OBI2008

Caderno de Tarefas

Modalidade Programação • Nível 2, Fase 2

26 de abril de 2008

A PROVA TEM DURAÇÃO DE CINCO HORAS

LEIA ATENTAMENTE ESTAS INSTRUÇÕES ANTES DE INICIAR A PROVA

- Este caderno de tarefas é composto por 8 páginas (não contando esta folha de rosto), numeradas de 1 a 8. Verifique se o caderno está completo.
- A prova deve ser feita individualmente.
- É proibido consultar a Internet, livros, anotações ou qualquer outro material durante a prova. É permitida a consulta ao *help* do ambiente de programação se este estiver disponível.
- As tarefas têm o mesmo valor na correção.
- A correção é automatizada, portanto siga atentamente as exigências da tarefa quanto ao formato da entrada e saída de seu programa.
- Não implemente nenhum recurso gráfico nas suas soluções (janelas, menus, etc.), nem utilize qualquer rotina para limpar a tela ou posicionar o cursor.
- As tarefas **não** estão ordenadas, neste caderno, por ordem de dificuldade; procure resolver primeiro as questões mais fáceis.
- 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*. Para problemas diferentes você pode escolher trabalhar com linguagens diferentes, mas apenas uma solução, em uma única linguagem, deve ser submetida para cada problema.
- 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.
- 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*.
- 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
www.sbc.org.br
Fundação Carlos Chagas
www.fcc.org.br

Auto Estrada

Nome do arquivo fonte: `auto.c`, `auto.cpp` *ou* `auto.pas`

Certas regiões resolveram o problema de tráfego intenso com a construção de auto estradas, que são estradas contendo em geral quatro ou mais pistas de rolagem em cada sentido, de forma que um número grande de carros possa passar sem que ocorram congestionamentos. O problema das auto estradas é que, junto com os carros temos um aumento considerável de ruído nas imediações da pista, o que incomoda os moradores das regiões próximas.

A GoTo engenharia, uma empresa do ramo de construção especializada em obras de estradas, encontrou uma solução engenhosa para o problema: instalar grandes painéis defletores de som de cada lado da auto estrada para tentar minimizar o ruído percebido pelos vizinhos.

Os painéis são construídos em blocos contínuos de 10 metros lineares. A auto estrada também é dividida em blocos de 10 metros de extensão, sendo cada bloco descrito por um código, como definido abaixo:

- **P** - Pista, trecho em linha reta sem curvas ou saídas. Deve-se instalar um painel de cada lado da auto estrada.
- **C** - Curva, trecho em curva de 90 graus na auto estrada. Deve-se instalar dois painéis de concreto do lado externo da curva; o outro lado fica sem painel instalado.
- **A** - Acesso, trecho em linha reta no qual existe uma entrada ou uma saída a partir de um dos lados da auto estrada (mas não do outro). Deve-se instalar um painel no lado onde **não** existe o acesso.
- **D** - Duplo acesso, trecho em linha reta no qual existem dois acessos (entradas ou saidas, em qualquer combinação possível), um de cada lado da rodovia. Nenhum painel deve ser instalado nesse bloco da auto estrada.

Apesar de ser uma empresa formada por engenheiros, nenhum dos funcionários da GoTo sabe programar, de forma que eles decidiram contrataram você como consultor independente. Você deve escrever um programa para, dado um mapa da auto estrada, determinar quantos painéis defletores são necessários para cobrir toda a extensão dessa auto estrada.

Entrada

A entrada contém um único conjunto de testes, que deve ser lido do *dispositivo de entrada padrão* (normalmente o teclado). A primeira linha contém um inteiro C ($1 \leq C \leq 10^6$), indicando o comprimento da auto estrada, em blocos de 10 metros. A linha seguinte contêm C caracteres, cada letra descrevendo um bloco de 10 metros da auto estrada, como definido acima.

Saída

Seu programa deve imprimir, na *saída padrão*, uma única linha contendo um número inteiro, representando quantas unidades de painel são necessárias para cobrir toda a extensão da auto estrada.

