Laboratorio lleno de perros

Contribución de Lautaro Lasorsa

Descripción del problema

En un congreso reciente, se usó el laboratorio de informática para realizar actividades. Había muchos perros merodeando. Los perros fueron entrando al laboratorio y obstruyendo el paso.

Más específicamente, el laboratorio puede representarse como una grilla de *NxM* baldosas. Cada baldosa puede:

- Estar libre, de modo que se puede transitar sobre ella.
- Tener un escritorio encima, de modo que no se puede transitar por allí.

El laboratorio tiene una única entrada, que puede abarcar varias baldosas aledañas a la entrada. Una baldosa libre es *accesible* si existe un camino que pasa exclusivamente por baldosas libres y que une la entrada con dicha baldosa. Es posible moverse desde una baldosa hacia otra cuando ambas comparten un lado.

Los escritorios están ubicados de forma tal que cuando no hay perros, todas las baldosas libres son accesibles. Se sabe que han ingresado P perros, todos en momentos distintos:

- Cuando un perro entra, **se recuesta sobre una baldosa libre, bloqueándola** de modo que ya no se puede transitar por allí. Como está muy cómodo allí, permanecerá indefinidamente en ese lugar.
- Los perros son ágiles y además se permiten el paso mutualmente, por lo que un perro podrá recostarse en una baldosa libre incluso si esta no es accesible para las personas.

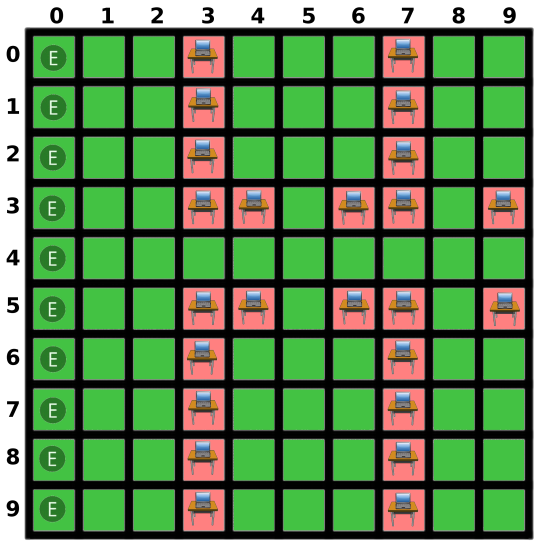
Debes escribir una función que dado el mapa del laboratorio y los ingresos perrunos, calcule cuántas baldosas accesibles había luego de entrar cada perro.

Detalles de implementación

Se debe implementar la funcion `laboratorio(mapa, perrosF, perrosC)`. Sus parámetros son:

- `mapa` es un arreglo de N cadenas de caracteres de longitud M, tal que el caracter en la posición `mapa[i][j]` describe la baldosa en fila i y columna j ($0 \leq i \leq N-1$, $0 \leq j \leq M-1$). Su valor será :
  - `.` si es una baldosa libre.
  - `#` si hay un escritorio.
  - `E` si es una baldosa aledaña a la entrada. Todas las baldosas `E` están en el borde del laboratorio, y son vecinas entre sí. Las baldosas `E` son, a su vez, baldosas libres.
- `perrosF`, `perrosC` son arreglos de P enteros, de modo que `perrosF[i]` y `perrosC[i]` indican respectivamente la fila y columna en la cual se ubicará el i-ésimo perro en orden de ingreso ($0 \leq i \leq P-1$).

La función debe retornar un arreglo de P enteros, que contenga las cantidades de baldosas libres accesibles luego de que ingresara cada perro, en orden de ingreso.



Evaluador local

El evaluador local lee de la entrada estándar:

- Una línea con los enteros N, M y P
- N líneas de M caracteres cada una, sin espacios, que describen el mapa del laboratorio.
- Luego P líneas, cada una con dos enteros. La i-ésima de estas líneas (contando desde 0) contiene los enteros `perrosF[i]` y `perrosC[i]`.

Escribe a la salida estándar una línea el arreglo retornado por la función `laboratorio`.

