OBI

OLIMPÍADA BRASILEIRA
DE INFORMÁTICA

CADERNO DE TAREFAS

PRIMEIRA FASE • 27/6/99 • 13:00 às 17:00

Instruções:

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 de entrada e de saída indicados nas tarefas.
5. Para submeter uma solução, copie o arquivo executável e o arquivo fonte para o seu diretório de trabalho ou disquete, conforme especificado pelo seu professor. Não serão consideradas submissões em que um destes dois arquivos esteja faltando.

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

Projeto Genoma

Um grande projeto mundial está em curso para mapear todo o material genético do ser humano: o Projeto Genoma Humano. As moléculas de DNA (moléculas que contêm material genético) podem ser representadas por cadeias de caracteres que usam um alfabeto de apenas 4 letras: ‘A’, ‘C’, ‘T’ e ‘G’. Um exemplo de uma tal cadeia é:

TCATATGCAAATAGCTGCATACCGA

Nesta tarefa você deverá produzir uma ferramenta muito utilizada no projeto Genoma: um programa que procura ocorrências de uma pequena cadeia de DNA (que vamos chamar de *p*) dentro de uma outra cadeia de DNA (que vamos chamar de *t*). Você deverá procurar dois tipos de ocorrência: a "direta" e a "complementar invertida".

Uma ocorrência *direta* é quando a cadeia *p* aparece como subcadeia dentro de *t*. Por exemplo, se
p = CATA
t = TCATATGCAAATAGCTGCATACCGA,
então *p* ocorre na forma direta na posição 2 e na posição 18 de *t*.

Uma ocorrência *complementar invertida* depende da seguinte correspondência entre as letras do DNA: ‘A’ ⇔ ‘T’ e ‘G’ ⇔ ‘C’. "Complementar o DNA" significa trocar as letras de uma cadeia de DNA seguindo essa correspondência. Se complementarmos a cadeia CATA, vamos obter GTAT. Mas além de complementar, é preciso também inverter, ou seja, de GTAT obter TATG. E é esta cadeia que deverá ser procurada, no caso da ocorrência complementar invertida. Assim, se *p* e *t* são as mesmas cadeias do exemplo anterior, então *p* ocorre na forma complementar invertida na posição 4 de *t*.

1. Tarefa

Sua tarefa é escrever um programa que, dadas duas cadeias *p* e *t*, onde o comprimento de *p* é menor ou igual ao comprimento de *t*, procura todas as ocorrências diretas e todas as ocorrências complementares invertidas de *p* em *t*.

2. Entrada de Dados

O arquivo GENOMA.IN contém vários conjuntos de teste. Cada conjunto de teste é composto por três linhas. A primeira linha contém dois inteiros positivos, M e N, M ≤ N, que indicam respectivamente o comprimento das cadeias de DNA *p* e *t*, conforme descrito acima. A segunda linha do conjunto de teste contém a cadeia *p*, e a terceira linha contém a cadeia *t*, onde *p* e *t* são compostas utilizando apenas os caracteres ‘A’, ‘C’, ‘G’ e ‘T’. O final do arquivo de testes é indicado quando M = N = 0 (este último conjunto de testes não é válido e não deve ser processado).
O arquivo GENOMA.IN contém ao menos um conjunto de teste que deve ser processado.

Exemplo de Entrada

```
2 4
AC
TGGT
4 25
CATA
TCATATGCAAATAGCTGCATACCGA
0 0
```

3. Saída de Dados

Seu programa deve produzir um arquivo de saída chamado GENOMA.OUT. Para cada conjunto de teste do arquivo de entrada seu programa deve produzir quatro linhas no arquivo de 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 deve aparecer a lista, em ordem crescente, com a posição inicial de cada ocorrência, na forma direta, do padrão *p* na seqüência *t*. Na terceira linha deve aparecer a lista, em ordem crescente, com a posição inicial de cada ocorrência, na forma complementar invertida, do padrão *p* na seqüência *t*. A quarta linha deve ser deixada em branco. A grafia mostrada no Exemplo de Saída, abaixo, deve ser seguida rigorosamente.

Exemplo de Saída

```
Teste 1
ocorrencia direta: 0
ocorrencia complementar invertida: 3

Teste 2
ocorrencia direta: 2 18
ocorrencia complementar invertida: 4
```

(esta saída corresponde ao exemplo de entrada acima)

4. Restrições

$1 \leq M \leq 15000$
$1 \leq N \leq 15000$
$M \leq N$
$M = 0$ e $N = 0$ apenas para indicar o fim do arquivo de entrada

Imagens de Satélite

A imagem de uma zona rural, gerada por satélite, deve ser analisada para determinar quantas construções existem na área da imagem. A imagem é capturada com uma câmara sensível a radiação infra-vermelha, que diferencia áreas construídas e áreas não construídas. Ao ser digitalizada, a imagem é dividida em um quadriculado de células com M linhas e N colunas. Na imagem digitalizada, células que não contêm qualquer construção recebem o código numérico 0. Células que contêm algum material de construção são representadas por um código numérico de 1 a 9, que indicam a densidade do material de construção.

