UNIDAD DE INGENIERÍA EN COMPUTACIÓN
Centro Central
Inteligencia Artificial
Profesor: Kenneth Obando Rodríguez

TEC | Tecnológico de Costa Rica

Trabajo Corto (%) Fecha de Entrega: 25 de marzo, 2025

Algoritmos A* y Min-max

1 A*: Peg Solitaire

Peg Solitaire es un juego de mesa en el que el objetivo es eliminar todas las piezas del tablero excepto una, mediante saltos similares a los del juego de damas. Un movimiento consiste en saltar una pieza sobre otra adyacente (horizontal o verticalmente) hacia un espacio vacío, removiendo la pieza saltada del tablero.

Tablero Inicial

El tablero clásico de Peg Solitaire tiene una configuración inicial con una pieza faltante en el centro:

```
  * * *
  * * *
* * * * *
* * o * *
* * * * *
  * * *
  * * *
```

Donde $\star$ representa una pieza y o representa un espacio vacío.

Requisitos

1. **Implementación del Algoritmo A*:**
   - Implementar el algoritmo A* utilizando una estructura de datos apropiada para la frontera (open list).
   - Diseñar una función de evaluación $f(n) = g(n) + h(n)$, donde:
     - $g(n)$ es el costo desde el estado inicial hasta el estado n.
     - $h(n)$ es una heurística que estime el costo desde el estado n hasta el estado objetivo.
   - Proveer una heurística admisible y consistente para el problema de Peg Solitaire.
2. **Representación del Tablero y Movimientos:**
   - Diseñar una estructura para representar el estado del tablero.

- Implementar una función que genere los posibles movimientos legales desde un estado dado.
- Implementar una función que aplique un movimiento y genere un nuevo estado del tablero.

3. **Función Objetivo:**
   - Definir claramente el estado objetivo (una única pieza restante en el centro del tablero).
   - Implementar una función que verifique si se ha alcanzado el estado objetivo.
4. **Salida y Resultados:**
   - Mostrar el camino desde el estado inicial hasta el estado objetivo, incluyendo los movimientos realizados.
   - Mostrar el número de movimientos realizados y el tiempo de ejecución del algoritmo.

2 Min-Max: Lines and Boxes o Timbiriche

Descripción del Problema

Dot and Boxes es un juego de mesa en el que dos jugadores se turnan para dibujar líneas entre puntos adyacentes en una cuadrícula. El objetivo es completar más cuadros que el oponente. Cuando un jugador completa un cuadro, recibe un punto y toma otro turno.

Tablero Inicial

El tablero inicial de Dot and Boxes es una cuadrícula de puntos. Aquí se muestra un ejemplo de una cuadrícula de 2x2:

```
•---•---•
|       |
•---•---•
|       |
•---•---•
```

Requisitos

1. **Implementación del Algoritmo Minimax:**
   - Implementar el algoritmo Minimax.
   - Diseñar una función de evaluación heurística que estime la ventaja de un jugador en cualquier estado del juego.
2. **Representación del Tablero y Movimientos:**
   - Diseñar una estructura para representar el estado del tablero, incluyendo líneas dibujadas y cuadros completados.
   - Implementar una función que genere los posibles movimientos legales desde un estado dado.
   - Implementar una función que aplique un movimiento y genere un nuevo estado del tablero.

3. **Función Objetivo:**
    - Definir claramente el estado objetivo (fin del juego con todos los cuadros completados).
    - Implementar una función que verifique si se ha alcanzado el estado objetivo.
4. **Salida y Resultados:**
    - Mostrar el camino desde el estado inicial hasta el estado objetivo, incluyendo los movimientos realizados.
    - Mostrar el número de movimientos realizados y el tiempo de ejecución del algoritmo.

Instrucciones

1. **Configuración del Entorno:**
    - Utilizar un lenguaje de programación como Python, JavaScript o Java.
    - Asegurarse de que el código sea modular y esté bien documentado.
    - El programa puede ejecutarse en cuaderno Jupyter o en consola
2. **Desarrollo del Código:**
    - Implementar las funciones principales: evaluación $f(n)$, generación de movimientos, aplicación de movimientos, y verificación del estado objetivo.
    - Asegurarse de que el algoritmo A* explore los estados de manera eficiente.
3. **Pruebas:**
    - Probar el algoritmo con el tablero inicial clásico y con otras configuraciones iniciales si se desea.
    - Verificar que el algoritmo encuentra una solución óptima.
    - Mostrar gráficas del desempeño del algoritmo, por ejemplo: tiempo de ejecución, tasa de éxito del algoritmo, etc.
4. **Documentación:**
    - Incluir un informe que describa la implementación, la heurística utilizada y los resultados obtenidos.
    - Explicar las decisiones de diseño y cualquier desafío encontrado durante la implementación.
5. **Video de Ejecución:**
    - Grabar un video de 5 a 10 minutos mostrando la ejecución de una prueba del algoritmo.
    - En el video, demostrar cómo el algoritmo resuelve el problema desde el estado inicial hasta el estado objetivo.

Evaluación

La práctica será evaluada según los siguientes criterios:

- **Correctitud del Algoritmo (40%)**: El algoritmo A* o Min-Max debe encontrar una solución válida.

- **Eficiencia (20%)**: La implementación debe ser eficiente en términos de tiempo y espacio.
- **Calidad de la Heurística (20%)**: La heurística debe ser admisible y consistente.
- **Documentación y Claridad (20%)**: El código debe estar bien documentado y el informe debe ser claro y completo.

3 Entregables

- La entrega se debe realizar antes de las 10:00 pm del día de la entrega en un archivo zip mediante el TecDigital, en los grupos de trabajo previamente establecidos.
- La entrega se realiza después de la hora de entrega, se le penalizará con 5 puntos porcentuales que se acumulan cada 24 horas. Por ejemplo si entrega a las 10:05 pm su evaluación tendrá una nota base de 95%, si entrega después de las 10:05 p.m. del siguiente día, su nota base será 90%, y así sucesivamente.