Criminales vs Oficiales

1. Descripción del problema

Ovidio es un criminal buscado, y N policías están dispuestos a atraparlo. Los oficiales quieren capturar a Ovidio sin importar el costo, y Ovidio también quiere eliminar a la mayor cantidad de oficiales posible (preferentemente a todos) antes de ser atrapZaado (o antes de escapar después de eliminarlos a todos). Ni los oficiales ni Ovidio se retirarán antes de lograr sus objetivos.

Ovidio y los oficiales están en una cuadrícula unidimensional con coordenadas que van desde -10^{10} hasta 10^{10}. Ovidio está inicialmente en la coordenada C, y el i-ésimo oficial está inicialmente en la coordenada P_i. Los oficiales y Ovidio luego se turnan para moverse, siendo el equipo de oficiales el que se mueve primero.

Durante su turno, los oficiales tendrán que moverse a una celda adyacente desocupada uno por uno, en el orden que prefieran. Cada oficial tendrá que moverse. En cada momento del tiempo, ningún oficial puede estar en la misma celda que otro oficial, y tampoco puede estar en la misma celda que Ovidio. Si un oficial no puede moverse a una celda adyacente desocupada, será eliminado (y la celda en la que estaba se desocupa). Es importante notar que el equipo de oficiales intentará moverse de tal manera que se eliminen la menor cantidad posible de oficiales. Por ejemplo, con $P = [2, 3, 4]$ y $C = 6$, las siguientes posiciones posibles de los oficiales pueden ser $[1, 2, 3]$, $[1, 2, 5]$, $[1, 4, 5]$, o $[3, 4, 5]$. Sin embargo, con $P = [2, 3, 4]$ y $C = 5$, el único conjunto posible de las siguientes posiciones de los oficiales es $[1, 2, 3]$.

Después del turno del equipo de oficiales, Ovidio también se mueve a una celda adyacente desocupada o es atrapado si no puede encontrar ninguna celda adyacente desocupada. El proceso se repite hasta que Ovidio es atrapado o todos los oficiales son eliminados y Ovidio escapa.

Debes encontrar el número máximo de oficiales que pueden ser eliminados por Ovidio, y si Ovidio puede escapar o no.

Como recordatorio, dos celdas con coordenadas X y Y son adyacentes si y solo si $|X - Y| = 1$.

2. Solución

Para afrontar este ejercicio vamos a ir demostrando leyes que suceden en ciertas situaciones nos van servir para demostrar otras Para este ejercicio llamemos distancia a la cantidad de casillas que existen entre X y Y, siendo esto dos posiciones cualquiera del arreglo

Sea c un criminal y p un oficial:

lema 1: Si c y p se encuentran a distancia par entonces c siempre elimina a p.
Después de cada ciclo c y p se van a seguir encontrando a distancia par dado que los 2 están obligados a moverse,como la intención de c es eliminar p y la de p eliminar a c la distancia tiene que disminuir pero antes de que c y p estén adyacentes la distancia sería 2 y el arreglo sería ...__c__p__...,como en ese momento le toca moverse a p la distancia disminuye a 1 y c se moverá hacia p lo que pone a c y p en casillas adyacentes,lo que obliga en el turno de p moverse hacia un solo lado porque tiene la otra casilla adyacente ocupada por c , para que esto se repita seguidamente c tiene que volver a moverse a la casilla adyacente con c ,hasta que p llegue al borde del arreglo no tenga casillas adyacentes libres. De aquí podemos sacar la siguiente conclusión si c se dirige siempre hacia p, p pierde en la menor cantidad de pasos posibles

lema 2: Si c y p se encuentran a distancia impar entonces p siempre elimina a c. Dado que están a distancia impar después de cada ciclo también se cumple esta invariante, entonces la menor distancia entre c y p al terminar algún ciclo es 1 , donde le tocaría jugar a p y se moverá hacia la casilla adyacente a c, obligando a c a caminar en una sola dirección, repitiendo esto n veces, c quedaría entre el borde y p por lo k perdería. Podemos observar que si p se dirige siempre hacia p pierde en la menor cantidad de pasos posibles .

