Departamento de Ingeniería Informática
Universidad de Santiago de Chile

VENCIENDO A KINGPIN: LABORATORIO MÉTODOS DE PROGRAMACIÓN

AUTOR(ES):
Bastián Escribano
Diego Riquelme

PROFESOR LABORATORIO:
Pablo Román

Santiago, Chile - Otoño 2021

Tabla de Contenidos

Introducción

En la siguiente evaluación, se pondrá a prueba las capacidades de análisis, investigación y además del desarrollo de resolución de problemas, todo esto llevado a un caso hipotético propuesto por la coordinación del laboratorio de métodos de programación.

El proyecto, está basado en la simulación de un juego implementado en C, donde se espera la interacción por parte del usuario mediante la consola y además archivos de texto con sus respectivas instrucciones, con el fin de poder entregar entradas, y este pueda obtener salidas que le permitan obtener respuesta al problema inicial.

Este proyecto, busca el desarrollo en forma práctica las diversas técnicas propuestas durante el curso, con el fin de trabajar estas mismas en un problema real y tangible como estudiantes del Departamento de Ingeniería Informática de la Universidad de Santiago de Chile y futuros ingenieros informáticos.

A modo de presentación del problema a resolver, se nos posiciona en la siguiente situación:

En la ciudad de Nueva York, han sucedido olas de ataques armados a los ciudadanos, por parte de pandilleros de la ciudad, la policía tras una extensa investigación, da con un principal sospechoso, Wilson Grant Fisk, empresario, dueño de las empresas Fisk, tras encontrarse con esta sorpresa, designa a Spiderman este trabajo con el fin de no ser descubiertos, y este a la vez, designa parte del trabajo a los estudiantes de ingeniería informática de la universidad de Santiago, además, se les pide a los alumnos que el problema sea escalable a cualquier ciudad donde el input del problema tenga el mismo formato.

Marco teórico

Como objetivo general, se busca la solución del problema de manera global y de forma satisfactoria, a partir de herramientas revisadas en el curso o investigadas por cuenta propia.

Para poder adentrarse al problema, se pueden definir cuáles serán los conceptos con los que se trabajarán y familiarizan con ellos en el siguiente informe, de esta manera, se pondrá en contexto las herramientas necesarias para resolver el problema.

La **recursividad**, es un concepto matemático que si bien, es una idea medianamente difícil de entender, en esta ocasión, la revisaremos dentro del campo de la informática, y esta, a la vez no es más que la propia llamada a una misma función. ¿Qué quiere decir esto?
Como dijo Einstein “locura es hacer lo **mismo** una y otra vez esperando obtener **resultados diferentes**”. Y es así como se atacará este problema y se obtendrá la solución. Hacer lo mismo varias veces de diferentes maneras, hasta obtener un resultado diferente al inicio o al esperado, que en este caso se trataría de la solución.

Epp, Susanna, en su libro de matemáticas discretas con aplicaciones *[1]*, habla sobre como la recursión es una de las ideas centrales en las ciencias de la computación, y para resolver dichos problemas, la recursión busca dividirse en problemas más pequeños que el original, para que este sea más fácil de resolver en propias llamadas dentro de dicha función.

Por otro lado, para seguir ahondando en la materia, es necesario conocer parte de un concepto fundamental para la resolución de problemas de este tipo, y hablamos de la *búsqueda en espacios de estados*, que es la sucesión de “estados” que tienen la finalidad de llegar a un estado final el cual sea satisfactorio para el problema en cuestión.

Para Fernando Berzal, docente del departamento de ciencias de la computación e A.I en la universidad de Granada, y Doctor en ciencias computacionales, la búsqueda en espacios de estado es la “Resolución del problema mediante la proyección de las distintas acciones del agente.”*[2]*, siendo el agente el programa responsable de realizar dicha búsqueda.

Ahora, ya se está en condiciones para seguir en la materia y analizar el problema planteado.

1 Epp, S. S. (1996). Discrete Mathematics With Applications (2nd ed.). Brooks/Cole Pub Co.
2 Berzal, F. (s. f.). Búsqueda en Inteligencia Artificial. Universidad de Granada. Recuperado 6 de mayo de 2021, de https://elvex.ugr.es/decsai/iaio/slides/A3%20Search.pdf

Descripción del problema

Se entregará un **archivo de entrada**, estructurado como si fuese una ciudad, el cual contendrá especificaciones con respecto a la matriz con la que trabajaremos.

Dicha matriz, será representada como una de tamaño **NxN**, cada edificio de Kingpin será una celda de esta, y el número establecido en cada una de estas celdas, **será la cantidad de veces que Spiderman tendrá que revisar dicho edificio** para que evitar que una nueva horda de pandilleros tome el control nuevamente en el edificio ya revisado.

