OBI

OLIMPÍADA BRASILEIRA DE INFORMÁTICA

OBI2000
CADERNO DE TAREFAS

10/6/2000 • 13:00 às 18:00

LEIA ATENTAMENTE ESTAS INSTRUÇÕES ANTES DE INICIAR A PROVA

1. É proibido consultar livros, anotações ou qualquer outro material durante a prova. É permitido a consulta ao *help* do ambiente de programação se este estiver disponível.
2. Todas as tarefas têm o mesmo valor na correção.
3. As tarefas não estão ordenadas, neste caderno, por ordem de dificuldade; procure resolver primeiro as questões mais fáceis.
4. Preste muita atenção no nome dos arquivos fonte indicados nas tarefas. Soluções na linguagem C devem ser arquivos com sufixo .c; soluções na linguagem C++ devem ser arquivos com sufixo .cc ou .cpp; soluções na linguagem Pascal devem ser arquivos com sufixo .pas.
5. Para problemas diferentes você pode escolher trabalhar com linguagens diferentes, mas apenas uma solução, em uma única linguagem, deve ser submetida para cada problema.
6. Ao final da prova, para cada solução que você queira submeter para correção, copie o arquivo fonte para o seu diretório de trabalho ou disquete, conforme especificado pelo seu professor.
7. Não utilize arquivos para entrada ou saída. Todos os dados devem ser lidos da entrada padrão (normalmente é o teclado) e escritos na saída padrão (normalmente é a tela). Utilize as funções padrão para entrada e saída de dados:
   - em Pascal: *readln, read, writeln, write*;
   - em C: *scanf, getchar, printf, putchar*;
   - em C++: as mesmas de C ou os objetos *cout* e *cin*.
8. Procure resolver o problema de maneira eficiente. Na correção, eficiência também será levada em conta.
9. Não implemente nenhum recurso gráfico nas suas soluções (janelas, menus, etc).

Sociedade Brasileira de Computação
http://www.sbc.org.br Email: sbc@sbc.org.br

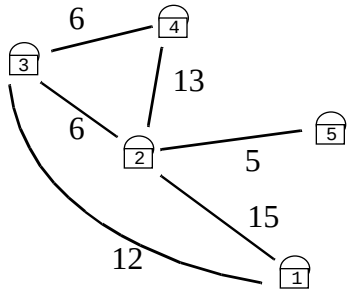
Rede ótica

Arquivo fonte: rede.c, rede.cc, rede.cpp ou rede.pas

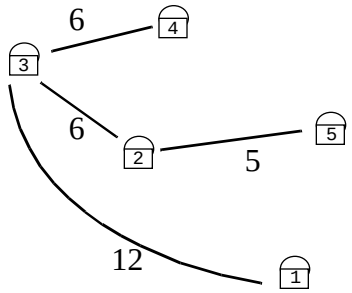
Os caciques da região de Tutuaçu pretendem integrar suas tribos à chamada “aldeia global”. A primeira providência foi a distribuição de telefones celulares a todos os pajés. Agora, planejam montar uma rede de fibra ótica interligando todas as tabas. Esta empreitada requer que sejam abertas novas picadas na mata, passando por reservas de flora e fauna. Conscientes da necessidade de preservar o máximo possível o meio ambiente, os caciques encomendaram um estudo do impacto ambiental do projeto. Será que você consegue ajudá-los a projetar a rede de fibra ótica?

1. Tarefa

Vamos denominar uma ligação de fibra ótica entre duas tabas de um *ramo* de rede. Para possibilitar a comunicação entre todas as tabas é necessário que todas elas estejam interligadas, direta (utilizando um ramo de rede) ou indiretamente (utilizando mais de um ramo). Os caciques conseguiram a informação do impacto ambiental que causará a construção dos ramos. Alguns ramos, no entanto, nem foram considerados no estudo ambiental, pois sua construção é impossível.



Ramos de rede possíveis com custo ambiental associado



Interligação das tabas com menor custo ambiental

Sua tarefa é escrever um programa para determinar quais ramos devem ser construídos, de forma a possibilitar a comunicação entre todas as tabas, causando o menor impacto ambiental possível.

2. Entrada

A entrada é composta de vários conjuntos de teste. A primeira linha de um conjunto de teste contém dois números inteiros positivos *N* e *M* que indicam, respectivamente, o número de tabas e o número de ramos de redes possíveis. As tabas são numeradas de 1 a *N*. As *M* linhas seguintes contêm três inteiros positivos *X*, *Y* e Z, que indicam que o ramo de rede que liga a taba *X* à taba Y tem impacto ambiental Z. Com os conjuntos de teste dados sempre é possível interligar todas as tabas. O final da entrada é indicado quando *N* = 0.

