadrada de tamanho $m \times m$ com um padrão do mesmo tipo e encontra todas as ocorrências da assinatura no totem.

Entrada

A entrada é composta de diversas instâncias. A primeira linha de cada instância consiste em dois inteiros n ($1 \leq n \leq 1000$) e m ($1 \leq m \leq 60$ e $m < n$), indicando os tamanhos dos quadrados. Nas n linhas seguintes temos n caracteres ``|'' ou ``_'' que representam os caracteres do quadrado $n \times n$. Nas m linhas seguintes temos m caracteres ``|'' ou ``_'' que representam os caracteres do quadrado $m \times m$.

A entrada termina com final de arquivo.

Saída

Para cada instância, você deverá imprimir um identificador `Instancia k`, onde k é o número da instância atual. Imprima a lista de todas ocorrências `linha coluna` (o canto superior esquerdo é a posição `0 0`) do quadrado $m \times m$ no quadrado $n \times n$. A saída deve estar ordenada primeiro pelas colunas e em seguida pelas linhas. Caso não tenha nenhuma ocorrência imprima uma linha com `nenhum ocorrencia`.

Após cada instância imprima uma linha em branco.

Exemplo de entrada

```
4 2
_|__
___|
__|_
||__
_|
|_
```

Exemplo de saída

```
Instancia 1
2 1
1 2
```

Problema J: A lei vai a Cavalo!

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

A Polícia Montada Real Canadense (Royal Canadian Mounted Police) é uma instituição muito famosa, cujas origens remontam ao século XIX. Sua tarefa é levar a lei aos locais mais longínquos do país continental. Hoje a polícia montada tem um efetivo de 25000 homens e cerca de 5000 cavalos.

Cada sede da RCMP tem um haras em que os animais são muito bem cuidados, e designados aos policiais com quem têm mais afinidade. Esta afinidade é inferida em observações dos oficiais com vários anos de experiência, observando os policiais montando os animais disponíveis. No Fairmont Banff Springs Haras, onde ficam os cavalos montados pelos policiais da região de Banff Springs, é necessário resolver o problema de decidir quais soldados montarão quais cavalos. Note que um cavalo pode ser montado por vários policiais, mas um policial só monta um determinado cavalo. Cada cavalo tem um limite de policiais que podem montá-lo. Ou seja, de posse da afinidade dos vários policiais com os animais que montou nos últimos tempos, deseja-se encontrar uma atribuição dos cavalos aos vários policiais, de tal forma que o maior número possível de policiais tenham um cavalo para montar.

Entrada

A entrada é composta de diversas instâncias. A primeira linha de cada instância consiste em três inteiros n ($1 \leq n \leq 100$), m ($1 \leq m \leq 100$) e k ($1 \leq k \leq 1000$) indicando o número de cavalos, o número de soldados e o número de afinidades. A linha seguinte contêm n inteiros $c_1, c_2, .., c_n$ indicando que no i-ésimo cavalo pode montar c_i ($1 \leq ci \leq 100$) soldados. Nas k linhas seguintes temos dois inteiros u ($1 \leq u \leq n$) e v ($1 \leq v \leq m$) indicando que existe afinidade entre o cavalo u e o soldado v.

A entrada termina com final de arquivo.

Saída

Para cada instância, você deverá imprimir um identificador `Instancia k`, onde k é o número da instância atual. Na linha seguinte imprima o número máximo de policiais que podem ter um cavalo para montar em uma atribuição.

Após cada instância imprima uma linha em branco.

Exemplo de entrada

```
5 3 7
1 1 1 1 1
1 1
1 2
2 1
2 2
2 3
4 3
5 3
```

Exemplo de saída

```
Instancia 1
3
```

Problema L: Fatorial

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

Joãozinho é um garoto esperto da sexta série. Ele gosta muito de matemática, e descobriu que sua professora é muito preguiçosa. Nas provas da matéria a professora pede que as crianças circulem a resposta com um quadrado colorido, e que façam o primeiro dígito diferente de *zero* (da direita para esquerda) do número especialmente grande com caneta. Joãozinho desconfiou que a professora olhava apenas para aquele dígito para corrigir a questão.

A turma aprendeu a calcular o fatorial de um número, e isso será cobrado na próxima prova. Joãozinho está convencido de que não precisa escrever de fato o número correto, desde que o primeiro dígito (olhando da direita para esquerda) seja o correto. Sua tarefa neste problema é ajudar Joãozinho a calcular para um número inteiro n da entrada, o primeiro dígito (da direita para esquerda) de $n!$ que seja diferente de *zero*.

Entrada

A entrada é composta de diversas instâncias. A primeira linha de cada instância consiste um inteiro n ($1 \leq n \leq 1000000$).

A entrada termina com final de arquivo.

Saída

Para cada instância, você deverá imprimir um identificador `Instancia k`, onde k é o número da instância atual. Na linha seguinte imprima o primeiro dígito (da direita para esquerda) diferente de *zero*.

Após cada instância imprima uma linha em branco.

Exemplo de entrada

```
5
```

Exemplo de saída

```
2
```

Problema M: Demonstração de honestidade!

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

Com o grande número de imigrantes argentinos no Canadá, o governo canadense está criando novas rodovias para as regiões mais distantes e isoladas habitadas por argentinos[2]. Foram feitas diversas licitações para descobrir quais empresas poderiam conduzir as obras de cada rodovia. Cada empresa divulgou os orçamentos para as rodovias que ela poderia construir.

Os canadenses são conhecidos pela intolerância à corrupção e querem a qualquer custo evitar que alguma empresa seja beneficiada acima das outras. Então decidiram que cada empresa pode ser contratada para fazer no máximo uma das rodovias. Dá para perceber que no Brasil as coisas funcionam da mesma forma. (Mas não vamos entrar neste mérito!)

Entre duas cidades apenas uma empresa pode ter sido escolhida para construir uma rodovia.

Sua tarefa é: Dado um conjunto de orçamentos para construção das rodovias que ligam as cidades decida se existe uma maneira de atribuir as construções para as empresas, atendendo a exigência do governo canadense, e que seja possível viajar de qualquer cidade para qualquer outra usando as rodovias construídas[3].

Entrada

A entrada é composta de diversas instâncias. A primeira linha de cada instância consiste em três inteiros n ($1 \leq n \leq 100$), m ($1 \leq 10000$) e k ($1 \leq k \leq 2n$) que indicam o número de cidades, número de orçamentos e o número de empresas. As próximas m linhas contém três inteiros u ($1 \leq u \leq n$), v ($1 \leq v \leq n$) e c ($1 \leq c \leq k$) indicando que a empresa c pode construir uma rodovia que liga a cidade u à cidade v.

As instâncias são separadas por uma linha em branco.

A entrada termina com final de arquivo.

Saída

Para cada instância, você deverá imprimir um identificador `Instancia k`, onde k é o número da instância atual. Na linha seguinte imprima `sim` se existe uma atribuição de construções de rodovias que atenda as exigências descritas acima, caso contrário imprima `nao`.

Após cada instância imprima uma linha em branco.

[2]Os argentinos não gostam de se misturar.

[3]O governo canadense quer a qualquer custo garantir que os argentinos (folgados) se comuniquem!

Exemplo de entrada

```
3 3 3
1 2 1
2 3 2
3 1 3

6 9 5
1 2 3
2 3 4
3 1 5
1 4 1
2 5 1
3 6 2
4 5 2
5 6 1
6 4 1
```

Exemplo de saída

```
Instancia 1
sim

Instancia 2
nao
```