OBI2006

Caderno de Tarefas

Modalidade Programação • Nível 2 • Fase Nacional

A PROVA TEM DURAÇÃO DE CINCO HORAS

LEIA ATENTAMENTE ESTAS INSTRUÇÕES ANTES DE INICIAR A PROVA

- Este caderno de tarefas é composto por 4 páginas (não contando esta folha de rosto), numeradas de 1 a 4. 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

Penalidade mínima

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

A Sra. Bastos é uma elaboradora de passatempos matemáticos e pediu para que você criasse um programa que conseguisse jogar de forma eficiente a sua mais nova criação.

O jogo consiste em um tabuleiro formado por casas dispostas em N linhas por N colunas. Cada casa contém um inteiro não-negativo. No começo do jogo, uma peça é colocada na casa localizada no canto superior esquerdo, ou seja, na posição (1,1). O objetivo do jogo é mover a peça até a casa localizada no canto inferior direito (posição (N,N)) somente movendo um único quadrado para baixo ou para a direita em cada passo. Além disso, a peça não pode ser colocada em nenhum quadrado que contenha o número zero.

O *custo* do caminho utilizado para percorrer o tabuleiro corresponde ao produto de todos os números das casas percorridos no caminho. A *penalidade* é definida utilizando a representação decimal do custo, sendo representada pelo número de dígitos zeros, contados da direita para a esquerda, antes do primeiro dígito diferente de zero. Por exemplo, um custo igual a 501000 tem penalidade 3, e um custo igual a 501 tem penalidade zero.

O objetivo do jogo é conseguir chegar à casa (N,N) através de um caminho "otimizado". Dizemos que o caminho foi otimizado se a penalidade for mínima.

Tarefa

Escreva um programa que, dado um tabuleiro, determine a penalidade do custo otimizado.

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 um inteiro N que indica o número de linhas e colunas do tabuleiro ($1 \leq N \leq 1000$). As N linhas seguintes contêm N inteiros I cada ($1 \leq I \leq 1000000$), que representam o valor da casa do tabuleiro naquela posição. Existe pelo menos uma solução possível para todos os casos de teste.

Saída

Seu programa deve imprimir, na *saída padrão*, uma única linha, contendo a penalidade do custo "otimizado".

Entrada	**Entrada**	**Entrada**

`3`
`1 2 3`
`4 5 6`
`7 8 9`
	`3`
	`5 7 6`
	`4 0 1`
	`3 2 5`
		`4`
		`1 3 0 0`
		`0 8 2 25`
		`6 5 0 3`
		`0 15 7 4`

Saída	**Saída**	**Saída**
`0`	`1`	`2`

Lobo Mau

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

Na fazenda do Sr. Amarante existe um certo número de ovelhas. Enquanto elas estão dormindo profundamente, alguns lobos famintos tentam invadir a fazenda e atacar as ovelhas. Ovelhas normais ficariam indefesas diante de tal ameaça, mas felizmente as ovelhas do Sr. Amarante são praticantes de artes marciais e conseguem defender-se adequadamente.

A fazenda possui um formato retangular e consiste de campos arranjados em linhas e colunas. Cada campo pode conter uma ovelha (representada pela letra 'k'), um lobo (letra 'v'), uma cerca (símbolo '#') ou simplesmente estar vazio (símbolo '.'). Consideramos que dois campos pertencem a um mesmo *pasto* se podemos ir de um campo ao outro através de um caminho formado somente com movimentos horizontais ou verticais, sem passar por uma cerca. Na fazenda podem existir campos vazios que não pertencem a nenhum pasto. Um campo vazio não pertence a nenhum pasto se é possível "escapar" da fazenda a partir desse campo (ou seja, caso exista um caminho desse campo até a borda da fazenda).

