TICS200: App #3

Universidad Adolfo Ibáñez

Profesor: Loreto Arriagada - loreto.arriagada.v@edu.uai.cl
Ayudante: Joaquin Leal - joleal@alumnos.uai.cl
15 de Mayo de 2023

Objetivos

- Aprender a estructurar un programa con paradigma **orientado funcional** en el lenguaje **Clojure** o **Python**.
- Desarrollar un entregable funcional utilizando la herramienta **replit**.

1. El Laberinto de Creta

Cuenta la antigua leyenda Greca que Dédalo, famoso arquitecto, escondió un minotauro dentro de un laberinto y todo aquel que entraba sin transitarlo por un recorrido válido, era devorado, más no Usted porque tiene el poder de la programación funcional con el que se pueden **crear n funciones y usarlas conjuntamente**.

Se adjunta un archivo de ejemplo **input.txt** que contiene un mapa del laberinto, cuyo formato se especifica en la Figura 1. Las casillas están identificadas por las coordenadas de fila y columna que se encuentran a los lados de la grilla, mientras que el contenido de cada celda está relacionado a la presencia o ausencia de un muro. Si el valor de la celda es 1, entonces ahí hay un muro y si el valor de la celda es 0, entonces ahí NO hay un muro. Note que los bordes del laberinto marcado en rojo también son muros. La casilla marcada en amarillo es la entrada o comienzo del recorrido y la casilla marcada en verde es la salida o final del recorrido.

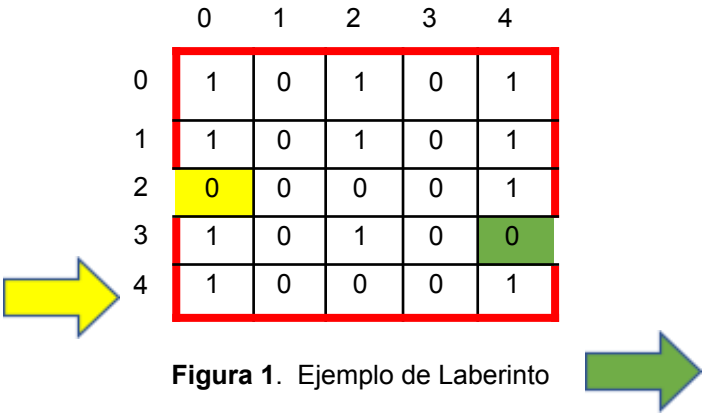

Figura 1. Ejemplo de Laberinto

1.1. Desafío

El desafío consiste leer un laberinto cualquiera de **n x m** en el formato del archivo de ejemplo, y a partir de este generar todas las posibles soluciones de recorrido válidas, sin repetir ninguna solución. Considere lo siguiente:

- Una solución de recorrido válida es aquella que comienza en la entrada, finaliza en la salida y no pasa más de una vez por cada casilla.
- Considere que el archivo de entrada siempre es válido, por lo que no se requieren reglas de validación para este.
- No obstante lo anterior, considere que en la columna 0 siempre debe haber solo una entrada que puede estar en cualquier fila y que en la última columna (4 para el archivo de ejemplo) siempre debe haber solo una salida que también puede estar en cualquier fila.

1.2. Archivo de salida

El archivo de salida contiene una lista donde cada elemento es otra lista con las coordenadas (fila, columna) de forma secuencial para cada recorrido válido. Como el mapa para el archivo de ejemplo que se presenta en la Figura 1 tiene 2 soluciones, el archivo de salida de ejemplo **output.txt** que se adjunta, contiene la siguiente información:

```
[[[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]]]
```

Figura 2. Ejemplo archivo de salida

1.3. Sobre la entrega

- La aplicación debe ser implementada en lenguaje **Clojure** o **Python** con **Paradigma Funcional**.
- Los equipos de trabajo se mantienen.
- El plazo para entrega es el 4 de Junio a las 22:30 hrs.
- La corrección se verificará en la plataforma https://replit.com
- Se debe entregar solo un archivo llamado **main.clj** o **main.py**
- El entregable señalado, debe incluir un comentario con los integrantes del grupo y solo uno de ellos debe realizar la entrega en el buzón habilitado en la plataforma WebCursos.

1.4. Recomendaciones

- Recomendaciones
    1. Escribir primero solución procedural
    2. Pasarlo a funcional eliminado ifs y fors. Para la bifurcación de la función recursiva usar if lineal
- Principal (Main)
    1. Cargar archivo input con numpy (si selecciona Python) para crear una matriz
    2. Capturar coordenadas de entrada y salida
    3. Crear arreglo de coordenadas con posibles caminos (casillas == 0)
    4. Crear arreglos vacíos con recorrido y soluciones
    5. Hacer llamada a función recursiva pasando como parámetro la coordenada de entrada
    6. Vaciar arreglo soluciones a archivo output
- Función recursiva
    1. Recibe como parámetro una coordenada
    2. Agregar parámetro al recorrido (append)
    3. Evaluar solución, sino continua su recorrido recursivo

       Si parámetro es igual a la salida

       Agregar arreglo recorrido al arreglo de soluciones

       Sino

       Hacer llamada recursiva con posibles (arriba, abajo, izquierda y derecha). Para posibles: debe existir (control de bordes) y no debe estar en el arreglo de recorrido (no pasar por la misma casilla más de una vez).

       Fin si
    4. Eliminar el parámetro “coordenada” al recorrido (pop)
- No usar códigos de otros alumnos ni de chatGPT, vuestras soluciones serán comparadas contra una base de datos de soluciones para el problema del Laberinto de Creta y contra diversas soluciones que entrega chatGPT.

3. Pauta de Evaluación

La aplicación será revisada utilizando la siguiente pauta de evaluación:

Concepto de evaluación	Descripción	Puntos	Descuento
Punto base	Punto base	1.0	No aplica
Aplicación del paradigma	Aplicar correctamente el paradigma funcional El uso de ciclos fuera de list comprehension serán considerados como doble desviación	3.0	0.5 por cada desviación
Funcionalidades	Cumplimiento del requerimiento	1.5	0.5 por cada incumplimiento funcional 0.5 por cada falla de validación o caída del programa (con excepción de las indicadas)
Lógica y lenguaje	Uso adecuado del lenguaje seleccionado y lógica empleada: - Optimización códigos fuentes - Uso de comentarios - Orden del código fuente - Uso adecuado de las estructuras de datos y sentencias del lenguaje	1.5	0.3 por cada desviación