Proyecto 1

Árboles Locos

Inteligencia Artificial

Elaborado por:

Rodas Arango, JUAN MANUEL - 2259571

García Castañeda, ALEX – 2259517

Gómez Agudelo, JUAN SEBASTIÁN – 2259474

Henao Aricapa, STIVEN – 2259603

Docente:

Triana Madrid, JOSHUA

Sede Tuluá

Noviembre 2024

Resumen del Proyecto

Este informe describe la implementación de un sistema visual en Pygame para la simulación de algoritmos de búsqueda en un grafo. El propósito principal fue demostrar el comportamiento de diferentes algoritmos de búsqueda, como BFS, DFS, UCS, y Greedy Best-First Search, IDS, DLS en diversos escenarios. El sistema incluye elementos interactivos como animaciones y resaltado de nodos y aristas, permitiendo analizar la expansión del árbol y el proceso de búsqueda.

Detalles de Implementación

Estructura del Grafo:

- **Formato del Nodo:**

[xy, adyacentes, costos, color_borde, color_relleno]

xy: Coordenadas para la visualización.

adyacentes: Índices de nodos conectados.

costos: Pesos de las aristas.

color_borde y **color_relleno:** Representación visual.

- **Construcción de Aristas**: Se emplea una función auxiliar **build_edges** que almacena las aristas en un diccionario con clave **(n1, n2)** y valor **[coordenadas, color].**

Visualización:

- Se usaron círculos para representar nodos y líneas para aristas.
- Los colores cambian dinámicamente según el estado del nodo:
    - **Gris**: No descubierto.
    - **Rojo**: Descubierto pero no completado.
    - **Azul**: Expansión completada.
    - **Magenta**: Nodo objetivo alcanzado.

Diseño Modular:

BFS (Breadth-First Search): Cola (FIFO)

- **Por qué**: En BFS, los nodos se exploran nivel por nivel. Se expande primero el nodo más cercano al nodo inicial, y luego los más lejanos.
- **Cómo funciona**: Una cola FIFO garantiza que los nodos que se agregaron primero serán procesados antes que los más recientes, logrando que los nodos se procesen en orden de su profundidad.

DFS (Depth-First Search): Pila (LIFO)

- **Por qué**: En DFS, se exploran completamente las ramas de un grafo antes de retroceder y explorar otras.
- **Cómo funciona**: Una pila LIFO permite que el nodo más reciente añadido sea el primero en procesarse, lo cual es ideal para ir profundizando en las ramas del grafo.

UCS (Uniform Cost Search): Cola de prioridad basada en costos acumulados (Heap Min)

- **Por qué**: UCS prioriza expandir primero los nodos con el menor costo acumulado desde el nodo inicial. Esto asegura que siempre se

evalúen primero las soluciones más baratas en términos de costo.
- **Cómo funciona**: Un heap min organiza los nodos en función de su costo acumulado. El nodo con el menor costo siempre estará en la cima y será el próximo a expandirse.

Greedy Search: Cola de prioridad basada en heurística (Heap Min)

- **Por qué**: Greedy Search selecciona el nodo que parece estar más cerca del objetivo según una función heurística.
- **Cómo funciona**: El heap min utiliza la heurística como criterio de prioridad, garantizando que se expanda primero el nodo con la menor estimación de distancia al objetivo.

IDS (Iterative Deepening Search): Pila (LIFO)

- **Por qué**: IDS combina la profundidad limitada de BFS con la estrategia de DFS. En cada iteración, se realiza una búsqueda DFS con un límite de profundidad incremental.
- **Cómo funciona**: En cada iteración, se utiliza una pila para realizar la exploración en profundidad, ya que se sigue la misma estrategia de recorrer primero los nodos más profundos dentro del límite actual.

ILS (Iterative Lengthening Search): Pila (LIFO)

- **Por qué**: ILS es similar al DFS, pero adapta los límites de profundidad para explorar soluciones más largas o costosas gradualmente.
- **Cómo funciona**: La pila LIFO es utilizada porque, al igual que DFS, las ramas se exploran profundamente antes de retroceder. Esto se repite con límites progresivos.
- Función **run_algorithm** selecciona el algoritmo apropiado según su nombre.

Algoritmos y Restricciones:

- Cada algoritmo tiene un límite de expansiones ***(num_expansions)*** para analizar el comportamiento con diferentes restricciones.

Interfaz de Usuario:

- Uso de fuentes personalizadas para mostrar mensajes en pantalla.
- Ventana interactiva que responde a eventos como cerrar la simulación.

Heurística para Greedy Best-First:

- La distancia euclidiana entre el nodo actual y el objetivo se utilizó como heurística.

Condiciones de pruebas:

El árbol de prueba está basado en el tablero del ratón que este caso es representado por el nodo[0] y el queso que está representado por el nodo[11], que son start_node y final_node respectivamente.