Exemplo de Entrada

```
3 3
1 2 10
2 3 10
3 1 10
5 6
1 2 15
1 3 12
2 4 13
2 5 5
3 2 6
3 4 6
0 0
```

3. Saída

Para cada conjunto de teste da entrada seu programa deve produzir uma lista dos ramos de redes que devem ser construídos. A lista deve ser precedida de uma linha que identifica o conjunto de teste, no formato "Teste *n*", onde *n* é numerado a partir de 1. A lista é composta por uma sequência de ramos a serem construídos, um ramo por linha. Um ramo é descrito por um par de tabas *X* e *Y* , com $X < Y$. Os ramos de rede podem ser listados em qualquer ordem, mas não deve haver repetição. Se houver mais de uma solução possível, imprima apenas uma delas. O final de uma lista de ramos deve ser marcado com uma linha em branco. A grafia mostrada no Exemplo de Saída, abaixo, deve ser seguida rigorosamente.

Exemplo de Saída

```
Teste 1
1 2
1 3

Teste 2
1 3
2 3
2 5
3 4
```

(esta saída corresponde ao exemplo de entrada acima)

4. Restrições

$0 \leq N \leq 100$ ($N = 0$ apenas para indicar o fim da entrada)
$1 \leq M \leq N(N\text{-}1)/2$
$1 \leq X \leq 100$
$1 \leq Y \leq 100$
$1 \leq Z \leq 100$

Quermesse

Arquivo fonte: querm.c, querm.cc, querm.cpp ou querm.pas

Os alunos do último ano resolveram organizar uma quermesse para arrecadar fundos para a festa de formatura. A festa prometia ser um sucesso, pois o pai de um dos formandos, Teófilo, dono de uma loja de informática, decidiu doar um computador para ser sorteado entre os que comparecessem. Os alunos prepararam barracas de quentão, pipoca, doces, ensaiaram a quadrilha e colocaram à venda ingressos numerados sequencialmente a partir de 1. O número do ingresso serviria para o sorteio do computador. Ficou acertado que Teófilo decidiria o método de sorteio; em princípio o sorteio seria, claro, computadorizado.

O local escolhido para a festa foi o ginásio da escola. A entrada dos participantes foi pela porta principal, que possui uma roleta, onde passa uma pessoa por vez. Na entrada, um funcionário inseriu, em uma lista no computador da escola, o número do ingresso, na ordem de chegada dos participantes. Depois da entrada de todos os participantes, Teófilo começou a trabalhar no computador para preparar o sorteio. Verificando a lista de presentes, notou uma característica notável: havia apenas um caso, em toda a lista, em que o participante que possuia o ingresso numerado com *i*, havia sido a *i-ésima* pessoa a entrar no ginásio. Teófilo ficou tão encantado com a coincidência que decidiu que o sorteio não seria necessário: esta pessoa seria o ganhador do computador.

1. Tarefa

Conhecendo a lista de participantes, por ordem de chegada, sua tarefa é determinar o número do ingresso premiado, sabendo que o ganhador é o único participante que tem o número do ingresso igual à sua posição de entrada na festa.

2. Entrada

A entrada é composta de vários conjuntos de teste. A primeira linha de um conjunto de teste contém um número inteiro positivo N que indica o número de participantes da festa. A linha seguinte contém a sequência, em ordem de entrada, dos N ingressos das pessoas que participaram da festa. O final da entrada é indicado quando $N = 0$. Para cada conjunto de teste da entrada haverá um único ganhador.

Exemplo de Entrada

```
4
4 5 3 1
10
9 8 7 6 1 4 3 2 12 10
0
```

3. Saída

Para cada conjunto de teste da entrada seu programa deve produzir três linhas. A primeira linha identifica o conjunto de teste, no formato "Teste *n*", onde *n* é numerado a partir de 1. A segunda

linha deve conter o número do ingresso do ganhador, conforme determinado pelo seu programa. A terceira linha deve ser deixada em branco. A grafia mostrada no Exemplo de Saída, abaixo, deve ser seguida rigorosamente.

Exemplo de Saída

```
Teste 1
3

Teste 2
10
```

(esta saída corresponde ao exemplo de entrada acima)

4. Restrições

$0 \leq N \leq 10000$ ($N = 0$ apenas para indicar o fim da entrada)

Bits Trocados

Arquivo fonte: bit.c, bit.cc, bit.cpp ou bit.pas

As Ilhas Weblands formam um reino independente nos mares do Pacífico. Como é um reino recente, a sociedade é muito influenciada pela informática. A moeda oficial é o Bit; existem notas de B$ 50,00, B$10,00, B$5,00 e B$1,00. Você foi contratado(a) para ajudar na programação dos caixas automáticos de um grande banco das Ilhas Weblands.