Exemplo de entrada	**Exemplo de saída**
`5` `DAPCD`	`5`

Exemplo de entrada	**Exemplo de saída**
`8` `AACCAAPP`	`12`

Exemplo de entrada	**Exemplo de saída**
14 ADCCPPPPPAADCP	21

Informações sobre a pontuação

- Para um subconjunto dos casos de teste totalizando 30 pontos, $1 \leq C \leq 100$.
- Para um subconjunto dos casos de teste totalizando 55 pontos, $1 \leq C \leq 10^3$.

Chuva

Nome do arquivo fonte: `chuva.c`, `chuva.cpp` *ou* `chuva.pas`

A robótica causou uma grande revolução nos processos industriais no mundo todo; atualmente, vários tipos de robôs são usados na fabricação de carros, equipamentos eletrônicos e até mesmo utensílios domésticos.

Uma fábrica possui um robô de manutenção, que constantemente precisa ser deslocado entre setores diferentes para executar vários serviços. A movimentação do robô é feita por controle remoto: ele pode andar qualquer distância, mas apenas nas quatro direções cardeais (norte, sul, leste e oeste).

Robôs são feitos de metal, e por isso é ideal que eles evitem contato direto com a água. Assim, em dias chuvosos, é ideal que a trajetória do robô passe por dentro de galpões, debaixo de marquises e toldos, etc. para evitar sua exposição à chuva.

A sua tarefa é escrever um programa que, dadas as informações sobre as áreas cobertas e ponto inicial e final do robô, determine uma trajetória para o robô que minimize a porção do trajeto feita sob chuva.

Entrada

A entrada contém um único conjunto de testes, que deve ser lido do *dispositivo de entrada padrão* (normalmente o teclado). A primeira linha da entrada contém quatro inteiros X_i, Y_i, X_f e Y_f ($0 \leq X_i, Y_i, X_f, Y_f \leq 10^6$), indicando, respectivamente, a posição atual e a posição final do robô — o robô começa na posição (X_i, Y_i) e deve terminar na posição (X_f, Y_f).

A linha seguinte da entrada contém um único inteiro N ($0 \leq N \leq 1000$), indicando o número de áreas cobertas na fábrica. Cada uma das N linhas seguintes contém quatro inteiros X_1, Y_1, X_2 e Y_2 ($0 \leq X_1 < X_2 \leq 10^6$, $0 \leq Y_1 < Y_2 \leq 10^6$), indicando uma região coberta.

Uma região coberta é um retângulo de lados paralelos aos eixos tal que (X_1, Y_1) e (X_2, Y_2) são vértices opostos do retângulo. Duas áreas cobertas podem ter regiões comuns. O robô pode entrar e sair de uma área coberta por qualquer ponto de seu perímetro, e pode trafegar livremente dentro da área coberta.

Saída

Seu programa deve imprimir, na *saída padrão*, uma única linha, contendo um número inteiro indicando a menor distância que o robô precisa percorrer sob chuva.

Exemplo de entrada	**Exemplo de saída**
`0 0 4 3` `0`	`7`

Exemplo de entrada	**Exemplo de saída**

`2 5 5 0`
`1`
`0 0 1 5`
	`5`

Exemplo de entrada	**Exemplo de saída**

`4 5 5 0`
`2`
`0 0 1 5`
`0 0 3 2`
	`5`

Informações sobre a pontuação

- Em um conjunto de casos de teste que totaliza 30 pontos, $X_1, Y_1, X_2, Y_2, X_i, X_f, Y_i, Y_f \leq 10$ e $N \leq 5$.
- Em um conjunto de casos de teste que totaliza 55 pontos, $X_1, Y_1, X_2, Y_2, X_i, X_f, Y_i, Y_f \leq 1000$ e $N \leq 100$.

Ortografia

Nome do arquivo fonte: `ortografia.c`, `ortografia.cpp` *ou* `ortografia.pas`

Um serviço de busca na Internet está preocupado com a crescente taxa de erros de ortografia de seus usuários, tornando mais difíceis as buscas por palavras chaves, que constantemente contêm erros de algumas letras, devidos a má digitação ou má ortografia.

O serviço funciona com base num dicionário de palavras. O usuário deve inserir uma palavra num campo de um formulário; o serviço então procura esta palavra no dicionário e retorna conteúdo que tenha relação com a palavra.

