OBI

OLIMPÍADA BRASILEIRA
DE INFORMÁTICA

OBI2003

CADERNO DE TAREFAS - SELETIVA IOI

5/7/2003 • 8:30 às 13:30

Leia atentamente estas instruções antes de iniciar a prova. Este caderno é composto de 10 páginas, numeradas de 1 a 10. Verifique se o caderno está completo.

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. **A correção é automatizada**, portanto siga atentamente as exigências da tarefa quanto ao formato da entrada e saída de seu programa.
3. **Não implemente nenhum recurso gráfico nas suas soluções** (janelas, menus, etc), nem utilize qualquer rotina para limpar a tela ou posicionar o cursor.
4. Todas as tarefas têm o mesmo valor na correção.
5. As tarefas **não** estão ordenadas, neste caderno, por ordem de dificuldade; procure resolver primeiro as questões mais fáceis.
6. 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.
7. Para problemas diferentes você pode escolher trabalhar com linguagens diferentes, mas apenas uma solução, em uma única linguagem, deve ser submetida para cada problema.
8. 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.
9. **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*.
10. Procure resolver o problema de maneira eficiente. Na correção, eficiência também será levada em conta. As soluções serão testadas com outras entradas além das apresentadas como exemplo nas tarefas.

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

Festa Junina

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

Conforme a tradição da sua escola, os alunos do último ano do ensino médio organizarão uma festa junina no colégio. Porém, o diretor da escola tem tido problemas nos últimos anos com a organização desta festa, e ele percebeu que a causa destes problemas é a presença de alunos que não se toleram na comissão organizadora. Assim, neste ano, o diretor resolveu que ele mesmo designaria a comissão organizadora da festa junina, de forma que não haja inimizades entre os membros da comissão. Para isto, o diretor distribuiu um formulário a todos alunos da turma; cada aluno deve listar os alunos com os quais ele não gostaria de participar da comissão organizadora. A partir destas informações, o diretor deseja montar uma comissão organizadora para a festa com o maior número possível de alunos, de forma a não sobrecarregar os seus integrantes.

1. Tarefa

Dadas as informações retiradas dos formulários de todos os alunos, sua tarefa é determinar qual o número máximo de alunos que a comissão organizadora pode ter.

2. Entrada

A entrada é composta de vários conjuntos de teste. A primeira linha de um conjunto de teste contém um número inteiro N, que indica o número de alunos na turma ($N \leq 20$). Os alunos são identificados seqüencialmente pelos números de 1 a N. A seguir, para cada um dos N alunos, seguindo a ordem dos números de identificação, há uma linha contendo a lista dos alunos com os quais este aluno não gostaria de participar na comissão organizadora. O final de uma lista é indicado pelo número zero, e o final da entrada é indicado por um conjunto de teste com $N = 0$.

Exemplo de Entrada

```
5
2 4 0
1 0
0
5 3 0
4 0
6
6 0
6 0
6 0
6 0
6 0
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 seqüencialmente a partir de 1. A segunda linha deve conter o número máximo de alunos que podem participar em uma mesma comissão organizadora, conforme calculado 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
5
```

(esta saída corresponde ao exemplo de entrada acima)

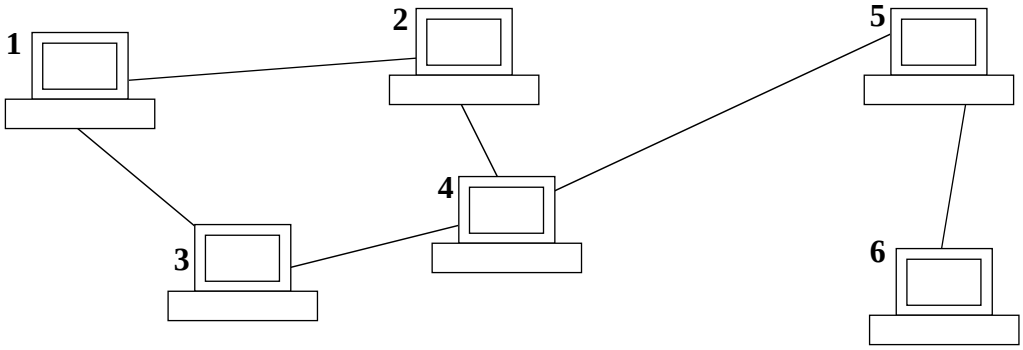
4. Restrições

$0 \leq N \leq 20$ ($N = 0$ apenas para indicar o fim da entrada)

Manutenção

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

Uma empresa possui vários computadores conectados em rede. Isto possibilita que os funcionários compartilhem recursos e consigam colaborar melhor para o desempenho de suas tarefas dentro da empresa. No entanto, as máquinas não estão diretamente conectadas todas entre si. Por economia de recursos, a topologia de rede adotada apresenta apenas um subconjunto das conexões possíveis, conforme o exemplo apresentado na figura abaixo. Note que as conexões são sempre bidirecionais.



