Laboratorio de Algoritmos: DFS, BFS y Backtracking

Julio Pérez

June 5, 2024

Objetivo

Entender y aplicar los algoritmos DFS, BFS y backtracking,

Participantes

2 ó 3 personas

Estructura del Laboratorio

1. Problema 1: Solucionador de Laberintos con DFS
2. Problema 2: Camino Más Corto en un Grafo con BFS
3. Problema 3: Problema de las N-Reinas con Backtracking
4. Problema 4: Combinando Algoritmos
5. Pruebas y Análisis

Problema 1: Solucionador de Laberintos con DFS

Descripción

Dado un laberinto representado como una cuadrícula 2D, encontrar un camino desde el inicio hasta el final usando DFS. EL número 1 representa una unidad de pared y los zeros indican caminos posibles. El laberinto está rodeado por paredes (límites de la matriz)

Entradas

Un laberinto representado como una lista 2D, y las coordenadas de inicio y fin.

Salidas

Una lista de tuplas que representan el camino desde el inicio hasta el final.

Ejemplo

```
laberinto = [
    [0, 1, 0, 0, 0],
    [0, 1, 0, 1, 0],
    [0, 0, 0, 1, 0],
    [0, 1, 1, 1, 0],
    [0, 0, 0, 0, 0]
]

inicio = (0, 0)
fin = (4, 4)
```

Salida Esperada

```
[(0, 0), (1, 0), (2, 0), (2, 1), (2, 2), (3, 2), (4,
   2), (4, 3), (4, 4)]
```

Problema 2: Camino Más Corto en un Grafo con BFS

Descripción

Dado un grafo, encontrar el camino más corto entre dos nodos usando BFS.

Entradas

Un grafo representado como una lista de adyacencia, y los nodos de inicio y fin.

Salidas

Una lista de nodos que representan el camino más corto desde el inicio hasta el final.

Ejemplo

```
grafo = {
    'A': ['B', 'C'],
    'B': ['A', 'D', 'E'],
    'C': ['A', 'F'],
    'D': ['B'],
    'E': ['B', 'F'],
    'F': ['C', 'E']
}

inicio = 'A'
fin = 'F'
```

Salida Esperada

```
['A', 'C', 'F']
```

Problema 3: Problema de las N-Reinas con Backtracking

Descripción

Colocar N reinas en un tablero de ajedrez de NxN de manera que ninguna reina amenace a otra usando backtracking.

Entradas

Un número entero n que representa el tamaño del tablero y el número de reinas.

Salidas

Una lista de listas, donde cada lista interna representa una solución posible con las posiciones de las reinas.

Ejemplo

```
n = 4
```

Salida Esperada

```
[[(0, 1), (1, 3), (2, 0), (3, 2)], [(0, 2), (1, 0),
   (2, 3), (3, 1)]]
```

Problema 4: Combinando Algoritmos

Descripción

Dado un laberinto con obstáculos móviles, encontrar el camino más corto desde el inicio hasta el final.

Entradas

- Un laberinto representado como una lista 2D.
- Las coordenadas de inicio y fin.
- Una lista de obstáculos móviles con sus rutas. Cada obstáculo se mueve una celda por cada ("segundo") iteración, y una vez alcanza el final de la lista las paredes móviles se mueven en sentido contrario, así infinitamente

Salidas

Una lista de tuplas que representan el camino más corto desde el inicio hasta el final evitando los obstáculos móviles.

Ejemplo

```
laberinto = [
    [0, 0, 0, 0, 0],
    [0, 1, 0, 1, 0],
    [0, 1, 0, 1, 0],
    [0, 0, 0, 1, 0],
    [0, 0, 0, 0, 0]
]

inicio = (0, 0)
fin = (4, 4)
obstaculos_moviles = [
    {'ruta': [(1, 1), (2, 1), (3, 1)], 'actual': (1,
        1)}
]
```

Salida Esperada

```
[(0, 0), (0, 1), (0, 2), (0, 3), (0, 4), (1, 4), (2,
   4), (3, 4), (4, 4)]
```

Pruebas y Análisis

Actividad

Probar las implementaciones con varias entradas.

Discusión

Analizar la complejidad de tiempo y espacio de cada solución.