El **archivo de entrada**, se dispondrá de la siguiente forma:

```
10                  ← Tamaño de la matriz NxN
4 1 2 1 1           ← El primer número representa la cantidad de edificios
2 2 1                 que se encuentran en dicha fila, los siguientes son cuantas veces
2 1 2                 se debe revisar cada edificio.
5 1 1 1 2 1
2 1 1
3 1 2 1
1 1
3 1 2 2
4 1 3 3 2
2 2 2               ← Últimos edificios en las filas
0                   ← Desde aquí parten las columnas en la matriz
0
1-1-1 1 1 1         ← Como podemos observar, aquí los números se empiezan a separar
4-1-2 2 2 2 1 1 2 1   por un guión, el cual nos indica la cantidad de edificios
2-1-5 1 1 1 2 1 2 3 2 consecutivos en los que se debe repartir la columna para los
1-3-2 1 1 1 1 3 2     mismos, luego, los números sin guión representan la cantidad de
2-1 2 1 2             veces que se deben revisar cada uno de forma vertical.
1 1
0
0
```

Figura 1: archivo de entrada

A partir del **archivo de entrada** que se entrega, **se requiere como salida un archivo de texto plano**, el cual pueda entender Spiderman, donde se indique la forma de la ciudad, la cantidad de edificios, su ubicación y la cantidad de veces que debe revisar cada uno, respetando los parámetros anteriormente mencionados, y a partir de la entrada de texto plano, obtendremos una matriz del siguiente tipo en el archivo de salida.

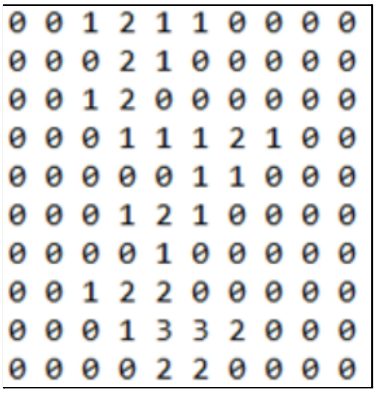

Figura 2: archivo salida 1

Luego, se debe mostrar a Spiderman, la forma en que debe recorrer la ciudad, para que pueda recorrer todos los edificios las veces necesarias.
Esto se debe implementar de igual manera a través de un archivo de texto plano y se desarrollará mediante los siguientes movimientos, los cuales se debe indicar con O,E,S,N separados por un espacio, donde cada uno será lo siguiente:

1.- O(oeste) ← **Spiderman se mueve a la izquierda**
2.- E(este) ← **Spiderman se mueve a la derecha**
3.- N(norte) ← **Spiderman se mueve arriba**
4.- S(sur) ← **Spiderman se mueve hacia abajo**

Este proceso, debe generar una salida en un archivo de texto plano “segundaSalida.out”, el cual debe tener la siguiente estructura:

E E S E N E O S S E O S O O O E S E S E E O S S S O O E O E S E N N S ← Linea de salida

Además, debe tener la funcionalidad de que la ciudad se pueda recorrer desde cualquier esquina del mapa, y entregue un output correspondiente.

Por último, no siempre se dispondrá del archivo de entrada *(ver fig 1)*, por lo que, se hará entrega en dicho caso de un archivo de salida *(ver fig 2)*, el cual, además, tendrá en la primera línea, el tamaño de la matriz NxN, a modo de ejemplo, se encontraría la siguiente matriz:

10 ← Tamaño de la matriz NxN

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

← Mapa de la matriz NxN

Descripción de la solución

Es importante establecer una **hipótesis del algoritmo** y ver si ésta cumple efectivamente con todos los requerimientos especificados en la descripción del problema.

En base y a la estructura del archivo de entrada, se podrá desde ya determinar el tamaño de la matriz, la cantidad de edificios que irán en esa columna y la cantidad de veces que debe pasar Spiderman por ese edificio**[i][j]**, refiriéndose a **[i][j]** como la posición del edificio.

Ahora para finalizar la misión, corresponde encontrar la posición para cada edificio dentro de la matriz para así indicar a Spiderman a través de puntos cardinales el siguiente movimiento para que la misión sea exitosa.

Analizando el archivo de entrada *(figura 1)* además de todo lo anteriormente mencionado, se indica el orden que deberán llevar los edificios de forma correlativa, es decir, una vez ya declarada la cantidad de edificios, el orden de la fila *(figura 1, línea 2)* estará dado de forma correlativa *(figura 3 matriz[0][2] hasta matriz[0][5]).*

		Columnas									
		1	2	3	4	5	6	7	8	9	10
Filas	1			1	2	1	1				
	2				2	1					
	3			1	2						
	4				1	1	1	2	1		
	5						1	1			
	6				1	2	1				
	7					1					
	8			1	2	2					
	9				1	3	3	2			
	10					2	2				

Figura 2: representación de la matriz

Al observar, es fácil darse cuenta que existen muchísimas combinaciones posibles, de hecho podrían existir varios caminos, o quizás no exista solución ni forma para lograr que la misión tenga éxito.

