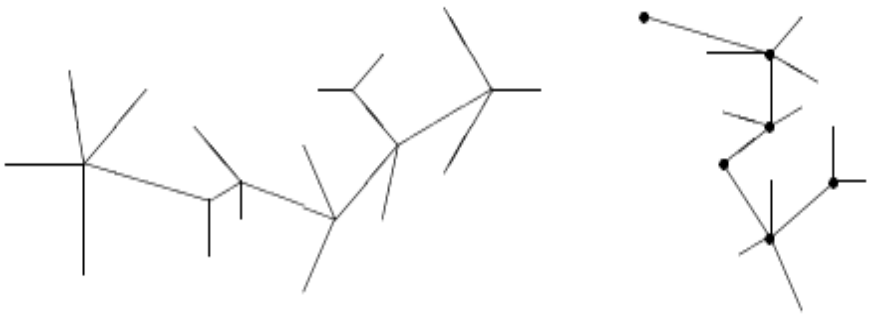
V - Maratona Doméstica da UDESC (Edileuza)
Departamento de Ciência da Computação
Caderno Oficial- set/2007

Problema A: Caterpillar (Lagarta)

Um grafo não-orientado é chamado de "lagarta" (*caterpillar* em inglês) se ele é conexo, não tem ciclos, e existe um caminho no grafo onde cada nó ou está neste caminho, ou é um vizinho de um nó neste caminho. Isto é, para ser um *cartepillar*, todo nó está num ramo principal ou caminho, ou é um vizinho imediato a este, em uma aresta de distância. Este caminho é chamado de *espinha da lagarta* e a espinha pode não ser única. Vocês vão simplesmente verificar grafos e ver se eles são lagartas. Por exemplo, o grafo abaixo a esquerda não é uma lagarta, mas o da direita o é. Uma possível espinha é mostrada por pontos.



Entrada:

Haverão múltiplos casos de teste. Cada caso inicia com uma linha contendo *n* indicando o numero de nós, numerados de 1 até *n* (um valor n=0 indica o fim das entradas). A próxima linha irá conter um inteiro *e* indicando o número de arestas. Iniciando na linha seguinte, estarão os pares n1 n2 indicam uma aresta não direcionada entre os nós n1 e n2. Isto é, as arestas são definidas pela seqüência de pares n1 a n2. Esta informação poderá se estender por várias linhas. Você pode assumir que ***n*** <= 100 e que ***e*** <= 300. Não assuma que os grafos nos testes são conexos ou acíclicos.

Saída:

Para cada caso de teste gerar uma linha de saída. Esta linha deverá ser uma das seguintes:

Graph g **is a caterpillar.**
Ou
Graph g **is not a caterpillar.**

O que for correto, onde *g* é o numero do grafo, iniciando em 1.

Exemplo de Entrada:
22

```
21
1 2 2 3 2 4 2 5 2 6 6 7 6 10 10 8 9 10 10 12 11 12 12 13 12 17 18 17
15 17 15 14 16 15 17 20 20 21 20 22 20 19
16
15
1 2 2 3 5 2 4 2 2 6 6 7 6 8 6 9 9 10 10 12 10 11 10 14 10 13 13 16 13
15
0
```

Exemplo de Saída:

```
Graph 1 is not a caterpillar.
Graph 2 is a caterpillar.
```

Problema B: Rápido com a Torta

A Pizzaria Pizazz é orgulhosa por entregar as pizzas aos clientes o mais rápido possível. Infelizmente, devido a contenção de gastos, eles só puderam contratar um motorista para as entregas. Ele espera por 1 ou mais (até 10) pedidos serem processados antes de iniciar as entregas. É desnecessário dizer que ele desejaria fazer a menor rota para entrega das mercadorias e retorno a pizzaria. Mesmo que isto implique em passar pela(s) mesma(s) localidades, ou mesmo pela pizzaria, mais de uma vez ao fazer o caminho. Você foi comissionado para fazer um programa para ajudá-lo.

Entrada:

A entrada consistirá em múltiplos casos de teste. A primeira linha irá conter um único inteiro n indicando o número de pedidos a entregar, onde 1<= ***n*** <=10. Após esta, existirão n+1 linhas, cada linha contendo n+1 inteiros indicando o tempo de viagem entre a pizzaria (numerada como 0) e as demais localidades (números 1 até n). Assim, o j-ésimo valor na i-ésima linha, indica o tempo para ir diretamente da localidade ***i*** até a localidade ***j*** sem visitar qualquer outra localidade no caminho. Note que podem haver caminhos mais rápidos entre *i* e *j* via outras localidades devido a diferentes limites de velocidades permitidas, distâncias, semáforos, etc. Adicionalmente, os tempos nem sempre são simétricos, isto é, o tempo de ir direto de *i* até *j* pode não ser o mesmo para ir de *j* até *i*. Um valor de entrada de n=0 irá terminar as entradas.