lema 3: Sea c y n oficiales p_i con $1 <= i < n$ situados a la derecha(sin perder generalidad) de c y cada p_i situado a distancia par de c,c elimina siempre a todos los oficiales. Como cada oficial esta a distancia par de c entonces entre ellos están a distancia impar, la menor distancia posible entre cada uno de ellos es 1 , si c en cada iteración se mueve en dirección a p_1 lo va a obligar a ir a la derecha por ley 1 , y entonces p_1 en algún momento va a ocupar la casilla izquierda de p_2 obligando a este a moverse solo a la derecha, de esta manera cada p_i va a eliminar la posibilidad de $p_i + 1$ de moverse a la izquierda, garantizando llegar a una iteración de la manera .._cp_1_p_2_p_3_p_4..._$p_n - 1$ donde va a tener que ser eliminado algún p_i por el palomar pues son n oficiales y n-1 espacios en blanco entre ellos, entonces se va a seguir cumpliendo que cada p_i está en una posición par respecto a c, cumpliendo la condición inicial, se repetirá este caso n veces hasta que se eliminen todos los p_i

lema 4: sea c y n oficiales p_i con $1 <= i < n$ situados a la derecha de c(sin perder generalidad) de los cuales n-1 van a estar a una distancia par y 1 a distancia impar y distribuidos de manera ramdon, sea ”p_j.el que se encuentra a distancia impar ,c siempre va a eliminar a todos los oficiales entre c y p_j . demostración :Si p_j es el primer oficial ,c pierde por lema 2 y se cumple la el lema 4. Si p_1 no es p_j entonces p_1 está a distancia par , así c avanzará para hacer retroceder a p_1 como en el lema 3 y cuando p_1 ocupe la casilla izquierda de p_2 ,este solo se va a poder mover hacia la derecha ,y así sucederá con p_i y p_{i+1} incluyendo p_j, quedando cp_0_p_1..._$p_{j-1}p_jp_{j+1}$....p_{n-1}_p_n,donde existen n p_{is} y n-2 espacios casillas vacias en el interior ,por tanto algún oficial tiene que ser eliminado, si se elimina p_j volvemos a la lema 3 donde c gana,por tanto hay que eliminar un p_i != p_j tal que se garantice que c elimine a la menor cantidad de oficiales Observemos que si se elimina siempre el primer oficial, después de la j-ésima iteración el primer oficial será p_j y se cumplirá la ley 2 donde p_j eliminará a c y la cantidad de oficiales eliminados sera j,sin embargo si se elimina algún oficial entre p_j y p_n , estuvieramos eliminando oficiales demás ya que de esta manera p_j nunca va a ser el más cercano a c para que se cumpla lema 2. De esta demostración podemos llegar a la conclusión de que la manera óptima para los oficiales de eliminarse ,es eliminar el más cercano a c

Observemos que cualquier secuencia de oficiales a un lado de c es una concatenación de las situaciones del lema 3 y lema 4

Lema 5: sean p_0 y p_1 dos oficiales que se encuentran a ambos lados de c y a distancia impar,c es eliminado .Esto se debe a que c por lema 2 va a estar obligado a retroceder por p_1 pero como p_0 también esta a distancia impar, esta se va a ir reduciendo hasta quedar _p_0_c_p_1_ => __p_0cp_1__dejando a sin opciones a c

Podemos llegar a la conclusión de que dada cualquier distribución en el array, y sean p_i y p_j los oficiales que están a una distancia impar de c tal que son el antecesor más cercano y el predecesor más cercano a c respectivamente, c eliminará a todos los oficiales que se encuentran entre p_i y p_j y estos son los que los eliminan a el si existen, pues aplicando el lema 4 eliminará a todos los oficiales entre c y p_j ,dado que p_j hace que c retroceda por lema 3,eliminando a todos entre c y p_i y quedando atrapado por estos por el lema 5, en caso de que no exista p_j o p_i será eliminado por el respectivo p_j o p_i que esté en el array y el borde de este.

El algoritmo sería pasar una ves por el array recolectando el predecesor y sucesor más cercano con distancia impar a c y todos los oficiales que se encuentran entre ellos 2, lo cual tendría una complejidad temporal de O(n).

```
arr1=['p','p','p','','p','','','c','','p',''] #output: 2
#         pi                         pj
arr2=['p','p','','p','','','c','','','p'] #output: 3
#     pi
arr3=['','p','','p','','','c','','','p','','p','','p'] #output: 5
```