Cotas

- $1 \leq N \times M \leq 200.000$
- $1 \leq P \leq 100.000$
- $0 \leq perrosF[i] \leq N - 1$
- $0 \leq perrosC[i] \leq M - 1$

Ejemplos

Si el evaluador local recibe la siguiente entrada:

```
4 4 5
...#
#...
E..#
E..#
1 3
0 0
0 1
2 0
3 0
```

Una implementación correcta deberá devolver:

```
11 10 9 8 0
```

Si en cambio recibe:

```
1 10 5
#....E...#
0 2
0 1
0 5
0 6
0 7
```

Se debe devolver:

```
6 6 0 0 0
```

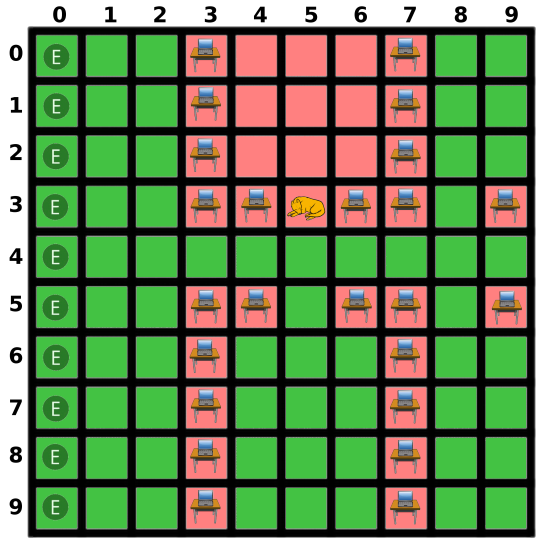
Y si en cambio se recibe :

```
10 10 6
E..#...#..
E..#...#..
E..#...#..
E..##.##.#
E.........
E..##.##.#
E..#...#..
E..#...#..
E..#...#..
E..#...#..
3 5
5 5
3 8
5 8
4 7
4 3
```

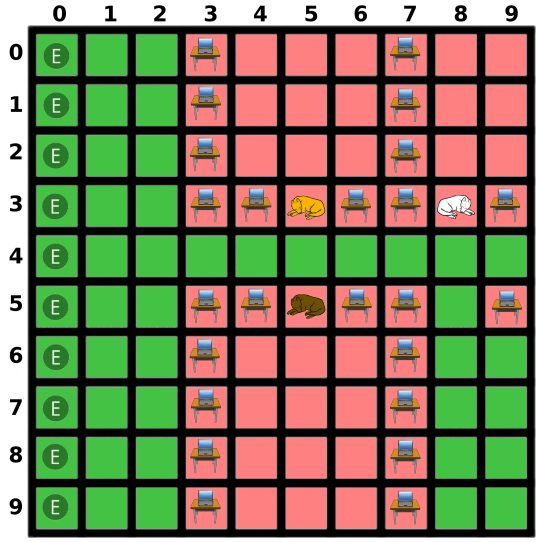
Se debe devolver:

```
66 53 46 37 34 30
```

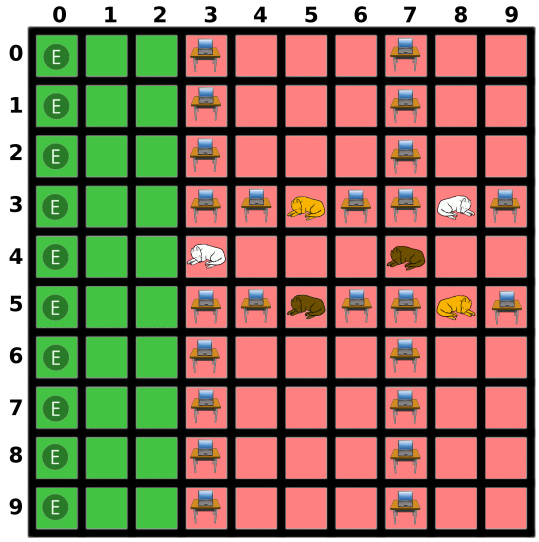
A continuación se muestra la situación luego de ingresado el primer perro:



Luego de ingresar el tercer perro quedaría:



Y la situación final es:



Subtareas

1. $N = 1$, y hay **exactamente una** baldosa aledaña a la entrada (8 puntos)
2. $N = 1$ (12 puntos)
3. $P = 1$ (15 puntos)
4. $N \times M \times P \leq 100.000$ (25 puntos)
5. Sin más restricciones (40 puntos)