Apesar de alguns computadores não estarem diretamente conectados entre si, eles ainda conseguem se comunicar porque existe um algoritmo de roteamento capaz de conectar dois computadores através de várias conexões diretas. No exemplo da figura, o computador 1 conseguiria se comunicar com o computador 5 através dos computadores 2 e 4 ou através dos computadores 3 e 4.

No entanto, freqüentemente as máquinas precisam passar por uma revisão de rotina. Quando uma máquina está em manutenção, ela precisa ser temporariamente desconectada da rede e levada para a oficina. Assim, o algoritmo de roteamento não tem mais como estabelecer conexões utilizando este computador, o que pode acabar desconectando duas ou mais partes da rede e prejudicando o trabalho na empresa. No exemplo dado, se o computador 2 fosse para a revisão não teríamos problema pois todas as outras máquinas ainda conseguiriam se comunicar entre si. No entanto, se o computador 4 fosse para a manutenção, as máquinas 1, 2 e 3 não conseguiriam se comunicar com as máquinas 5 e 6.

Se a remoção de uma máquina desconectar o restante da rede, impedindo que outros computadores se comuniquem, é necessário deixar uma máquina substituta em seu lugar durante o período de manutenção, o que representa um custo extra no orçamento da empresa.

1. Tarefa

Sua tarefa é escrever um programa que identifique quais são os computadores que precisam ser substituídos durante a sua manutenção, para que os demais continuem se comunicando através da rede.

2. Entrada

A entrada é constituída de vários conjuntos de teste. A primeira linha de um conjunto de teste contém dois números inteiros N e M, que indicam respectivamente o número de computadores na rede e o número de conexões diretas entre eles ($1 \leq N \leq 400$, e $N\text{-}1 \leq M \leq N(N\text{-}1)/2$). Os computa-

dores são identificados por números de 1 a *N*. As *M* linhas seguintes contêm, cada uma, um par de números inteiros *X* e *Y*. Cada linha representa uma conexão existente na rede, indicando que os computadores *X* e *Y* possuem uma conexão direta entre si. O final da entrada é indicado por um conjunto de teste com $N = M = 0$. Você pode assumir que todos os conjuntos de teste representam redes conexas, onde todos os computadores conseguem se comunicar entre si através do algoritmo de roteamento.

Exemplo de Entrada

```
6 6
1 2
3 1
2 4
3 4
4 5
5 6
4 6
1 2
1 3
1 4
2 3
2 4
3 4
4 3
1 2
1 3
1 4
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 a lista com os computadores que precisam ser substituídos durante sua manutenção. Esta lista deve estar ordenada de forma crescente, e cada valor deve ser seguido de um espaço em branco. Caso nenhum dos computadores da rede precise ser substituído, escreva "nenhum" na saída. A terceira linha deve ser deixada em branco. O formato mostrado no exemplo de saída abaixo deve ser seguido rigorosamente.

Exemplo de Saída

```
Teste 1
4 5

Teste 2
nenhum

Teste 3
1
```

(esta saída corresponde ao exemplo de entrada acima)

4. Restrições

$0 \leq N \leq 400$ ($N = 0$ apenas para indicar o fim da entrada)
$N - 1 \leq M \leq N(N-1)/2$
$1 \leq X \leq N$
$1 \leq Y \leq N$
$X \neq Y$

Parque Jurássico

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

O DNA é uma molécula envolvida na transmissão de caracteres hereditários e na produção de proteínas, que são os principais constituintes de seres vivos. O DNA é formado pelas bases nitrogenadas adenina (A), guanina (G), citosina (C) e timina (T).

A identificação da seqüência de bases que constitui uma determinada parte do DNA pode ajudar a descoberta da cura de doenças que atacam seres vivos. Teoricamente, a identificação do DNA pode também permitir a recriação de espécies extintas, como na estória do escritor americano Michael Crichton.

O professor de Biologia de sua escola, prof. Estevão Espilbergo, conseguiu amostras de células de uma espécie de mosquito extinto a milhares de anos, e pretende, ambiciosamente, recriar o animal a partir de seu DNA. Para isso, conseguiu que um laboratório de genômica fizesse a identificação das bases das células. No entanto, pelo estado precário das células obtidas, o resultado não foi dos melhores. O professor Estevão recebeu do laboratório duas seqüências, com a informação de que essas seqüências contêm, provavelmente, muitos "buracos", ou seja, entre uma base e outra corretamente detectadas podem existir bases não detectadas.

O prof. Estevão então decidiu combinar as duas seqüências para formar uma seqüência única, e precisa de sua ajuda.

1. Tarefa

Sua tarefa é escrever um programa que determine a menor seqüência que contenha, como subseqüências, as duas seqüências obtidas pelo laboratório. Dizemos que uma seqüência S_1 é subseqüência de uma outra seqüência S_2 se acrescentando-se alguns elementos a S_1 obtém-se S_2. Por exemplo, ACGT é uma subseqüência de ATCGAAT, pois basta inserir um T após o A e dois A's após o G.

2. Entrada

A entrada possui vários conjuntos de teste. Cada conjunto de teste é composto por duas linhas, cada uma contendo uma seqüência *S* composta por caracteres 'A', 'C', 'G' e 'T'. O final da entrada é indicado por uma linha contendo o caractere '#'.

Exemplo de Entrada