Saída:
Para cada caso de teste, você deverá ter como saída apenas um número, indicando o mínimo tempo para entregar todas as pizzas e retornar a pizzaria.

Exemplo de Entrada:

```
3
0 1 10 10
1 0 1 2
10 1 0 10
10 2 10 0
```

```
0
```

Exemplo de Saída:
```
8
```

Problema C: Mahershalalhashbaz, Nebuchadnezzar, e Billy Bob Benjamin vão para as Regionais

A ACM (Association for Computing Machinery) está considerando fazer novas regras para suas competições regionais de programação, em parte, para resolver alguns problemas.

O programa que imprime os crachás para cada time foi projetado para utilizar o mesmo tamanho de fonte de letras para imprimir todos os crachás de um mesmo time. Entretanto, isto significa que, se um membro do time tiver um nome muito longo então, uma fonte muito pequena será utilizada para imprimir todos os crachás do time. Isto não parecerá muito agradável para alguém que tenha um nome muito curto como por exemplo “AL”.

A solução inicial proposta pela ACM era impor um limite para o tamanho do nome dos competidores. Alguém levantou que isto poderia discriminar times de certas regiões onde nomes longos são mais comuns. Por exemplo, crianças que tem atores conhecidos nascidos na mesma cidade são mais prováveis de terem os nomes dos mesmos adicionados aos seus – imagine quantas crianças de Oakland (Califórnia-USA) foram batizadas com o nome de Mahershalalhashbaz Ali, desde que este se tornou um dos astros da série de televisão ***The 4400***.

Como solução intermediária, a ACM decidiu mudar a regra para exigir que “*nenhum membro de um time pode ter o comprimento do nome distante mais do que duas unidades do comprimento médio do tamanho dos nomes de todos os membros do seu time*”. Usando esta regra, um time constituído por “MAHERSHALALHASHBAZ”, “AL” e “BILL” deveria ser desqualificado (o tamanho médio dos nomes é 8, pois os nomes de AL e BILL estão a uma distância do comprimento médio superior a 2). Entretanto, “MAHERSHALALHASHBAZ”,“NEBUCHADNEZZAR”, e “BILLYBOBBENJAMIN” seria aceitável (o tamanho médio é 16 e o comprimento dos nomes de todos os membros do time está a uma distância deste número menor ou igual que 2).

Dados os nomes de n estudantes, determine se eles podem ser colocados em times de k membros cada, de forma que cada time cumpre as novas normas da ACM.

Entrada:

A entrada irá consistir de múltiplos casos de teste. Cada caso inicia com uma linha que consiste em dois inteiros positivos, ***n*** e ***k***. Onde n <= 1000 e k <= 8 e ***n*** é divisível por ***k*** . Em seguida tem-se n linhas, cada uma contendo um único nome todo em letras maiúsculas, sem espaços em branco antes, após ou no meio do nome. Estes são os nomes dos estudantes que precisam ser organizados em times de tamanho k cada um. Nenhum nome pode exceder 80 caracteres. O último caso de teste é seguido por uma linha contendo dois zeros.

Saída:

Para cada caso de teste, haverá uma saída com o número do teste(iniciando em 1) seguido pela palavra “yes” (significando que é possível organizar os estudantes em times de tamanho k e que nenhum estudante do time tem nome cujo tamanho é em distância de 2 da média do tamanho dos nomes naquele time), ou “no” se não for possível. Use o formato mostrado no exemplo de saída. Insira uma linha em branco entre os casos.

