ToPAS'09
Torneio de Programação para Alunos do Secundário

Departamento de Ciência de Computadores

http://www.dcc.fc.up.pt/topas/

Conjunto de Problemas

Faculdade de Ciências da Universidade do Porto

15 de Maio de 2009

Este conjunto de problemas deverá conter sete (7) problemas e dezasseis (16) páginas.
Se faltar algum problema, por favor avise a organização.

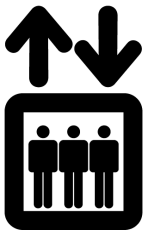
ToPAS'09
Torneio de Programação para Alunos do Secundário

Dep. Ciência de Computadores – FCUP
15 de Maio de 2009

Conteúdo

Problema A

Vai Subir?



A companhia de elevadores Suba-Suba optou por um novo o sistema de controlo de grupos de elevadores de forma a minimizar o consumo de energia. Para um grupo de dois elevadores, identificados como 1 e 2, o sistema enviará o elevador que se encontre mais próximo do andar de onde foi feita a chamada. Se os elevadores se encontram à mesma distância então será enviado o que encontra num andar mais elevado. Se ambos os elevadores se encontram no mesmo andar, accionará sempre o elevador 1. Nem sempre os elevadores estarão parados e disponíveis a serem chamados. Poderão estar em serviço, em manutenção ou avariados. Neste caso, enviará o único elevador disponível, ou um erro se ambos estiverem indisponíveis.

Tarefa

Pretende-se programar o sistema que controla a chamada de um grupo dois elevadores segundo a especificação dada. O programa receberá como input o número do andar em que um elevador foi chamado, bem como a situação actual de cada um dos elevadores; indicará o número do elevador melhor posicionado para efectuar o percurso. O input é constituído por três inteiros separados por um espaço: o número do andar de onde foi feita a chamada, a situação do elevador 1, seguida da situação do elevador 2. O número do andar onde foi chamado o elevador é um inteiro entre -100 e 100. A situação dos elevadores 1 e 2 poderá ser o número de um andar (que é também um valor entre -100 e 100) ou o número 999 para indicar que se encontra indisponível. O output é 0, 1 ou 2. O valor 0 é reservado para o caso de ambos os elevadores estarem indisponíveis. Caso contrário será indicado o número do elevador melhor posicionado para responder à chamada (1 ou 2).

Exemplo 1

Input

```
2 4 2
```

Output

```
2
```

Exemplo 2

Input
```
1 999 -1
```

Output
```
2
```

Exemplo 3

Input
```
3 1 5
```

Output
```
2
```

Exemplo 4

Input
```
4 6 6
```

Output
```
1
```

Exemplo 5

Input
```
7 999 999
```

Output
```
0
```

Problema B

Problemas de Agenda

As pessoas com agendas sobrecarregadas têm alguma dificuldade em marcar reuniões. Há as que dizem "posso das x às y, mas às y já tenho outra reunião" e as que dizem "não posso das z às t, mas às t já estou livre". As outras possibilidades não serão interessantes agora e para simplificar vamos supor que dizem sempre "posso".

Tarefa

Escrever um programa que, dada a duração (em horas) prevista para uma reunião entre duas pessoas e dois pares de inteiros A B e C D, verifica se é possível agendar a reunião e quando. Neste caso, a primeira pessoa diria "posso das A horas às B horas, mas às B já tenho outra reunião" e a segunda diria "posso das C horas às D horas, mas às D já tenho outra reunião", sendo A, B, C e D inteiros entre 9 e 19. Assume-se que A é menor do que B e que C é menor do que D.

A reunião tem de decorrer entre as 9 e as 19 horas e a sua duração é um inteiro. A primeira linha de dados tem essa duração e as duas seguintes os dois pares referidos. Conforme o caso, o programa escreve uma linha ou com a palavra `Impossivel` (sem acento) ou com um inteiro se a reunião só se puder realizar a uma hora ou com dois inteiros se houver flexibilidade (estes inteiros definem o intervalo em que se pode fixar o início da reunião).