Para contornar o problema de ortografia, você foi contratado para fazer um programa que tenta adivinhar qual palavra o usuário pretendia procurar, independente de haver erros de ortografia nela.

Para este problema vamos definir a *distância* entre duas palavras A e B como sendo o número de operações, descritas abaixo, necessárias para transformar A em B:

1. Retirar uma letra de A.
2. Adicionar uma letra a A, em qualquer posição.
3. Trocar qualquer letra de A por outra letra, na mesma posição.

O serviço de busca definiu que a palavra P fornecida pelo usuário pode se referir a uma palavra D do dicionário se está a uma distância de no máximo 2 de D.

Exemplos:

- A palavra 'tu' pode se referir à palavra do dicionário 'tubo', realizando duas vezes a operação 2.
- A palavra 'crto' pode se referir à palavra do dicionário 'corte', realizando uma vez a operação 2 e uma vez a operação 3.
- A palavra 'crto' pode se referir à palavra do dicionário 'curto', realizando uma vez a operação 2.
- A palavra 'hortgrafea' não pode se referir à palavra do dicionário 'ortografia'.

Você deve escrever um programa que, dado um dicionário de palavras, descubra para cada palavra fornecida pelo usuário a quais palavras do dicionário ela pode se referir, nas condições descritas acima.

Entrada

A entrada contém um único conjunto de testes, que deve ser lido do *dispositivo de entrada padrão* (normalmente o teclado). A primeira linha contém 2 inteiros N, M, representando respectivamente o número de palavras contidas no dicionário ($1 \leq N \leq 1000$) e o número de palavras a serem analisadas ($1 \leq M \leq 100$). Cada uma das N linhas seguintes conterá uma palavra pertencente ao dicionário. Cada uma das M linhas seguintes conterá uma palavra a ser analisada, fornecida pelo usuário. Cada palavra pode ter de 1 a 20 letras, contendo apenas letras de 'a' a 'z', minúsculas.

Saída

Seu programa deve imprimir, na *saída padrão*, M linhas, sendo uma linha para cada palavra fornecido pelo usuário. Cada linha deve conter todas palavras do dicionário às quais a palavra fornecida pode se referir. No caso de haver mais de uma palavra em uma linha da resposta, elas devem ser separadas por um espaço em branco, apareçendo na ordem que elas foram dadas na entrada, como pode ser visto no exemplo de saída abaixo. No caso de não haver nenhuma palavra em uma linha da resposta, deixe-a em branco.

Exemplo de entrada	**Exemplo de saída**
3 3 pato pateta caneca pat ccanecos pata	pato pato pateta

Informações sobre a pontuação

- Em um conjunto de casos de teste que totaliza 30 pontos, $N \leq 10$ e $M \leq 10$.
- Em um conjunto de casos de teste que totaliza 55 pontos, $N \leq 50$ e $M \leq 500$.

Frete da Família Silva

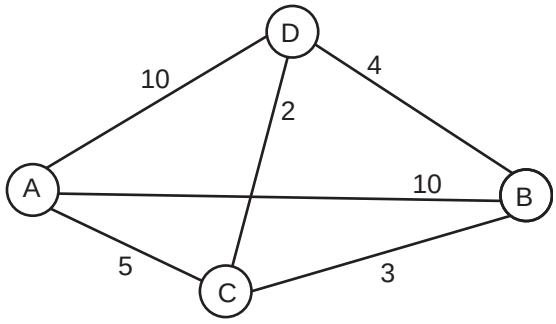
Nome do arquivo fonte: `frete.c`, `frete.cpp` *ou* `frete.pas`

Houve uma determinada época no planeta Terra em que a população estava grande demais, e determinadas medidas foram tomadas para sanar esse problema. Uma vez que as primeiras colônias já haviam se estabelecido no planeta Marte, todos os países concordaram em mandar para lá algumas pessoas. O presidente de Pizzalândia, Lagosta da Silva, era uma pessoa que valorizava a família, e decidiu que não ia separar famílias em nome dessa atitude. Resolveu, então, mandar uma família inteira para Marte. No caso, a dele mesmo, a família Silva, provavelmente a mais numerosa do planeta.