La forma de establecer un camino entre edificios es algo difícil de comprender, de hecho, este problema no se aleja mucho de un laberinto, sin embargo, se sabe que nuestro cliente es una prioridad muy importante, por lo tanto se debe cumplir con todos los requerimientos y utilizar la menor cantidad de tiempo posible.

Para esto, la misión es encontrar un algoritmo que cumpla con esta expectativa o que esté lo más factiblemente cerca.
La interpretación de un algoritmo así, no puede ser otorgado en una tercera persona, no puede ser determinado de forma omnipresente, sería ilógico determinar dónde puedo ir sin conocer el camino ni mucho menos sin estar ahí, es por eso que debemos pensar que nosotros somos la posición.

De esta forma, Spiderman determinará hacia dónde ir, sabemos que en un laberinto existen paredes y caminos hacia donde dirigirse, en este caso podría representarse de la siguiente manera *(figura 4).*

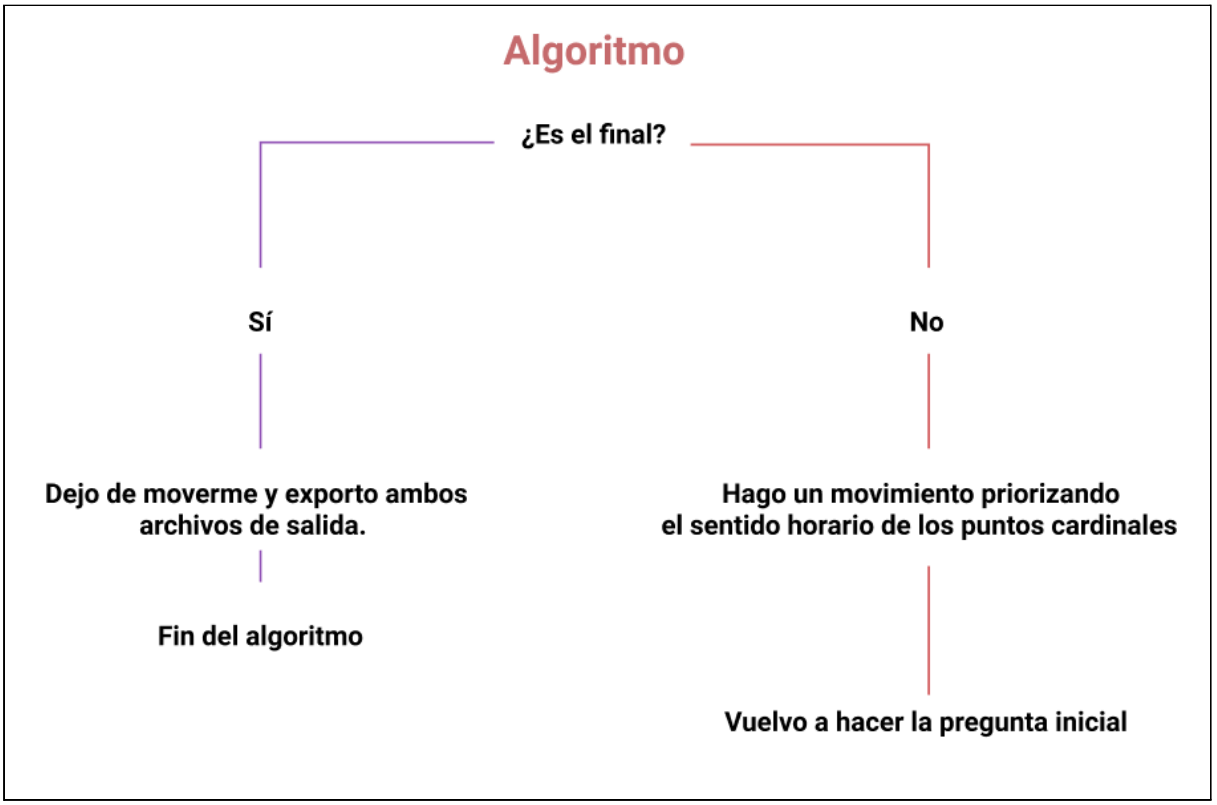

Figura 4: hipótesis de algoritmo

"Sería ilógico determinar dónde puedo ir sin conocer el camino ni mucho menos sin estar ahí". La lógica dentro de la búsqueda de un camino es utilizar todos los recursos disponibles para encontrar una solución, en la búsqueda de esta solución, existe una prioridad de avance, con lo cual podemos asumir que nuestra hipótesis se encuentra en lo correcto.