1. Tarefa

Os caixas eletrônicos das Ilhas Weblands operam com todos os tipos de notas disponíveis, mantendo um estoque de cédulas para cada valor (B$ 50,00, B$10,00, B$5,00 e B$1,00). Os clientes do banco utilizam os caixas eletrônicos para efetuar retiradas de um certo número inteiro de Bits.

Sua tarefa é escrever um programa que, dado o valor de Bits desejado pelo cliente, determine o número de cada uma das notas necessário para totalizar esse valor, de modo a minimizar a quantidade de cédulas entregues. Por exemplo, se o cliente deseja retirar B$50,00, basta entregar uma única nota de cinquenta Bits. Se o cliente deseja retirar B$72,00, é necessário entregar uma nota de B$50,00, duas de B$10,00 e duas de B$1,00.

2. Entrada

A entrada é composta de vários conjuntos de teste. Cada conjunto de teste é composto por uma única linha, que contém um número inteiro positivo *V*, que indica o valor solicitado pelo cliente. O final da entrada é indicado por $V = 0$.

Exemplo de Entrada

```
1
72
0
```

3. Saída

Para cada conjunto de teste da entrada seu programa deve produzir três linhas na saída. A primeira linha deve conter um identificador do conjunto de teste, no formato “Teste *n*”, onde *n* é numerado a partir de 1. Na segunda linha devem aparecer quatro inteiros *I*, *J*, *K* e *L* que representam o resultado encontrado pelo seu programa: *I* indica o número de cédulas de B$50,00, *J* indica o número de cédulas de B$10,00, *K* indica o número de cédulas de B$5,00 e *L* indica o número de cédulas de B$1,00. A terceira linha deve ser deixada em branco. A grafia mostrada no Exemplo de Saída, abaixo, deve ser seguida rigorosamente.

Exemplo de Saída

```
Teste 1
0 0 0 1

Teste 2
1 2 0 2
```

(esta saída corresponde ao exemplo de entrada acima)

4. Restrições

$0 \leq V \leq 10000$ (V= 0 apenas para indicar o fim da entrada)

Saldo de gols

Arquivo fonte: saldo.c, saldo.cc, saldo.cpp ou saldo.pas

Hipólito é um torcedor fanático. Coleciona flâmulas, bandeiras, recortes de jornal, figurinhas de jogadores, camisetas e tudo o mais que se refira a seu time preferido. Quando ganhou um computador de presente em uma festa, resolveu montar um banco de dados com os resultados de todos os jogos de seu time ocorridos desde a sua fundação, em 1911. Depois de inseridos os dados, Hipólito começou a ficar curioso sobre estatísticas de desempenho do time. Por exemplo, ele deseja saber qual foi o período em que o seu time acumulou o maior saldo de gols. Como Hipólito tem o computador há muito pouco tempo, não sabe programar muito bem, e precisa de sua ajuda.

1. Tarefa

É dada uma lista, numerada seqüencialmente a partir de 1, com os resultados de todos os jogos do time (primeira partida: 3 x 0, segunda partida: 1 x 2, terceira partida: 0 x 5 ...). Sua tarefa é escrever um programa que determine em qual período o time conseguiu acumular o maior saldo de gols. Um *período* é definido pelos números de seqüência de duas partidas, *A* e *B*, onde $A \leq B$. O saldo de gols acumulado entre *A* e *B* é dado pela soma dos gols marcados pelo time em todas as partidas realizadas entre A e B (incluindo as mesmas) menos a soma dos gols marcados pelos times adversários no período. Se houver mais de um período com o mesmo saldo de gols, escolha o maior período (ou seja, o período em que B - A é maior). Se ainda assim houver mais de uma solução possível, escolha qualquer uma delas como resposta.

2. Entrada

Seu programa deve ler vários conjuntos de teste. A primeira linha de um conjunto de teste contém um inteiro não negativo, *N*, que indica o número de partidas realizadas pelo time (o valor $N = 0$ indica o final da entrada). Seguem-se *N* linhas, cada uma contendo um par de números inteiros não negativos *X* e *Y* que representam o resultado da partida: *X* são os gols a favor e *Y* os gols contra o time de Hipólito. As partidas são numeradas sequencialmente a partir de 1, na ordem em que aparecem na entrada.

