Trabajo Práctico: Recorrido del Caballo de Ajedrez

Objetivo:

El objetivo es implementar y comparar diferentes enfoques algorítmicos para resolver el problema del **recorrido del caballo** en un tablero de ajedrez, utilizando las técnicas de **Backtracking** y **Branch & Bound (B&B)**. El recorrido debe visitar todas las casillas del tablero exactamente una vez, en lo que se conoce como el **problema del caballo**.

Descripción del Problema:

Dado un tablero de ajedrez de tamaño N×N y una posición inicial para el caballo, el objetivo es encontrar un camino en el que el caballo visite cada casilla exactamente una vez. El caballo se mueve en forma de "L", es decir, dos casillas en una dirección (vertical u horizontal) y luego una casilla en una dirección perpendicular, o una casilla en una dirección y dos casillas en la perpendicular.

Instrucciones Generales:

- El trabajo deberá ser realizado en el lenguaje de programación Python o Java.
- Se debe implementar la solución del recorrido del caballo utilizando **Backtracking**, **Branch & Bound (B&B)**.
- Cada solución deberá ser analizada en cuanto a su eficiencia y el número de nodos explorados en cada técnica.
- Probar las implementaciones con tableros de diferentes tamaños y posiciones iniciales del caballo.

Parte 1: Backtracking para el Recorrido del Caballo

Descripción:

Implementa el algoritmo de backtracking para resolver el problema del recorrido del caballo. Este algoritmo debe explorar todas las posibilidades de movimiento del caballo para encontrar una solución.

Tareas:

1. **Backtracking básico**:

- Implementa el recorrido del caballo utilizando backtracking. Debes intentar moverte a cada una de las posiciones posibles y retroceder si no es posible completar el recorrido.
- La solución debe funcionar para tableros de diferentes tamaños N× ×N.

2. **Análisis**:
    - Mide el número de nodos explorados (movimientos realizados) y el tiempo de ejecución para diferentes tamaños de tablero (por ejemplo, N=5, N=6, N=8).
    - ¿Qué sucede cuando N es muy grande? Reflexiona sobre la complejidad temporal de esta solución.

Parte 2: Optimización con Branch & Bound (B&B)

Descripción:

Optimiza la búsqueda utilizando **Branch & Bound** para reducir el número de caminos que exploras. Puedes utilizar heurísticas para guiar la búsqueda de soluciones, por ejemplo, priorizando movimientos que lleven a menos opciones en futuras jugadas (principio de Warnsdorff).

Tareas:

1. **Branch & Bound**:
    - Modifica la solución de backtracking para incorporar **B&B**, utilizando heurísticas que determinen cuáles caminos parecen más prometedores. Por ejemplo, una heurística que prefiera los movimientos que dejen la menor cantidad de opciones futuras.
    - Implementa el cálculo de cotas (bounds) que te permitan podar caminos que no puedan llevar a una solución completa.
2. **Análisis**:
    - Mide el número de nodos explorados y el tiempo de ejecución, comparándolo con el algoritmo de backtracking puro.
    - Reflexiona sobre cómo B&B afecta el rendimiento del algoritmo en tableros de tamaño creciente.

Parte 3: Comparación Experimental

Tareas:

- Realiza pruebas con tableros de tamaños N×N para valores N=5,6,8,10
- Compara los 2 enfoques (Backtracking puro, B&B) en cuanto a:
    - **Número de nodos explorados**: ¿Cuántas decisiones se realizaron antes de llegar a la solución?
    - **Tiempo de ejecución**: ¿Qué tan rápido es cada enfoque?
    - **Facilidad de implementación**: Reflexiona sobre cuál de las técnicas fue más fácil de implementar y ajustar para este problema.

Parte 4: Informe Final

Incluir:

1. **Explicación detallada de cada implementación**: Explica cómo implementaste cada técnica (Backtracking, B&B) para resolver el problema del recorrido del caballo.
2. **Comparación de resultados**: Incluye gráficos y tablas que muestren el número de nodos explorados y el tiempo de ejecución de cada técnica para cada tamaño de tablero.
3. **Reflexiones**: Explica por qué ciertas técnicas funcionaron mejor que otras. Reflexiona sobre la aplicabilidad de cada una en problemas de optimización similares.

Entregables:

1. **Código fuente** de las implementaciones.
2. **Informe en formato PDF** que incluya el análisis y las comparaciones mencionadas anteriormente.
3. **Pruebas realizadas** en instancias de tableros con diferentes tamaños y posiciones iniciales del caballo.

Criterios a evaluar:

1. Correctitud del algoritmo: debe encontrar un recorrido válido en caso de que exista.
2. Optimización: minimización de nodos explorados y tiempo de ejecución gracias a la poda mediante B&B.
3. Heurística empleada para guiar la búsqueda y la poda de ramas no prometedoras.
4. Comparación de resultados entre diferentes tamaños de tableros.

Presentación:

1. Se hará sobre una PC que muestre los resultados obtenidos. Los mismos deben ser:
    1. Una presentación visual con el recorrido
    2. O bien, una forma clara de identificar cada uno de los escaques seleccionados
2. Cada uno de los participantes tendrá un tiempo igual para dar su parte de la presentación.
3. Se realizarán preguntas generales sobre método de selección de nodos, backtracking y cualquier componente del trabajo, tanto de código como de elemento de resolución.
4. La aprobación será individual.