Durante a noite, as ovelhas conseguem combater os lobos que estão no mesmo pasto, da seguinte forma: se em um determinado pasto houver mais ovelhas do que lobos, as ovelhas sobrevivem e matam todos os lobos naquele pasto. Caso contrário, as ovelhas daquele pasto são comidas pelos lobos, que sobrevivem. Note que caso um pasto possua o mesmo número de lobos e ovelhas, somente os lobos sobreviverão, já que lobos são predadores naturais, ao contrário de ovelhas.

Tarefa

Escreva um programa que, dado um mapa da fazenda do Sr. Amarante indicando a posição das cercas, ovelhas e lobos, determine quantas ovelhas e quantos lobos estarão vivos na manhã seguinte.

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 dois inteiros R e C que indicam o número de linhas ($3 \leq R \leq 250$) e de colunas ($3 \leq C \leq 250$) de campos da fazenda. Cada uma das R linhas seguintes contém C caracteres, representando o contéudo do campo localizado naquela linha e coluna (espaço vazio, cerca, ovelha ou lobo).

Saída

Seu programa deve imprimir, na *saída padrão*, uma única linha, contendo dois inteiros, sendo que o primeiro representa o número de ovelhas e o segundo representa o número de lobos que ainda estão vivos na manhã seguinte.

Entrada

```
6 6
...#..
.##v#.
#v.#.#
#.k#.#
.###.#
...###
```

Saída

```
0 2
```

Entrada

```
8 8
.######.
#..k...#
#.####.#
#.#v.#.#
#.#.k#k#
#k.##..#
#.v..v.#
.######.
```

Saída

```
3 1
```

Entrada

```
9 12
.###.#####..
#.kk#...#v#.
#..k#.#.#.#.
#..##k#...#.
#.#v#k###.#.
#..#v#....#.
#...v#v####.
.####.#vv.k#
.......####
```

Saída

```
3 5
```

Sub-seqüências

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

Uma *subseqüência* de uma *seqüência de caracteres* S é definida como uma seqüência de caracteres de S, não necessariamente consecutivos, na mesma ordem em que eles ocorrem na seqüência original.

Dadas duas seqüências de caracteres, S_1 e S_2, dizemos que S_1 possui grau N de independência em relação a S_2 se, dada qualquer subseqüência de tamanho N de S_1, não é possível formar tal subseqüência a partir de S_2.

Por exemplo, o grau de independência da seqüência S_1=‘`ababaa`’ em relação à seqüência S_2=‘`abbaa`’ é igual a 3, pois todas as subseqüências de S_1 de tamanho 1 (‘`a`’, ‘`b`’) e todas as subseqüências de tamanho 2 (‘`aa`’, ‘`ab`’, ‘`ba`’, ‘`bb`’) podem ser formadas a partir de S_2, mas a subseqüência ‘`bab`’, de tamanho 3, não pode ser formada a partir de S_2.

Tarefa

Escreva um programa que, dadas duas seqüências S_1 e S_2, determine o grau N de independência de S_1 em relação a S_2.

Entrada

A entrada contém um único conjunto de testes, que deve ser lido do *dispositivo de entrada padrão* (normalmente o teclado). A entrada contém três linhas. A primeira linha contém dois inteiros N e M que indicam respectivamente o comprimento da seqüência S_1 (1 <= N <= 2000) e o comprimento da seqüência S_2 (1 <= M <= 2000). A segunda linha contém a seqüência S_1 e a terceira linha contém a seqüência S_2. As seqüências são formadas somente pelas letras minúsculas sem acento (’a’ - ’z’). As seqüências possuem no máximo 2000 caracteres. Sempre existe uma solução para os casos de teste fornecidos.

Saída

Seu programa deve imprimir, na *saída padrão*, uma única linha, conténdo o grau N de indepedência de S_1 em relação a S_2.

Entrada	**Entrada**	**Entrada**
`ababaa` `abbaa`	`babab` `babba`	`banana` `anbnaanbaan`
Saída	**Saída**	**Saída**
`3`	`3`	`5`