Exemplo de Entrada:

```
3 3
MAHERSHALALHASHBAZ
AL
BILL
6 3
MAHERSHALALHASHBAZ
AL
NEBUCHADNEZZAR
BILL
BILLYBOBBENJAMIN
JILL
0 0
```

Exemplo de Saída:

```
Case 1: no

Case 2: yes
```

Problema D: Labirintos Esquerdos

“*As instruções de virar sempre a esquerda lembra-me que este era um procedimento comum para achar o vão central de certos labirintos*” em *The garden of forking paths* (O jardim dos caminhos bifurcados) by Jorge Luis Borges,

O problema:

Um bibliotecário exumou um vasto catálogo de labirintos na Biblioteca de Babel. É nossa obrigação classificar todos eles e nós contamos com sua ajuda. Os planos dos labirintos já estão digitalizados e necessitamos que você escreva um programa para decidir se o vão central do labirinto pode ser atingido seguindo o simples procedimento de virar sempre a esquerda em qualquer interseção, e então este pode ser chamado de um “*labirinto esquerdo*”.

O processo de digitalização reduz o plano/mapa de um labirinto a uma grade de células, sendo cada célula é um bloco de parede ou simplesmente o chão. Paredes são blocos de células formando entre estes corredores verticais ou horizontais. Cada labirinto tem uma única entrada, um furo na parede exterior. E um único vão central.

O vão central difere dos corredores pelo formato das células: uma célula de chão no corredor tem pelo menos 2 blocos de paredes, uma em cada lado. Você pode assumir que cada plano/mapa tem apenas um labirinto e que suas paredes externas podem ser contornadas sem sair do plano.

Entrada:

As duas primeiras linhas de entrada são inteiros menores do que 100,

n – o número de linhas no mapa

m – o número de caracteres por linha no mapa

As linhas seguintes, cada uma com m caracteres, tem apenas dois caracteres válidos:

- (cancela ou cerquilha) representando um bloco de paredes;

. - (ponto) representando parte do chão.

Saída:

Seu programa deve escrever YES se um dado labirinto for um labirinto esquerdo ou não, caso contrário.

Exemplo de Entrada:

```
30
50
..................................................
..................................................
..................................................
.....##############.............############.....
.....#.........#..#............#.........#.....
.....#.#######.#.##............#.########.#.....
.....#.#.....#....#............#.#......#.#.....
.....#.#.#####.####............#.#.####.#.#.....
.....#.#.....#...#################.#.##.#...#.....
.....#.#####.###.......................#.#.#.....
.....#.........#.###############.#.##########.....
.....#########.#.#.#..........#.#..#............
............#.#.#.#..........###.##............
............#.#.#.#..........#....#............
............#.#...#..........#.##.#............
```

Exemplo de Saída:

YES

Problema E: Busca Maluca

Muitas pessoas gostam de resolver quebra-cabeças difíceis, alguns dos quais podem leva-los a loucura. Um tipo de quebra cabeças pode ser achar um número primo escondido em um texto. Este número pode ser o número de diferentes cadeias de caracteres (substrings) de um dado tamanho, que existam no texto. Como você descobrirá logo, você realmente necessita da ajuda de um computador e de um bom algoritmo para resolver este quebra-cabeças.

Problema:

Sua tarefa é escrever um programa no qual são dados: N tamanho das cadeias de caracteres, NC o número de caracteres diferentes que possam ocorrer no texto e o próprio texto. Este programa deve determinar o número de diferentes cadeias (substrings) de tamanho N que aparecem no texto.

Como um exemplo, considere N=3, NC=4 e o texto "daababac". As diferentes cadeias (substrings) de tamanho 3 que possam ser encontradas neste texto são: "daa"; "aab"; "aba"; "bab"; "bac". Neste caso, a resposta deve ser 5.

Entrada:

A primeira linha do arquivo de entrada consiste em dois números, N e NC, separados por exatamente um espaço. Esta é seguida pelo texto onde a busca tem início. Você pode assumir que o número de cadeias (substrings) formadas pelo possível conjunto de caracteres não exceda a 16 milhões.

Saída:

O programa deve ter no arquivo de saída apenas um inteiro correspondente ao número de diferentes cadeias (substrings) de tamanho N no texto dado.

Exemplo de arquivo de entrada:

3 4
daababac

Exemplo de arquivo de saída:

5

Problema F: Partição em Fração Unitária

Uma fração a qual o numerador é **1** e o denominador é um inteiro positivo, é chamada de fração unitária. A representação de um número positivo racional p/q como a soma finita de muitas frações unitárias é chamada de partição de p/q em frações unitárias. Por exemplo, 1/2 + 1/6 é a partição de 2/3 em frações unitárias. A ordem da adição é desconsiderada. Por exemplo, nós não distinguimos 1/6 + 1/2 de 1/2 + 1/6 .
Dados quatro inteiros positivos ***p, q, a***, e ***n.*** Conte o número de partições de p/q em frações unitárias satisfazendo as duas condições a seguir:

- A partição é a soma de até **n** frações unitárias
- Produto dos denominadores das frações unitárias é menor ou igual que ***a***.