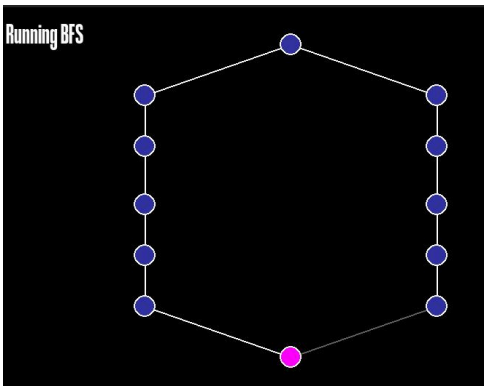
El árbol tiene ciertas modificaciones como se puede ver, hay unos nodos que tiene como costo 7 esto se hace para que se pueda ver el funcionamiento del UCS, también se configura la ubicación x y de un nodo para que se pueda ver claramente en

funcionamiento de la heurística con la distancia euclidiana.

```
graph = [
    [
        (400, 50),   # x, y position
        [1, 2],      # adjacent nodes
        [7,1]         # cost
    ],
    [
        (200, 120),  # 1
        [0, 3],
        [7,7]
    ],
    [
        (600, 130),  # 2
        [0, 4],
        [1,1]
    ],
    [
        (200, 190),  # 3
        [1, 5],
        [7,1]
    ],
    [
        (600, 190),  # 4
        [2, 6],
        [1,1]
    ],
```

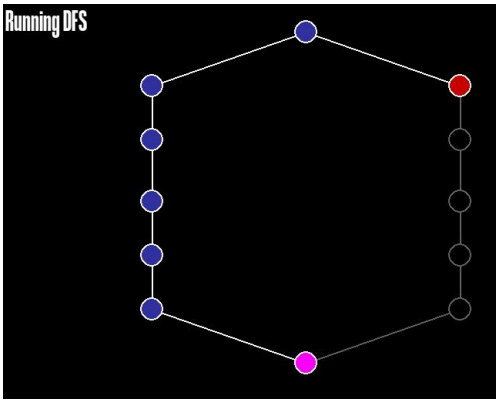
Pruebas de ejecución

BFS:



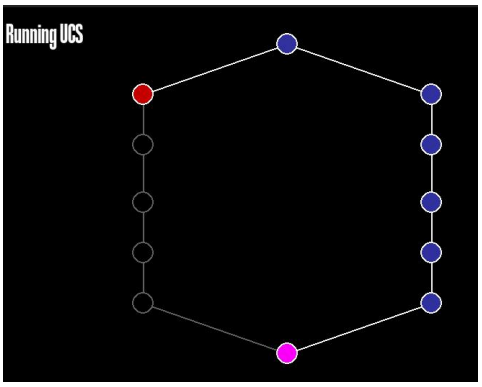
Objetivo alcanzado tras 12 expansiones

DFS:



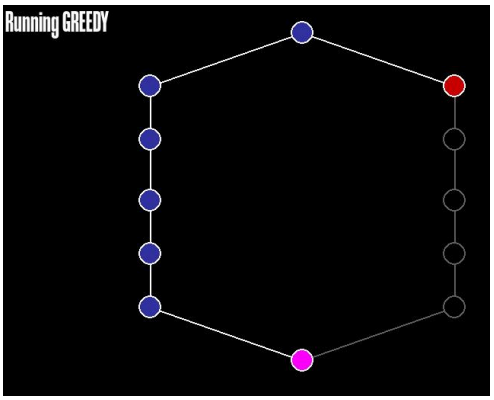
Objetivo alcanzado tras 7 expansiones

UCS:



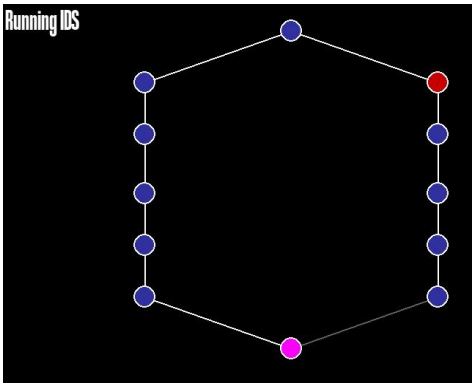
Objetivo alcanzado tras 7 expansiones (Se elige el camino derecho ya que es el más barato).

Greedy:



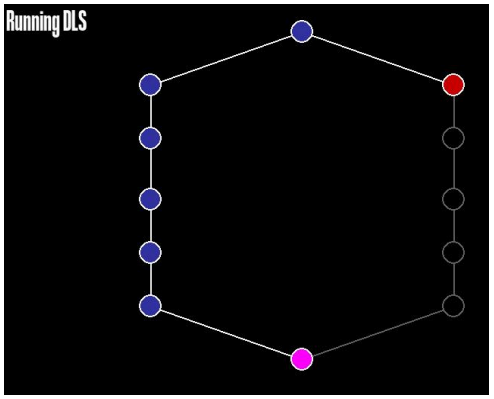
Objetivo alcanzado tras 7 expansiones (Se elige el camino izquierdo porque el nodo número 1 tiene una heurística menor a la del nodo 2) - Heurística basada en la distancia Euclidiana.

IDS:



Objetivo alcanzado en profundidad 6

DLS:



Objetivo alcanzado en profundidad 6