Búsqueda del tesoro (tesoro)

Los Ositos Barrigoncitos Interdepartamentales emprendieron un viaje a la ciudad de La Paz para una gran expedición, más en específico irán en busca de un valioso tesoro ubicado en la gran isla cuadrada en medio del lago Titicaca, famosa por su peculiar forma cuadrada. Por motivos de cartografía, cada punto de la isla puede ser identificado por un par de enteros no negativos (x, y), donde x y y son las coordenadas de la isla. El tesoro está escondido alguno de estos puntos.

Para esta gran expedición cuentas con un radar que funciona de la siguiente manera: dadas las coordenadas (x, y), el radar te indica la distancia Manhattan entre el punto (x, y) y el punto donde se encuentra el tesoro. La distancia Manhattan entre dos puntos (x_1, y_1) y (x_2, y_2) es la suma de los valores absolutos de las diferencias de sus coordenadas, es decir, $|x_1 - x_2| + |y_1 - y_2|$.

La isla está compuesta por $(10^5 + 1)^2$ puntos. Es decir, cada punto de la isla tiene coordenadas (x, y) donde $0 \leq x, y \leq 10^5$. El radar puede ser usado fuera de la isla.

Confiando en tu capacidad de cálculo, trajiste un solo equipo de excavación, por lo que solo puedes cavar en un punto de la isla. Además de esto, quieres minimizar la cantidad de veces que usarás el radar. La cantidad de veces que usarás el radar, tendrá un valor r, dependiendo de cuanto valga, será calificado el éxito de tu expedición.

Descripción de la tarea

Tu programa deberá implementar la función `buscar()`, de la cual ya existe una versión a medias en los archivos tesoro.cpp y tesoro.py. Más especificamente:

- `buscar()`
  - Esta función se llama una vez por cada caso de prueba.
  - No tiene ningún parámetro de entrada.
  - Para obtener la distancia Manhattan entre el punto (x, y) y el tesoro, debes llamar a la función `usarRadar(x, y)`. Esta función ya esta implementada en el evaluador y no debes preocuparte por ella.
  - Cuando hayas encontrado el punto donde se encuentra el tesoro, debes llamar a la función `excavar(x, y)`. No es necesario que hayas llamado a `usarRadar(x, y)` antes de llamar a `excavar(x, y)`. Esta función ya está implementada en el evaluador y no debes preocuparte por ella.

- Luego de llamar a `excavar(x, y)`, la función `buscar()` debe terminar.

El resto del código necesario para evaluar tu solución ya está implementado en los archivos grader.cpp y grader.py. Tú solo debes preocuparte de implementar la función `buscar()`.

Uso del evaluador "grader"

Para realizar pruebas con el evaluador puedes usar una de las siguientes opciones:

1. Compilar el evaluador junto con tu solución usando el comando `g++ -std=gnu++17 -O2 -Wall -pipe -static -o tesoro grader.cpp tesoro.cpp` y luego ejecutar el programa `tesoro`.
2. Ejecutar los comandos `compile_cpp.sh` o `compile_py.sh` y luego ejecutar `run_cpp.sh` o `run_py.sh` respectivamente.

Entrada del evaluador

El evaluador leerá una sola línea que contiene dos enteros x y y, indicando la posición del tesoro que se quiere encontrar usando el radar la menor cantidad de veces que puedas.

Por cada caso de prueba, el evaluador llamará a la función `buscar()` una sola vez.

Salida del evaluador

Si se ha encontrado el tesoro, el evaluador imprimirá la cantidad de veces que se usó el radar y se procederá con la asignación de puntos. **Es muy importante que tu programa llame a la función `excavar(x, y)` cuando encuentres el tesoro** para que el evaluador funcione de manera correcta.

Límites

- $0 \leq x, y \leq 10^5$

Subtareas

Si se encuentra el tesoro usando el radar r veces:

1. (100% del puntaje) $0 \leq r \leq 2$.
2. (95% del puntaje) $3 \leq r \leq 4$.
3. (85% del puntaje) $5 \leq r \leq 10$.
4. (70% del puntaje) $11 \leq r \leq 18$.
5. (60% del puntaje) $19 \leq r \leq 35$.

Si se utiliza el radar más de 35 veces, el puntaje se calcula utilizando la fórmula:

$$\frac{73.26}{\ln(r + 3.49 \times 10^{-6})} + 39.39$$

.

En caso de que no se encuentre el tesoro o se excave una posición equivocada, el puntaje será 0.

El puntaje total de la tarea es el mínimo puntaje obtenido entre todos los casos de prueba.