Você deve supor que as construções não se sobrepõem na imagem, e construções distintas são separadas por uma distância de pelo menos o tamanho de uma célula. Desta maneira, uma célula escura pertence a uma única construção, e células escuras adjacentes pertencem à mesma construção. Células adjacentes são vizinhas imediatas nas direções horizontal, vertical ou diagonal.

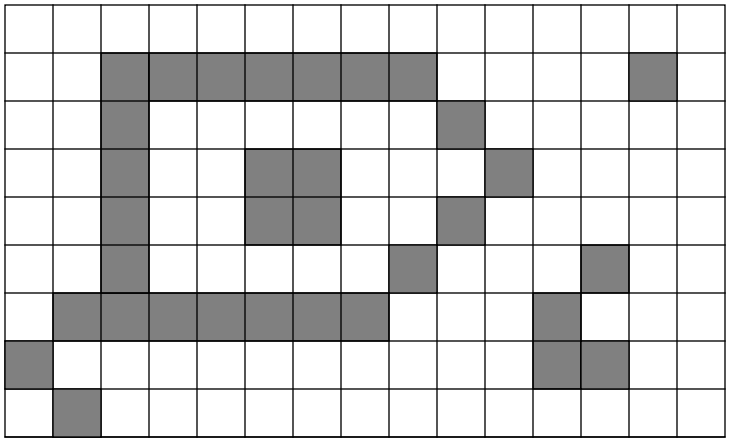


Imagem com 4 construções.

Note que, obedecidas as restrições acima, uma construção pode “circundar” outras construções, como mostrado na figura acima. Neste caso, as contruções devem ser consideradas distintas.

1. Tarefa

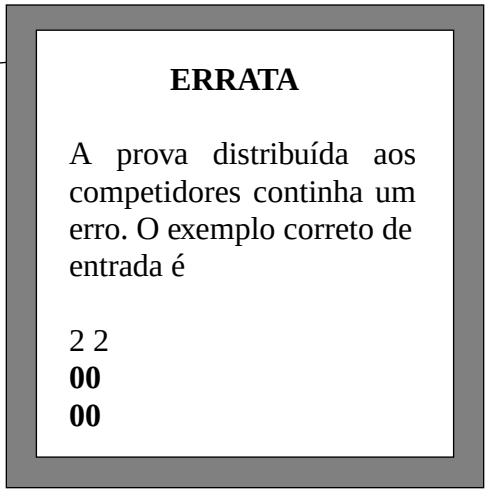
Sua tarefa é escrever um programa que, dada uma imagem de satélite digitalizada, determine quantas construções distintas aparecem na imagem.

2. Entrada de Dados

O arquivo IMAGEM.IN contém vários conjuntos de teste. A primeira linha de um conjunto de testes contém dois inteiros positivos, M e N, que indicam respectivamente o número de linhas e o número de colunas da imagem a ser analisada. As M linhas seguintes contêm N dígitos cada (dígitos entre 0, 1, 2..., 9), correspondendo à imagem enviada pelo satélite. O final do arquivo de testes é indicado quando M = N = 0 (este último conjunto de testes não é válido e não deve ser processado).
O arquivo IMAGEM.IN contém ao menos um conjunto de teste que deve ser processado.

Exemplo de Entrada

```
2 2
0 0
0 0
9 15
000000000000000
007677888000080
008000000900000
006003300080000
005003300800000
004000009000600
034556780005000
300000000004400
020000000000000
0 0
```



(note que o segundo teste deste conjunto de entrada corresponde ao exemplo da Figura acima)

3. Saída de Dados

Seu programa deve produzir um arquivo de saída chamado IMAGEM.OUT. Para cada conjunto de teste do arquivo de entrada seu programa deve produzir três linhas no arquivo de 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 deve aparecer o número de construções presentes na imagem de teste, precedido de "Numero de construcoes:". A terceira linha deve ser deixada em branco. A grafia mostrada no Exemplo de Saída, abaixo, deve ser seguida rigorosamente (note que não são usados acentos).