Exemplo 1

Input

```
1
9 12
12 16
```

Output

```
Impossivel
```

Exemplo 2

Input

```
2
10 15
10 17
```

Output

```
10 13
```

Problema C

Dilema do Prisioneiro

"Dois suspeitos, A e B, são presos pela polícia. A polícia tem provas insuficientes para os condenar, mas, separando os prisioneiros, oferece a ambos o mesmo acordo: se um dos prisioneiros, confessando, testemunhar contra o outro e esse outro permanecer em silêncio, o que confessou sai livre enquanto o cúmplice silencioso cumpre 10 anos de sentença. Se ambos ficarem em silêncio, a polícia só pode condená-los a 6 meses de cadeia cada um. Se ambos traírem o comparsa, cada um leva 5 anos de cadeia. Cada prisioneiro faz a sua decisão sem saber que decisão o outro vai tomar, e nenhum tem certeza da decisão do outro. A questão que o dilema propõe é: o que vai acontecer? Como vai o prisioneiro reagir?"

Esta é a formulação original do "Dilema do Prisioneiro" feita em 1950, e desde então usado em Teoria dos Jogos para modelar situações tão banais, como o que fazem duas pessoas que acidentalmente trocam as suas malas, ou tão complexas, como o sistema que durante a Guerra Fria ficou conhecido por Mutual Assured Destruction (MAD).

Em Teoria dos Jogos são particularmente interessantes os modelos básicos que permitem jogos repetidos, com memória das jogadas anteriores. Imagine-se um jogo de cartas: as regras de uma jogada podem ser repetidas por várias mãos; os (bons) jogadores lembram-se das cartas jogadas anteriormente; e no final são somados os ganhos de cada mão.

O Dilema do Prisioneiro é também um jogo básico que pode ser jogado de um forma iterada. Em cada iteração os pontos do jogador são dados por uma matriz de ganhos. No exemplo clássico apresentado acima a matriz de ganhos seria a seguinte (tomando os meses de prisão como perdas):

A ; B	Cooperar	Desertar
Cooperar	-6 ; -6	-120 ; 0
Desertar	0 ; -120	-60 ; -60

A versão iterada do Dilema do Prisioneiro modela muitas situações de cooperação e confronto, desde as explicações para as relações entre os seres vivos baseadas na teoria da evolução Darwiniana, às teorias económicas para as relações entre as empresas.

Tarefa

Pretende-se escrever um programa para calcular os pontos de cada jogador do Dilema do Prisioneiro iterado e determinar o vencedor. Como input é dada a matriz de ganhos seguida de uma sequência

de n jogadas. A matriz é dada por linhas de 2 valores, respectivamente os ganhos dos jogadores A e B, para os seguintes casos: A e B cooperam; A coopera e B deserta; A deserta e B coopera; A e B desertam. Segue-se uma linha com o número n de jogadas e n linhas com um par de valores 0 ou 1 (0 se desertam e 1 se cooperam) respectivamente para os jogadores A ou B.

O programa indicará o número de pontos acumulado por cada um dos jogadores. Cada linha terá o nome dum jogador (A ou B) seguido dum espaço e da respectiva pontuação, sendo os jogadores apresentados por ordem decrescente de pontuação. Se os jogadores empatam então terá apenas uma linha com referência AB seguida dum espaço e do valor das pontuação de ambos.

Exemplo 1

Input

```
-1 -1
-20 0
0 -20
-10 -10
4
1 0
0 1
1 1
0 0
```

Output

```
AB -31
```

Exemplo 2

Input

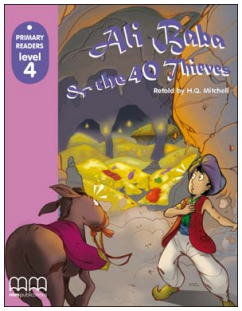
```
5 5
3 7
7 3
2 2
3
1 1
1 1
1 0
```

Output

```
B 17
A 13
```

Problema D

Ali Baba e a Biometria



A voz funciona como uma impressão digital para identificação de cada indíviduo, à semelhança da íris ou do seu DNA. As suas características únicas são determinadas quer pela forma como as cordas vocais vibram quer por padrões resultantes de outros aspectos físicos do indíviduo. Como seria a história de Ali Baba se a caverna do tesouro estivesse protegida por sistema biométrico baseado em reconhecimento de voz? Não bastaria pronunciar a fórmula mágica "Abre-te Sésamo". Mas, conseguiria tirar algum partido do facto do sistema de autenticação utilizar medidas probabilísticas e, consequentemente, poder tomar decisões erradas (os "falsos positivos" que serviriam os seus interesses) ?

Tarefa

Pretende-se analisar uma sequência interacções de Ali Baba com o sistema biométrico da caverna e escrever um programa para determinar o número máximo de vezes seguidas que conseguiu abrir a porta sem dificuldades. O programa analisará um registo de ocorrências em que `1 1` designa *abertura sem problemas* e `1 0` um *insucesso*. Depois dum insucesso haverá sempre uma sequência de zero ou mais tentativas falhadas (cada uma representada por `0`) que terminará por `1` (sucesso) ou por `2` (desistência). Cada linha de input apresenta uma ocorrência, com excepção da última linha que é sempre `-1 -1`. O output será uma linha com o valor pedido.

