TICS200: App #3

Justo Miguel Vargas
justo.vargas@edu.uai.cl

Universidad Adolfo Ibañez — 06 de junio de 2022

Objetivos

- Comprender el paradigma de Programación Funcional.

1. El laberinto de Creta

Cuenta la leyenda que Dédalo escondió un minotauro dentro de un laberinto, en la Grecia antigua [1]. Muchos fueron deborados, mas no usted porque tiene el poder de la programación funcional!

Se le pasará un archivo que contiene el mapa, cuyo formato se especifica en la Figura 1. Las casillas están identificadas por las coordenadas que se encuentran a los lados de la grilla, mientras que el contenido de cada celda está relacionado a la presencia de un muro. Si el valor de la celda es 1, entonces ahí hay un muro. Si el valor de la celda es 0, entonces ahí no hay un muro.

	0	1	2	3	4
0	1	0	1	0	1
1	1	0	1	0	1
2	0	0	0	0	1
3	1	0	1	0	0
4	1	0	0	0	1

Figura 1: Ejemplo de laberinto

1.1. Tarea

La tarea consiste en, dado un laberinto de mxn, se debe generar un archivo con una lista de todos los únicos caminos desde el punto de partida hasta el punto de meta.

El laberinto será ingresado a través de un archivo con el siguiente formato:

[1] https://es.wikipedia.org/wiki/Laberinto_de_Creta

input.txt

```
1 0 1 0 1
1 0 1 0 1
0 0 0 0 1
1 0 1 0 0
1 0 0 0 1
```

1.2. Reglas

- Un camino no puede pasar a través de un bloque muro.
- Ninguna celda debe ser visitada más de una vez.
- Un camino debe comenzar desde el punto de partida dado hasta el punto de meta.
- La lista de caminos que deben retornar debe ser única, cada camino debe ser mencionado solamente una vez.
- El archivo de entrada será válido, no será necesario comprobar si existe punto de partida o existe punto de meta. A su vez, siempre habrá por lo menos un camino desde el punto de partida hasta el punto de meta.
- Los puntos de partida y meta corresponden a bloques libres ubicados en el extremo izquierdo y derecho, respectivamente. Siempre habrá solamente un punto de partida y un punto de meta.
- Los puntos de partida y meta no son los mismos.

1.3. Ejemplo de salida

A continuación se presenta un ejemplo de salida para el problema que se ve en la Figura 1. La salida es un archivo que contiene una lista donde cada elemento es otra lista con las coordenadas de cada camino posible entre el punto de partida y el punto de meta.

output.txt

```
[[[2,0],[2,1],[2,2],[2,3],[3,3],[3,4]],
    ↪ [[2,0],[2,1],[3,1],[4,1],[4,2],[4,3],[3,3],[3,4]]]
```

1.4. Restricciones y comentarios adicionales

- El archivo de input estará bien formateado
- Debe utilizar programación funcional. No se aceptan loops.
- Se debe programar en Python o JavaScript

1.5. HINTS

- Puede crear más de una función y utilizarlas conjuntamente
- ¿Es un movimiento válido? ¿Qué necesitas para esto? la posición actual en el laberinto, el camino actual (para evitar visitar un bloque dos veces). Cuidado con los límites.
- Si un movimiento es válido, entonces debe probarlo actualizando el camino.

2. Sobre la entrega

- Se debe trabajar en los grupos de proyectos conformados al principio del semestre
- La tarea se debe entregar el día **25 de Junio** a las 23:59.
- Por cada día de atraso se descontará 1 punto. Por ejemplo si entrega la tarea a las 00:00 del 26 de Junio la nota máxima que puede obtener es un 6.0
- La entrega se debe realizar por medio de github usando la tecnica del fork.
- La entrega sera en el repo creado con el numero de grupo asignado

[2] `http://webcursos.uai.cl`