Exemplo de Entrada

```
2
2 3
7 1
9
2 2
0 5
6 2
1 4
0 0
5 1
1 5
6 2
```

```
0 5
3
0 2
0 3
0 4
0
```

3. Saída

Para cada conjunto de teste da entrada seu programa deve produzir três linhas na saída. A primeira linha deve conter um identificador do conjunto de teste, no formato "Teste *n*", onde *n* é numerado a partir de 1. A segunda linha deve conter um par de inteiros *I* e *J* que indicam respectivamente a primeira e última partidas do melhor período, conforme determinado pelo seu programa, exceto quando o saldo de gols do melhor período for menor ou igual a zero; neste caso a segunda linha deve conter a expressão "nenhum". A terceira linha deve ser deixada em branco. A grafia mostrada no Exemplo de Saída, abaixo, deve ser seguida rigorosamente.

Exemplo de Saída

```
Teste 1
2 2

Teste 2
3 8

Teste 3
nenhum
```

(esta saída corresponde ao exemplo de entrada acima)

4. Restrições

$0 \leq N \leq 10000$ ($N = 0$ apenas para indicar o fim da entrada)
$1 \leq A \leq N$
$A \leq B \leq N$
$0 \leq X \leq 50$
$0 \leq Y \leq 50$

Macaco-prego

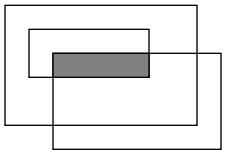
Arquivo fonte: macaco.c, macaco.cc, macaco.cpp ou macaco.pas

O macaco-prego é um animal irrequieto e barulhento, merecedor também dos adjetivos desordeiro e despudorado. A sua cabeça, encimada por uma densa pelagem negra ou marrom-escura, semelhante a um gorro, torna seu aspecto inconfundível. Apesar de ser o macaco mais comum nas matas do país, uma de suas sub-espécies encontra-se seriamente ameaçada de extinção: o macaco-prego-do-peito-amarelo, que se distingue das demais pela coloração amarelada do peito e da parte anterior dos braços.

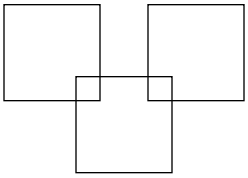
Um grande esforço foi feito pelos primatologistas para aumentar a população dos macacos-prego-do-peito-amarelo. Sabe-se que eles se alimentam de plantas, das quais consomem preferencialmente frutos e brotos. Alimentam-se também de muitos animais, preferencialmente lesmas, lagartas e rãs, e preferem as florestas mais densas. Para determinar o melhor local do país para criar uma nova reserva ambiental para os macacos-prego-do-peito-amarelo, o governo fez um levantamento das regiões no país onde as condições preferidas desses animais ocorrem: regiões de floresta densa, regiões com frutos, regiões com muitos brotos, etc. Ajude a salvar os macacos-prego-do-peito-amarelo.

1. Tarefa

As regiões propícias para o macaco-prego-do-peito-amarelo foram determinadas como retângulos cujos lados são todos verticais ou horizontais. Sua tarefa é encontrar o local ideal para a reserva ambiental, definida como a interseção de todas as regiões dadas.



Conjunto de três regiões com interseção preenchida em cinza



Conjunto de três regiões com interseção vazia

As regiões foram divididas de tal forma que uma região não tangencia qualquer outra região. Assim, a interseção entre quaisquer duas regiões ou é um retângulo ou é vazia.

2. Entrada

Seu programa deve ler vários conjuntos de teste. A primeira linha de um conjunto de teste contém um inteiro não negativo, *N*, que indica o número de regiões (o valor $N = 0$ indica o final da entrada). Seguem-se *N* linhas, cada uma contendo quatro números inteiros *X*, *Y*, *U* e *V* que descrevem uma região: o par *X*, *Y* representa a coordenada do canto superior esquerdo e o par *U*, *V* representa a coordenada do canto inferior direito de um retângulo.

Exemplo de Entrada

```
3
0 6 8 1
1 5 6 3
2 4 9 0
3
0 4 4 0
3 1 7 -3
6 4 10 0
0
```

3. Saída

Para cada conjunto de teste da entrada seu programa deve produzir três linhas na saída. A primeira linha deve conter um identificador do conjunto de teste, no formato "Teste *n*", onde *n* é numerado a partir de 1. A segunda linha deve conter as coordenadas do retângulo de interseção encontrado pelo seu programa, no mesmo formato utilizado na entrada. Caso a interseção seja vazia, a segunda linha deve conter a expressão "nenhum". A terceira linha deve ser deixada em branco. A grafia mostrada no Exemplo de Saída, abaixo, deve ser seguida rigorosamente.

Exemplo de Saída

```
Teste 1
2 4 6 3

Teste 2
nenhum
```

(esta saída corresponde ao exemplo de entrada acima)

4. Restrições

$0 \leq N \leq 10000$ ($N = 0$ apenas para indicar o fim da entrada)
$-10000 \leq X \leq 10000$
$-10000 \leq Y \leq 10000$
$-10000 \leq U \leq 10000$
$-10000 \leq V \leq 10000$