Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

Tarea 3

IIC2133 - Estructuras de Datos y Algoritmos

Primer semestre, 2018

Entrega código: 18/06/2018
Entrega informe: No hay informe

Objetivos

- Modelar y resolver un problema de búsqueda en el espacio de estados.
- Diseñar un sistema de hashing para un problema específico.

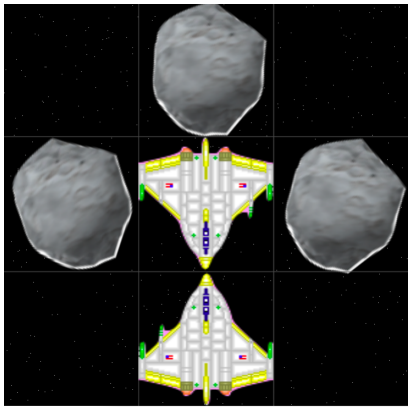
Introducción

Una flota de naves del planeta Espacial ha sido rodeado por naves del planeta Asteroidius mediante una emboscada. La tecnología Asteroidiana permite convertir en piedra a sus objetivos, pero deben mantenerse quietos cerca de su objetivo por un rato. Basta con una nave para petrificar la flota entera, pero para aumentar la probabilidad de éxito suelen enviar varias. Mientras haya una nave Asteroidiana en pie, las naves Espaciales no podrán moverse, y su sistema de direccionamiento se verá completamente restringido.

La flota Espacial debe destruir a sus enemigos en el menor tiempo posible, o será petrificada.

Problema

La flota Espacial está acostumbrada a este tipo de ataques, por lo que tienen preparado un sistema de defensa para impedir que las naves enemigas cumplan con su cometido. Este consiste en enviar la información del campo de batalla a un comando central, el cual se encargará de responder con el plan de acción.



Representación del problema. Se puede ver que las naves enemigas tienen, coincidentalmente, forma de asteroides

La información enviada incluye la ubicación de todas las naves, amigas y enemigas, en el campo de batalla. El comando central debe computar secuencia de acciones que se deben realizar para destruir a las naves enemigas en el menor tiempo posible.

El problema que tienen las naves Espaciales es que solo son capaces de dispararles a un enemigo si es que lo pueden ver, es decir, si es que la nave está apuntando directamente hacia el enemigo. En caso de que una nave no esté mirando a su objetivo, las naves Espaciales tienen cargas gravitacionales que pueden disparar para hacer girar otra nave Espacial, incluso bajo la influencia de las naves Asteroidianas.

Deberás escribir un programa en C, que dado la información enviada por la flota de naves, encuentre la mínima cantidad de acciones a realizar para poder destruir todas las naves enemigas. Para esto se espera que utilices BFS sobre el espacio de estados. Considera que solo una acción puede ser realizada en un momento dado.

Acciones

Para simplificar la modelación del problema puedes considerar que para cada estado solo existe una operación a realizar: Disparar.

Disparar(x,y): Significa que la nave posicionada en x,y dispara. Este disparo irá dirigido en la dirección que se encuentra apuntando la nave, y chocará con el primer obstáculo que encuentre. El obstáculo puede ser una nave aliada o enemiga. Si es que el obstáculo es un enemigo, este se destruirá. Si es que el obstáculo es una nave aliada, esta girará 90° hacia la **derecha**. Una nave puede estar mirando en cualquiera de las 4 direcciones: arriba, abajo, derecha o izquierda

Input

Como input recibirás un archivo de texto con las siguientes características:

- La primera línea indica las dimensiones del plano: Y X.
- Las siguientes Y líneas contienen X valores. Los numeros en las posiciones x_i, y_j puede tomar los siguientes valores:
  - 0: indica que en la coordenada x_i, y_j hay una nave apuntando para arriba.
  - 1: indica que en la coordenada x_i, y_j hay una nave apuntando para la derecha.
  - 2: indica que en la coordenada x_i, y_j hay una nave apuntando para abajo.
  - 3: indica que en la coordenada x_i, y_j hay una nave apuntando para la izquierda.
  - 4: indica que en la coordenada x_i, y_j hay un enemigo.
  - 5: indica que en la coordenada x_i, y_j no hay nada.

El input del ejemplo anterior seria el siguiente:

```
3 3
5 4 5
4 2 4
5 0 5
```

Output

Debes imprimir en consola los comandos a ejecutar en orden. No debes imprimir **Disparar**(x,y), sólo debes x,y.

El output esperado para el ejemplo anterior es:

1,2
1,1
1,2
1,1
1,2
1,1

Interfaz

Para probar que tu output es correcto se preparó una interfaz gráfica a la que puedes acceder con el siguiente link: https://iic2133-puc.github.io/

Informe y análisis

¡Esta tarea no tiene informe!

Evaluación

Se probará tu programa con distintos test. Se evaluará que la matriz obtenida luego de ejecutar las operaciones del output no contenga ninguna nave enemiga. Además se evaluará que se obtenga el resultado en la cantidad mínima de pasos. Tu programa deberá ser capaz de obtener el output los tests dentro de 10 segundos. Pasado ese tiempo el programa será terminado y se asignarán 0 puntos en ese test.

Entrega

Deberás entregar tu tarea en el repositorio que se te será asignado; asegúrate de seguir la estructura inicial de éste.

Se espera que tu código compile con `make` dentro de la carpeta **Programa** y genere un ejecutable de nombre `solver` en esa misma carpeta. **No modifiques código fuera de la carpeta *src/solver*.**
Se recogerá el estado de la rama `master` de tu repositorio, 1 minuto pasadas las 23:59 horas del día de entrega. Recuerda dejar ahí la versión final de tu tarea. No se permitirá entregar tareas atrasadas.

Bonus

Manejo de memoria perfecto (+5 % a la nota de *Código*)

Se aplicará este bonus si **valgrind** reporta en tu código 0 leaks y 0 errores de memoria, considerando que tu programa haga lo que tiene que hacer.