Tal família estabeleceu-se em Marte sem problemas, ainda mais com novas invenções que havia por lá. Uma delas era a pílula de nanicolina, substância descoberta naquele planeta, próximo à uma região onde existem pedras voadoras, pedras macias e até pedras falantes. Lendas dizem que algum outro ser extra-terrestre depositou a nanicolina ali num passado distante, enquanto visitava o planeta. O efeito da pílula de nanicolina é a diminuição de tamanho de quem a toma, por um determinado tempo. Tal pílula foi, então, produzida em escala industrial e hoje em dia é distribuída pelos governos marcianos aos colonos que lá residem.

A família Silva, todos os anos, encontra-se em alguma das muitas colônias em Marte para celebrar o aniversário da chegada deles ao planeta. O chefe da família é quem sempre paga o transporte de todos. O transporte é feito através de ônibus-flutuadores fretados. Como todos podem tomar pílulas da nanicolina e ficarem minúsculos, podemos dizer que dentro de cada ônibus-flutuador cabem infinitas pessoas, e que o efeito da pílula vai durar durante toda a viagem.

Assim, o preço de uma viagem de ônibus-flutuador entre duas colônias não depende do número de pessoas que viajam, sendo um preço fixo. Isso permite que algumas economias sejam feitas. Suponha que existam quatro colônias dos Silvas em Marte, ilustrados abaixo:



Os círculos representam as colônias, e as conexões entre elas representam as estradas existentes.

O número nas conexões representa o preço de uma viagem de ônibus-flutuador em qualquer direção. Ou seja, Uma viagem da colônia A direto para a colônia C (ou de C para A), custa 5 moedas de silício, não importa o número de passageiros.

Suponha que o grande encontro seja na colônia A. Se o chefe da família pagar o frete de B para A, de C para A e de D para A, vai acabar gastando 25 moedas.

Mas uma coisa que poderia ser feita, também, é: os Silvas das colônias B e D vão para a C. Da C, todos vão para a colônia A. Isso tudo teria um gasto de somente 10 moedas.

Este ano o número de colônias dos Silvas aumentou muito em Marte, e o chefe da família está muito preocupado com o dinheiro que vai gastar para pagar todas as viagens. Então ele contratou você, que é o melhor programador daquele planeta, a fazer um programa que recebe as informações a respeito das colônias, das estradas e dos fretes de ônibus-flutuadores, e determine qual é a menor quantidade de dinheiro necessária para custear o transporte de todos os Silvas para o encontro. O desespero do chefe da família é tanto que ele não se importa em qual colônia será o encontro, desde que os custos sejam minimizados.

Você pode assumir que:

- Entre duas colônias diferentes existe no máximo uma estrada direta.

- Sempre existe um caminho (de uma ou mais estradas) entre quaisquer duas colônias.

Entrada

A entrada contém um único teste, a ser lido da *entrada padrão*. A primeira linha contém dois inteiros: N e M ($2 \leq N \leq 1000$, $1 \leq M \leq 10.000$), que representam, respectivamente, o número de colônias e o número de estradas existentes. Depois, seguem M linhas com 3 inteiros: P, Q e U ($0 \leq P, Q \leq N-1$, $1 \leq U \leq 1000$), indicando que existe uma estrada de mão dupla entre as colônias P e Q, cujo custo do frete de viagem entre essas duas colônias é U moedas.

Saída

Seu programa deve imprimir, na *saída padrão*, um único inteiro, representando o número mínimo de moedas necessárias para custear o transporte de todos os Silvas à colônia onde será realizada o encontro.

Exemplo de entrada	**Exemplo de saída**
4 6 0 1 10 0 2 5 0 3 10 1 2 3 1 3 4 2 3 2	10

Exemplo de entrada	**Exemplo de saída**
4 6 0 1 1 0 2 1 0 3 1 1 2 3 1 3 4 2 3 2	3

Informações sobre a pontuação

- Em um conjunto de casos de teste que totaliza 30 pontos, $N \leq 10$ e $M \leq 100$.
- Em um conjunto de casos de teste que totaliza 55 pontos, $N \leq 100$ e $M \leq 1000$.