Exemplo

Input

```
1 1
1 0
0
0
1
1 1
```

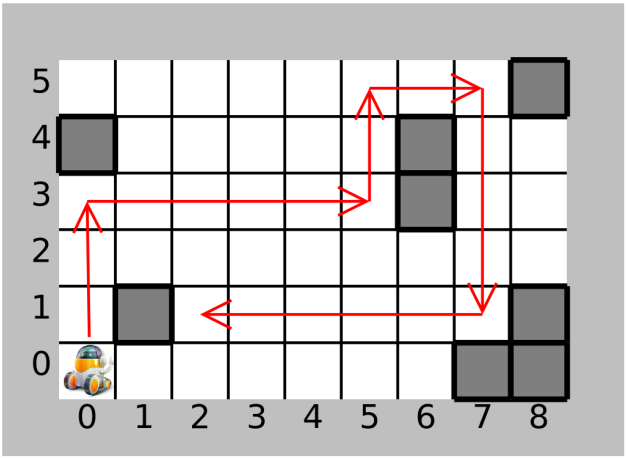
```
1 1
1 0
1
1 1
1 0
0
2
1 0
0
0
1
1 1
1 1
1 0
2
1 1
-1 -1
```

Output

```
2
```

Problema E

Robôs Ricochete



A fábrica *Robôs Ricochete, Lda.* produz robôs autónomos para limpeza de grandes superfícies em horário pós-laboral. Os robôs movimentam-se sobre uma grelha no plano executando sequências das seguintes instruções:

F deslocar-se em frente até encontrar um obstáculo;

D rodar sobre si mesmo 90° à direita;

E rodar sobre si mesmo 90° à esquerda.

A figura ilustra um percurso dum robô numa grelha 9×6, começando no canto inferior esquerdo e executando a sequência "F, D, F, E, F, D, F, D, F, D, F".

Pretende-se determinar a distância total percorrida pelo robô (número de células visitadas); são dados a dimensão da grelha, a posição dos obstáculos e sequência de instruções. Para simplificar o enunciado, supõe-se que cada obstáculo ocupa exactamente uma célula da grelha; supõe-se ainda que o robô se encontra inicialmente no canto inferior esquerdo (coordenada $(0, 0)$) voltado para cima (i.e. na direcção do vector $(0, 1)$).

Tarefa

Escrever um programa para determinar a distância total percorrida pelo robô.

Os dados dos problema são introduzidos pela seguinte ordem, um valor por cada linha: o número de colunas e o número de linhas da grelha seguido do número de obstáculos e as coordenadas de cada obstáculo (coluna, linha); por último, a sequência de instruções do robô (usando caracteres `F`,

`D`, `E`). Pode assumir que a grelha tem dimensão máxima 30×30 e que o programa do robô tem no máximo 50 instruções.

A saída é um inteiro que representa a distância total percorrida pelo robô.

Exemplo

Input

```
9
6
8
0
4
1
1
6
3
6
4
7
0
8
0
8
1
8
5
FDFEFDFDFDF
```

Output

```
21
```

Problema F

Sentemo-nos

Nalguns cinemas, salas de espectáculo e recintos desportivos não há lugares marcados. As pessoas vão escolhendo lugares da sua preferência e, por vezes, acabam por "dar um jeitinho", mudando-se para a cadeira do lado para que um grupo se possa sentar na sua fila. Interessa-nos observar a dinâmica duma fila.

As pessoas chegam sozinhas ou em grupo e apresentam-se pela esquerda ou pela direita. Para que um grupo se sente, é necessário haver lugares suficientes na fila; se não, procurará outra fila, e não será muito importante para o que se segue. Um grupo não se separa. A primeira pessoa do grupo ocupa um lugar da sua preferência e as restantes sentam-se à sua esquerda se o grupo entrar pela esquerda, ou à sua direita se entrar pela direita. O lugar que a primeira pretende ocupar está sempre livre. Se não houver lugares para todos mas a posição imediatamente ao lado da ocupada pelo primeiro estiver livre, este desloca-se de modo a criar lugares para os que faltam. Se mesmo assim não for possível sentar todos, então o grupo senta-se numa das extremidades (para evitar mais confusão), ocupando os lugares mais interiores. Preferencialmente, senta-se na extremidade em que se encontra desde que tal não obrigue à deslocação de mais pessoas do que seria necessário se se sentasse na outra extremidade. Nos deslocamentos, as posições relativas das pessoas sentadas anteriormente mantêm-se, deslocando-se apenas as pessoas que estiverem mais perto da extremidade em que o grupo ficará. Para isso, ocupam lugares adjacentes que estejam livres ou que fiquem livres porque outras se deslocam também.