Exemplo de Saída

```
Teste 1
Numero de construcoes: 0

Teste 2
Numero de construcoes: 4
```

(esta saída corresponde ao exemplo de entrada acima)

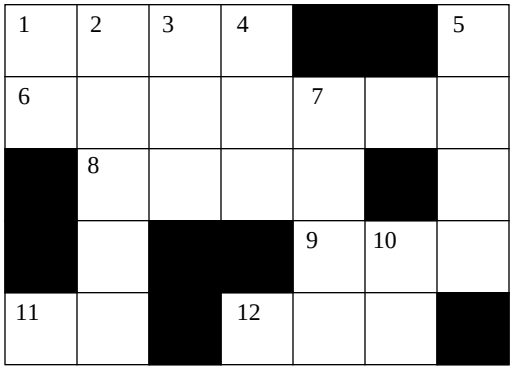
4. Restrições

$1 \leq M \leq 80$
$1 \leq N \leq 50$
M = 0 e N = 0 apenas para indicar o fim do arquivo de entrada

Palavras Cruzadas

O conhecido passatempo de palavras cruzadas é composto por uma grade retangular de quadrados brancos e pretos e duas listas de definições. Uma das listas de definições é para palavras escritas da esquerda para a direita nos quadrados brancos (nas linhas) e a outra lista é para palavras que devem ser escritas de cima para baixo nos quadrados brancos (nas colunas). Uma palavra é uma seqüência de *dois ou mais* caracteres do alfabeto. Para resolver um jogo de palavras cruzadas, as palavras correspondentes às definições devem ser escritas nos quadrados brancos da grade.

As definições correspondem às posições das palavras na grade. As posições são definidas por meio de números inteiros seqüenciais colocados em alguns quadrados brancos. Um quadrado branco é numerado se uma das seguintes condições é verificada: (a) tem como vizinho à esquerda um quadrado preto e como vizinho à direita um quadrado branco; (b) tem como vizinho acima um quadrado preto e como vizinho abaixo um quadrado branco; (c) é um quadrado da primeira coluna à esquerda e tem como vizinho à direita um quadrado branco; d) é um quadrado da primeira linha acima e tem como vizinho abaixo um quadrado branco. Nenhum outro quadrado é numerado. A numeração começa em 1 e segue seqüencialmente da esquerda para a direita, de cima para baixo. A figura abaixo ilustra um jogo de palavras cruzadas com numeração apropriada.



Um jogo de palavras cruzadas numerado corretamente.

Uma palavra *horizontal* é escrita em uma seqüência de quadrados brancos em uma linha, iniciando-se em um quadrado numerado que tem um quadrado preto à esquerda ou que está na primeira coluna à esquerda. A seqüência de quadrados para essa palavra continua da esquerda para a direita, terminando no quadrado branco imediatamente anterior a um quadrado preto, ou no quadrado branco da coluna mais à direita da grade.

Uma palavra *vertical* é escrita em uma seqüência de quadrados brancos em uma coluna, iniciando-se em um quadrado numerado que tem um quadrado preto acima ou que está na primeira linha acima. A seqüência de quadrados para essa palavra continua de cima para baixo, termi-

nando no quadrado branco imediatamente anterior a um quadrado preto, ou no quadrado branco da coluna mais abaixo da grade.

1. Tarefa

Sua tarefa é escrever um programa que recebe como entrada vários jogos de palavras cruzadas *resolvidas* e produz as listas de palavras verticais e horizontais que constituem as soluções.

2. Entrada de Dados

O arquivo CRUZ.IN contém vários conjuntos de teste. A primeira linha de um conjunto de testes contém dois inteiros positivos, M e N, que indicam respectivamente o número de linhas e o número de colunas do jogo de palavras cruzadas. Cada uma das M linhas seguintes contém N caracteres (caracteres do alfabeto ou o caractere '*'), correspondendo a um jogo de palavras cruzadas resolvido. O caractere '*' é utilizado para representar um quadrado preto. O final do arquivo de testes é indicado quando M = N = 0 (este último conjunto de testes não é válido e não deve ser processado).
O arquivo CRUZ.IN contém ao menos um conjunto de teste que deve ser processado.

Exemplo de Entrada

```
1 8
*PASCAL*
3 3
*M*
BIT
*L*
5 7
ATOS**J
MEMORIA
*COLE*V
*L**DIA
LA*VER*
0 0
```

3. Saída de Dados

Seu programa deve produzir um arquivo de saída chamado CRUZ.OUT. Para cada conjunto de teste do arquivo de entrada seu programa deve produzir duas listas, uma para as palavras horizontais e uma para as palavras verticais. A lista das palavras horizontais deve ser precedida por uma linha de cabeçalho onde está escrito "Horizontais:"; este cabeçalho só deve aparecer quando houver palavras horizontais no jogo (por exemplo, em um jogo com apenas uma coluna não há palavras horizontais). Da mesma forma, a lista das palavras verticais deve ser precedida por uma linha onde está escrito "Verticais:". As listas devem ser numeradas e apresentadas na ordem crescente de numeração da grade original. Deve ser deixada uma linha em branco após cada teste. A grafia mostrada no Exemplo de Saída, abaixo, deve ser seguida rigorosamente.