Spiderman debe moverse en sentido horario respecto a los puntos cardinales, priorizando el sentido, es decir, deberá ejercer un movimiento con prioridad **Norte-Oeste-Sur-Este** para cada nueva posición, es decir, cada vez que Spiderman ejecute un nuevo movimiento, tendrá que realizar su próximo movimiento con el mismo orden de prioridad de los puntos cardinales en sentido horario.

La implementación de este algoritmo sin duda resolvería este problema y daría con la solución, empero aún hace falta detallar qué pasaría en algunos **casos hipotéticos**

Caso 1: Existe una solución: el algoritmo encontraría una única solución.
Caso 2: Existen varias soluciones: sin embargo, solo se mostraría la solución priorizada por la búsqueda de acuerdo a los puntos cardinales.
Caso 3: No existe solución: el algoritmo no mostraría ninguna solución.

Absolutamente todos los casos son dependientes del archivo de entrada, así mismo, los archivos de salida dependen del mismo.

La construcción de un **camino gráfico** será en base a un **arreglo auxiliar de transiciones** realizadas, es decir, cada vez que se realice un movimiento, este quedará registrado, para finalmente generar una nueva matriz con este camino gráfico.

El reemplazo de la transición por un carácter perteneciente al código **ASCII** en la posición de la matriz de entrada será el camino denotado como la solución al problema.

Así mismo con los **puntos cardinales**, cada vez que se realice un movimiento, este será guardado para al final del trayecto, sea exportado en formato de archivo.

Para obtener una estructura más ordenada del algoritmo, el replanteamiento de algunos conceptos será útil para el presente lector de este informe.

El movimiento prioritario por puntos cardinales, será reemplazado por **transiciones**, es decir, existen 4 transiciones posibles en una posición, el cual será nombrado como **estado**.

Incluyendo estos conceptos al desarrollo de la solución y además en base a los requerimientos de la solución, se puede intuir que el método de solución a nuestro problema será por el algoritmo de **Búsqueda en Espacios de Estados.**

Como requerimiento además se indica que debe existir la posibilidad de poder elegir la transición a un usuario que esté ejecutando el programa.
Para cumplir esta expectativa solo basta con utilizar el algoritmo de Búsqueda en Espacios de Estados, ya que sería el mismo usuario de consola quien realizaría los movimientos.

En esta situación **sólo podrían darse los casos 1 y 2** mencionados anteriormente.

Si planteamos de manera **recursiva** esta solución al caso, se podrá llegar a una respuesta satisfactoria para todas las entradas que cumplan el formato establecido.

Conclusiones

La implementación del algoritmo y de la solución podría ser exitosa, sin embargo aún falta evaluar y rectificar algunos aspectos. El algoritmo está pensado para encontrar el camino exitoso e indicado de una forma recursiva, es decir, este algoritmo sería ejecutado de forma periódica, revisando absolutamente todos los posibles caminos, hasta que el estado inicial sea igual al estado final.

El algoritmo pretende además de resolver el problema, guardar todas las transiciones realizadas y su punto cardinal, respectivamente, para así ser exportado de forma correcta.

Que la implementación del algoritmo sea exitosa no significa necesariamente que existe un camino para lograr la captura de Kingpin, esto dependerá de la construcción de la matriz, de la cantidad de veces que se solicite pasar por cada edificio y de la forma en que están ordenados en sus filas y columnas, respectivamente.

Anexos

El formato de solución al problema está siendo desarrollado por un marco teórico, se pretende que la implementación del algoritmo sea llevada a cabo por un control de versiones **Git** para facilitar el trabajo en equipo y el control de versiones de la solución.

En este caso, se utilizará **GitHub**, específicamente el repositorio ***CGame***, por Bescri que aún debe ser actualizado, pero que se hará una vez obtenido el feedback del informe.

https://github.com/Bescri/CGame

La actualización del repositorio respecto al tiempo otorgado a este, será distribuido como el integrante lo estime conveniente, es decir, no existe un patrón de colaboración al proyecto ni orden. El trabajo será por metas que y deberán cumplirse antes de la fecha estimada.

Para la construcción de interfaces gráficas, mockups y diagramas de flujo se utilizará **Figma.**

https://www.figma.com/

El desarrollo de informes será a través de **Google Docs**.

https://docs.google.com/

Metas

Las metas serán adoptadas por los integrantes como estos lo estimen conveniente, es decir, cada integrante podrá elegir qué desarrollar, de hecho puede intervenir en el desarrollo ya en curso de otro colaborador, no existen reglas de aporte, la única regla anteriormente mencionada es cumplir todas las metas antes de la fecha estimada.