Por exemplo, se (p,q,a,n) = (2,3,120,3), você deverá reportar 4 pois:

$$\begin{aligned}\frac{2}{3} &= \frac{1}{3} + \frac{1}{3} \\ &= \frac{1}{2} + \frac{1}{6} \\ &= \frac{1}{4} + \frac{1}{4} + \frac{1}{6} \\ &= \frac{1}{3} + \frac{1}{6} + \frac{1}{6}\end{aligned}$$

enumera todas as partições válidas.

Entrada:
O arquivo de entrada é uma seqüência de até 200 conjuntos de dados seguidos de um terminador. Um conjunto de dados é uma linha contendo 4 inteiros positivos ***p, q, a*** e ***n.*** Que satisfazem a **p, q** <= 800, **a** <=12000 e **n**<=7. Os inteiros são separados por um espaço. O terminador ou finalizador de arquivo é dado por uma única linha contendo quatro zeros separados por um espaço. Esta não é parte das entradas, mas apenas marca o fim de arquivo de entrada.

Saída:
O arquivo de saída deve ser composto de linhas cada qual contendo um único inteiro. Nenhum outro caractere deve aparecer na saída. O inteiro de saída corresponde a um conjunto de dados *p, q, a, n* e deve ser o número de todas as partições de p/q em até ***n*** frações unitárias de forma que o produto dos denominadores das frações unitárias seja menor ou igual que ***a***.

Exemplo de Arquivo de Entrada:

```
2 3 120 3
2 3 300 3
2 3 299 3
2 3 12 3
2 3 12000 7
54 795 12000 7
2 3 300 1
2 1 200 5
2 4 54 2
0 0 0 0
```

Exemplo de Arquivo de Saída para esta Entrada:

```
4
7
6
2
42
1
0
9
3
```

Problem G: Cleaning Robot

(Este problema propositalmente não foi traduzido. Contudo, seu propósito e solução é quase igual ao primeiro programa do warmup)

Here, we want to solve path planning for a mobile robot cleaning a rectangular room floor with furniture.

Consider the room floor paved with square tiles whose size fits the cleaning robot (1 × 1). There are 'clean tiles' and 'dirty tiles', and the robot can change a 'dirty tile' to a 'clean tile' by visiting the tile. Also there may be some obstacles (furniture) whose size fits a tile in the room. If there is an obstacle on a tile, the robot cannot visit it. The robot moves to an adjacent tile with one move. The tile onto which the robot moves must be one of four tiles (i.e., east, west, north or south) adjacent to the tile where the robot is present. The robot may visit a tile twice or more.

Your task is to write a program which computes the minimum number of moves for the robot to change all 'dirty tiles' to 'clean tiles', if ever possible.

Input

The input consists of multiple maps, each representing the size and arrangement of the room. A map is given in the following format.
w h
c11 c12 c13 ... c1w
c21 c22 c23 ... c2w
...
ch1 ch2 ch3 ... chw

The integers w and h are the lengths of the two sides of the floor of the room in terms of widths of floor tiles. w and h are less than or equal to 20. The character cyx represents what is initially on the tile with coordinates (x, y) as follows.

'.' : a clean tile (ladrilho limpo)
'*' : a dirty tile (ladrilho sujo)
'x' : a piece of furniture (obstacle) (mobília)
'o' : the robot (initial position) (posição inicial do robo)

In the map the number of 'dirty tiles' does not exceed 10. There is only one 'robot'.

The end of the input is indicated by a line containing two zeros.

Output

For each map, your program should output a line containing the minimum number of moves. If the map includes 'dirty tiles' which the robot cannot reach, your program should output -1.

Sample Input

```
7 5
.......
.o...*.
.......
.*...*.
.......
15 13
.......x.......
...o...x....*..
.......x.......
.......x.......
.......x.......
...............
xxxxx.....xxxxx
...............
.......x.......
.......x.......
.......x.......
..*....x....*..
.......x.......
10 10
..........
..o.......
..........
..........
..........
.....xxxxx
.....x....
```

```
.....x.*..
.....x....
.....x....
0 0
```

Output for the Sample Input

```
8
49
-1
```

PS: Basicamente no problema acima, o robo deve fazer uma limpeza no ambiente, contudo, efetuando o menor deslocamento possível.