Exemplo de Saída

```
Teste 1
Horizontais:
  1. PASCAL

Teste 2
Horizontais:
  2. BIT
Verticais:
  1. MIL

Teste 3
Horizontais:
  1. ATOS
  6. MEMORIA
  8. COLE
  9. DIA
  11. LA
  12. VER
Verticais:
  1. AM
  2. TECLA
  3. OMO
  4. SOL
  5. JAVA
  7. REDE
  10. IR
```

(esta saída corresponde ao exemplo de entrada acima)

4. Restrições

$1 \leq M \leq 100$
$1 \leq N \leq 100$
M = 0 e N = 0 apenas para indicar o fim do arquivo de entrada

Trem ou Caminhão?

A produtora de refrigerantes CaraCola precisa enviar com freqüência grandes carregamentos para as suas distribuidoras em outros estados. Para isso ela pode utilizar uma transportadora que trabalha com caminhões ou uma transportadora que trabalha com trens. As duas transportadoras competem agressivamente para conseguir o serviço, mas seus custos dependem do momento (por exemplo, se há ou não caminhões disponíveis, etc.). A cada carregamento, a CaraCola consulta as duas transportadoras, que informam as condições de preço vigentes no momento, para o estado desejado. Sua tarefa é escrever um programa que, baseado nas informações das transportadoras, decida se o melhor é enviar o carregamento por trem ou por caminhão.

As transportadoras informam os seus custos na forma de duas variáveis, representando duas parcelas. Uma parcela é um custo fixo A que independe do peso do carregamento, e a outra parcela é um custo variável B que depende do peso do carregamento, em kilogramas. A CaraCola utiliza o peso do carregamento para calcular o custo dos transporte por trem e por caminhão e decidir qual empresa transportadora contratar. Por exemplo, suponha que a transportadora por trem informa que o seu custo fixo é A = R$ 450,00 e o seu custo por kilograma é B = R$ 3,50. Suponha ainda que a transportadora por caminhão informa que seu custo fixo é A = R$ 230,00 e o seu custo por kilograma é B = R$ 3,70. Neste caso, para um carregamento que pesa 2354 kg a CaraCola decide que é melhor fazer o envio por trem, pois 450 + 3,50 x 2354 < 230 + 3,70 x 2354. Se a diferença entre os custos for menor do que R$ 1,00 a CaraCola prefere o transporte por trem.

1. Tarefa

Sua tarefa é escrever um programa que recebe como entrada vários casos, cada um apresentando uma lista de custos, e determina se a CaraCola deve enviar o carregamento por trem ou por caminhão.

2. Entrada de Dados

O arquivo COLA.IN contém vários conjuntos de teste. Cada conjunto de teste é composto por uma linha, que contém cinco valores. O primeiro valor é um número inteiro positivo K que representa o peso, em kilogramas, do carregamento. Os quatro valores restantes são números reais A, B, C e D que representam os custos informados pelas empresas de transporte. A e B representam respectivamente o custo fixo e o custo variável por kilograma informado pela empresa que utiliza trem. C e D representam respectivamente o custo fixo e o custo variável por kilograma informado pela empresa que utiliza caminhão. Os custos são apresentados sempre com precisão de dois algarismos decimais. O final do arquivo de testes é indicado quando K = 0 (este último conjunto de testes não é válido e não deve ser processado).
O arquivo COLA.IN contém ao menos um conjunto de teste que deve ser processado.

Exemplo de Entrada

```
2354 450.00 3.50 230.00 3.70
1000 411.50 2.85 411.50 2.85
2327 325.00 3.10 557.50 3.00
0
```

ERRATA

A prova distribuída aos competidores continha um erro. O exemplo correto é:

```
2354 450.00 3.50 230.00 3.70
1000 411.50 2.85 411.50 2.85
2327 325.00 3.10 556.50 3.00
```

3. Saída de Dados

Seu programa deve produzir um arquivo de saída chamado COLA.OUT. Para cada conjunto de teste do arquivo de entrada seu programa deve produzir três linhas no arquivo de 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 deve aparecer resposta, no formato "envie por trem" ou "envie por caminhao". A terceira linha deve ser deixada em branco. A grafia mostrada no Exemplo de Saída, abaixo, deve ser seguida rigorosamente (note que não são usados acentos).

Exemplo de Saída

```
Teste 1
envie por trem

Teste 2
envie por trem

Teste 3
envie por caminhao
```

(esta saída corresponde ao exemplo de entrada acima)

4. Restrições

$1 \leq K \leq 5000$
$0 \leq A \leq 1000.00$
$0 \leq B \leq 1000.00$
$0 \leq C \leq 1000.00$
$0 \leq D \leq 1000.00$
K = 0 apenas para indicar o fim do arquivo de entrada