Ninguém terá de se deslocar para dar lugar a uma pessoa que não esteja em grupo. Não haverá mudanças de fila.

Tarefa

Escrever um programa para analisar uma sequência de chegadas a uma fila e indicar a disposição final dos grupos nessa fila. Os grupos são identificados pelo número de ordem de chegada (contam também os que vão ter de procurar outra fila).

Na primeira linha de input tem um inteiro (positivo e menor ou igual a 50) que representa o número de lugares da fila. Nas linhas seguintes tem uma sequência de ternos de inteiros `E N L` em que: `E` é 0 ou 1 conforme o grupo se apresente pela esquerda ou direita, `N` (maior ou igual a 1) é o número de elementos do grupo e `L` identifica o lugar que o primeiro elemento pretende. O número que

identifica esse lugar tem por referência a extremidade em que o grupo se encontra (por exemplo, `0 4 3` significa que o primeiro elemento pretende a terceira posição a contar da esquerda, enquanto em `1 4 3` seria a terceira a contar da direita). O cenário termina por `0 0 0`.

O output é a descrição do estado final da fila, vista da esquerda para a direita, uma posição em cada linha: as posições livres terão um traço; as ocupadas terão o identificador do grupo a que a pessoa pertence.

Exemplo

Input

```
14
0 1 2
1 4 6
1 3 10
0 10 1
1 3 7
0 2 8
0 0 0
```

Output

```
5
5
5
1
3
3
3
-
2
2
2
2
6
6
```

Problema G

Linhas Coloridas

No jogo de tabuleiro *Color Lines*, o objectivo é alinhar cinco ou mais bolas da mesma cor, tantas vezes quantas conseguir, para contabilizar pontos. O tabuleiro tem dimensão 9×9. O alinhamento pode ser vertical, horizontal ou em qualquer diagonal.

Normalmente o jogo começa com cinco bolas no tabuleiro posicionadas ao acaso. Em cada jogada, o jogador selecciona uma bola qualquer e a posição para onde a quer mover. A bola desloca-se para essa posição desde que ela esteja livre e exista um caminho livre até lá (esse caminho é sempre uma sequência de passos unitários horizontais e/ou verticais quaisquer). Na nova posição, a bola pode completar um ou vários alinhamentos, cada um com 5 ou mais bolas (como nos exemplos). Sempre que tal acontece, o jogador acumula pontos e as bolas envolvidas saem do tabuleiro. Só saem bolas que pertençam a um alinhamento de 5 ou mais bolas. Na situação ilustrada pela figura mais à direita, completaria um alinhamento de 6 bolas e um de 5 (e sairiam 10 bolas do tabuleiro).

Quando não perfaz alinhamentos, entram novas bolas, ocupando posições aleatórias. O jogo termina quando o tabuleiro ficar sem qualquer espaço livre.

Pretende-se apenas saber se o jogador consegue completar algum alinhamento numa certa jogada e, caso o consiga, quantas bolas sairão do tabuleiro (antes de entrarem outras).

Tarefa

Escrever um programa para simular uma jogada a partir dum certo estado e indicar quantas bolas saem do tabuleiro nessa jogada.

Cada posição do tabuleiro é identificada por um par (I,J) de inteiros entre 1 e 9, sendo I o número da linha e J o da coluna: (1,1),(1,9),(9,9) e (9,1) são os cantos inferior esquerdo, inferior direito, superior direito e superior esquerdo, respectivamente. Em cada posição tem um inteiro (de 1 a 7).

O input começa pelo estado do tabuleiro, dado por linhas, do topo para a base (0 significa posição livre). As duas linhas seguintes dão a posição da bola seleccionada e da célula para onde a vai deslocar, se conseguir.

O output é uma linha com o número de bolas que sairam (possivelmente 0).

Os exemplos seguintes referem-se às jogadas esquematizadas nas duas figuras mais à esquerda (as cores azul claro, azul escuro, verde, vermelho, laranja, rosa, e amarelo são representadas por inteiros de 1 a 7).

Exemplo 1

Input

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

Output

```
6
```

Exemplo 2

Input

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

Output

```
6
```