```
AAATTT
GAATCT
ACGT
ATCGAAT
#
```

3. Saída

Para cada conjunto de teste, o seu programa deve escrever três linhas na saída. A primeira linha deve conter um identificador do conjunto de teste, no formato "Teste *n*", onde *n* é numerado seqüencialmente a partir de 1. A segunda linha deve conter uma seqüência de comprimento

mínimo que contenha as duas seqüências da entrada como subseqüências. Se houver mais de uma seqüência de comprimento mínimo, seu programa pode escrever qualquer uma delas. 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
GAAATCTT

Teste 2
ATCGAAT
```

(esta saída corresponde ao exemplo de entrada acima)

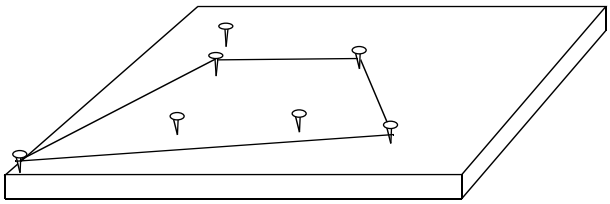
4. Restrições

1 ≤ número de caracteres de *S* ≤ 100

Elástico

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

Este é o nome sugestivo de um jogo muito comum entre um grupo de crianças (todos filhos de professores de geometria). Para este jogo, as crianças utilizam uma prancha retangular de madeira, na qual uma tachinha é fixa no canto inferior esquerdo. Uma borrachinha elástica é amarrada nessa tachinha. As crianças então pregam várias outras tachinhas espalhadas pelo espaço restante. O objetivo do jogo é formar, com a borrachinha, um polígono convexo com pelo menos 3 vértices (tachinhas). Ganha o jogo quem formar o polígono com o maior número de vértices.



Um exemplo de configuração vencedora

1. Tarefa

Você deve implementar um programa que, dada uma instância do jogo, determine o número de vértices da melhor solução possível.

2. Entrada

A entrada é composta de vários conjuntos de teste. A primeira linha de cada conjunto de teste contém um número inteiro N ($2 \leq N \leq 200$), que indica o número de tachinhas pregadas no pedaço de madeira. A seguir, são dadas N linhas, cada uma contendo dois números inteiros X e Y que representam a posição de uma tachinha em coordenadas cartesianas com relação à tachinha presa no canto inferior esquerdo, considerada a origem do sistema de coordenadas. O final da entrada é dado por um conjunto de teste com $N = 0$. Você pode assumir que não existem duas tachinhas com as mesmas coordenadas e que não existem três tachinhas alinhadas (na mesma reta) dentro de uma mesma instância do problema.

Exemplo de Entrada

```
6
4 3
2 2
2 4
3 2
3 1
1 5
8
10 8
3 9
2 8
2 3
9 2
```

```
9 10
10 3
8 10
0
```

3. Saída

Para cada conjunto de teste seu programa deve escrever três linhas na saída. A primeira linha deve conter um identificador do conjunto de teste, no formato "Teste *n*", onde *n* é numerado seqüencialmente a partir de 1. A segunda linha deve conter o número de vértices da melhor solução para a instância dada. A terceira linha deve ser deixada em branco. O formato do exemplo de saída abaixo deve ser seguido rigorosamente.

Exemplo de Saída

```
Teste 1
5

Teste 2
8
```

(esta saída corresponde ao exemplo de entrada acima)

4. Restrições

$2 \leq N \leq 200$ ($N = 0$ apenas para indicar o final da entrada)
$1 \leq X \leq 1000$
$1 \leq Y \leq 1000$