Informe de Diseño y Reflexiones Finales

1. Diseño de Solución

El proyecto se estructura en torno al paradigma funcional, utilizando tipos inmutables, funciones puras y recursión para explorar todos los caminos posibles que puede recorrer el mago dentro del bosque de runas. Se definieron los siguientes tipos claves:

- `Coordenada`: para representar la posición del mago en la matriz.
- `Bosque`: una matriz de enteros que representan runas.
- `Estado`: una estructura que contiene la posición actual, la energía restante y el camino recorrido.

La lógica principal se basa en:

- La función `movimientos`, que genera los movimientos posibles del mago según las reglas del juego.
- La función `explorar`, que recursivamente busca todos los caminos desde el punto inicial hasta la meta.
- La función `resolverBosque`, que selecciona el camino que deja al mago con la mayor energía final.

Para mejorar el rendimiento y evitar combinaciones excesivas, se decidió inicialmente restringir los movimientos permitidos a derecha, abajo y diagonal. Sin embargo, se mantiene la posibilidad de extenderlo con izquierda y arriba si se desea realizar una búsqueda completa.

2. Decisiones Tomadas

- Se optó por usar `foldl1`, `filter` y listas por comprensión para recorrer y procesar los caminos.
- Se agregó manejo interactivo de entradas para permitir al usuario probar el programa con cualquier matriz en formato Haskell.
- Se agregaron trazas con `Debug.Trace` para facilitar la depuración de rutas largas.
- Para mantener la inmutabilidad, se evitó cualquier uso de variables mutables o bucles.

3. Posibles Mejoras

- Implementar búsqueda más eficiente como A* o poda alfa-beta.
- Agregar teletransportadores "T" y doble salto "D" como parte del sistema de runas especiales.
- Permitir lectura desde archivo .txt para matrices más grandes.
- Visualizar el recorrido del mago gráficamente o como matriz impresa.

4. Reflexión sobre la Experiencia

Desarrollar una búsqueda de caminos bajo el paradigma funcional fue un desafío, especialmente al prescindir de estructuras mutables y bucles tradicionales. Este proyecto me permitió profundizar en el uso de la recursión de manera controlada y en el diseño de soluciones declarativas. Esta experiencia fortaleció mi comprensión de Haskell y de los principios esenciales de la programación funcional, entre ellos la inmutabilidad, la composición de funciones y la pureza.

Me di cuenta de que, aunque puede ser más difícil al principio, el enfoque funcional produce código más predecible, seguro y fácil de razonar en problemas complejos como este, ademas la escalabilidad que tiene el programa puede a futuro caclular caminos en matrices mas grandes que 7x7 que fue lo maximo que se probó, usando un documento .txt con la descripcion de la matriz.

Explicación de uso de IA

● ¿Qué tipo de ayuda proporcionó la herramienta?

Utilicé una herramienta de inteligencia artificial (ChatGPT) para asistir en la planificación de la arquitectura del programa, la organización del código en Haskell y la validación del cumplimiento del paradigma funcional. También se utilizó para generar ejemplos de entrada/salida, proponer mejoras de eficiencia y redactar secciones del informe de diseño de forma más clara y estructurada, por ultimo la utilización de esta herramienta facilito la creación de pruebas generando aleatoriamente matrices NxN y energias dadas.

● ¿Cómo validaron o contrastaron las sugerencias?

Cada sugerencia generada por la herramienta fue revisada manualmente y contrastada con el enunciado del proyecto. Además, se probó y depuró el código en el entorno de desarrollo para asegurar que cumpliera con los requisitos funcionales y las restricciones de implementación en Haskell (uso de funciones puras, recursión, sin mutabilidad).