TICS200: App #3 – El Bosque de las Runas Mágicas (Paradigma Funcional)

Profesores

- María Loreto Arriagada loreto.arriagada.v@edu.uai.cl
- Paulina González paulina.gonzalez.p@edu.uai.cl
- Justo Vargas justo.vargas@edu.uai.cl

Ayudante

- Diego Duhalde dduhalde@alumnos.uai.cl

1. Objetivos

- Comprender el paradigma de Programación Funcional 2.
- Practicar herramientas de apoyo a la programación: GIT (fork, pull requests, commits balanceados), debuggers, etc.

2. Enunciado

Un mago quiere atravesar un bosque encantado lleno de runas que modifican su energía. Cada celda del bosque contiene una runa con un valor (positivo o negativo). El mago parte desde la esquina superior izquierda y quiere llegar a la esquina inferior derecha maximizando su energía restante.

2.1 Requerimientos funcionales

1. El bosque es una matriz de enteros (puede ser de tamaño NxN).
2. El mago puede tener los siguientes movimientos:
    a. A la derecha o hacia abajo.
    b. En sentido Diagonal abajo-derecha.
    c. A la izquierda, solo si no vuelve a una celda ya visitada.
    d. Hacia arriba, solo si no vuelve a una celda ya visitada.
3. Los movimientos diagonales consumen 2 unidades extra de energía.
4. Si se pasa por una celda con valor 0, es una trampa: pierde 3 puntos de energía adicionales.
5. En cada celda, suma (o resta) el valor de la runa a su energía. 6.
6. El mago comienza con una energía inicial (por ejemplo, 12). 7.
7. El recorrido final debe ser uno de los caminos posibles que deje al mago con la mayor energía posible al final.
8. Si en algún momento la energía es menor que 0, el camino se invalida.

2.2 Supuestos y detalles

Entrada

1. Matriz de runas: [[2, -3, 1, 0, 2, 3], [-5, 4, -2, 1, 0, -4], [1, 3, 0, -3, 2, 2], [2, -1, 4, 0, -5, 1], [0, 2, -3, 3, 4, -1], [1, 0, 2, -2, 1, 5]]
2. Energía inicial: 12

Salida esperada

1. La lista con las coordenadas del camino válido con mayor energía final.
2. La energía final.

2.3 Restricciones de implementación

1. Usar programación funcional pura (sin variables mutables, ni bucles). 2.
2. Solución basada en recursión y/o map/filter/reduce según el lenguaje. 3.
3. Lenguaje: Haskell

3. Bonus y Detalles de la Entrega

1. Agregar runas especiales: si una celda es "T" (teletransportador), puede enviar al mago a otra coordenada fija si la energía es suficiente.
2. Opción de diagonales si el mago encuentra una runa "D" (doble salto).
3. Fecha de entrega: 9 y 10 de Junio, dependiendo las indicaciones del profesor de cada sección. Hora: 23:59 hrs
4. Por cada día de atraso se descuenta 1 punto, comenzando a las 00:00 del día siguiente. Ejemplo: si entregan a las 00:00 del día siguiente, la nota máxima es 6.0.

4. Formato de Entrega (vía repositorio GitHub)

Repositorio de trabajo

1. Deberán crear un repositorio para el grupo que se llame App3 en GitHub (o el que indique el curso).
2. Asegurarse que el repositorio sea privado al grupo de trabajo.
3. En ese repositorio, agregar a todos los integrantes del grupo como colaboradores, y dar acceso a dicho repositorio al profesor y al ayudante.

Commits balanceados y Pull Request

1. Cada integrante del grupo debe tener aproximadamente la misma cantidad de commits.
2. Se evaluará la participación equitativa a través del historial de

commits.

3. La entrega oficial por medio de WEBC indicando la URL del repo (o como indique la asignatura).

Estructura del repositorio

- Código fuente en haskell, organizado y documentado correctamente.
- Reflexiones finales / autoevaluación:
    - ¿Qué fue lo más desafiante de implementar en paradigma funcional?
    - ¿Qué aprendizajes surgieron del proyecto?
- Explicación de uso de IA (si aplica):
    - ¿Qué tipo de ayuda proporcionó la herramienta?
    - ¿Cómo validaron o contrastaron las sugerencias?
- README explicando cómo compilar/ejecutar el programa, con info de cada integrante (nombre, correo, etc.).

5. Rúbrica de Evaluación

Criterio	Peso	Descripción
1. Funcionamiento general	30%	● El proyecto compila y se ejecuta correctamente. ● El programa cumple con los requerimientos funcionales: recibe una matriz, calcula correctamente el mejor camino respetando las reglas del juego.
2. Paradigma Funcional	30%	Se utiliza un enfoque funcional consistente: funciones puras, recursión, inmutabilidad, composición de funciones y estructuras funcionales.
3. Informe de diseño y reflexiones finales	10%	Documento claro y estructurado que explique: diseño de solución, decisiones tomadas, posibles mejoras y reflexión sobre la experiencia
4. Uso de Git (commits y pull request) / Organización del repositorio/ Presentación	20%	● Commits equilibrados entre integrantes (aporte individual visible). ● Estructura del repositorio clara, con README que indique cómo compilar/ejecutar.
5. Presentación	10%	● Presentar en clases la solución la app ● Presentar la app funcionando.

6. Ejemplo de Uso

- App3 MatrizBosqueInicial EnergiaInicial

7. Conclusión

La App #3 busca afianzar conocimientos del paradigma funcional con la incorporación de un nuevo lenguaje de programación como es Haskell. ***Aseguren un uso equilibrado de GIT para evidenciar la contribución de cada integrante.***