Maximiliano Galindo email: maximilianogalindo7@gmail.com

Tarea 1 IA

de Maximiliano Galindo (Curso de Inteligencia Artificial)

Parte 2

Búsqueda en Anchura

El recorrido del reportado por el algoritmo ”Búsqueda en Anchura” fue:

```
Tarea_1/bin/python3/home/max/maestria/AI-Class-Homework/Tarea_1/BFS.py
Arad-->Sibiu-->Fagaras-->Bucharest
```

Búsqueda en Profundidad

El recorrido del reportado por el algoritmo ”Búsqueda en Profundidad” fue:

```
Tarea_1/bin/python3/home/max/maestria/AI-Class-Homework/Tarea_1/DFS.py
Arad-->Timisoara-->Lugoj-->Mehadia-->Dobreta-->Craiova-->Pitesti-->Bucharest
```

Búsqueda Iterativa

El recorrido del reportado por el algoritmo ”Búsqueda Iterativa” fue:

```
    Tarea_1  /bin/python3  /home/max/maestria/AI-Class-Homework/Tarea_1/Iterative-Search.py
Arad-->Sibiu-->Fagaras-->Bucharest
```

Búsqueda de Costo Uniforme

El recorrido del reportado por el algoritmo ”Búsqueda Uniforme” fue:

```
    Tarea_1 /bin/python3 /home/max/maestria/AI-Class-Homework/Tarea_1/Uniform-Cost-Search.p
Arad-->Sibiu-->Rimnicu Vilcea-->Pitesti-->Bucharest
Total Cost: 418
```

Búsqueda Voraz

El recorrido del reportado por el algoritmo ”Búsqueda Voraz” fue:

```
    Tarea_1 /bin/python3 /home/max/maestria/AI-Class-Homework/Tarea_1/Greedy-Search.py
Arad-->Sibiu-->Fagaras-->Bucharest
Total Cost: 450
```

Búsqueda A^*

El recorrido del reportado por el algoritmo "A^*" fue:

```
Tarea_1 /bin/python3 /home/max/maestria/AI-Class-Homework/Tarea_1/A*-Search.py
Arad-->Sibiu-->Rimnicu Vilcea-->Pitesti-->Bucharest
Total Cost: 418
```

Parte 3: Realizar en análisis del algoritmo de búsqueda voraz.

¿Es completo?

El algoritmo de búsqueda voraz no es completo. Esto significa que no garantiza encontrar la solución, ya que toma decisiones locales óptimas basadas solo en la función heuristica en cada paso sin considerar el panorama completo. En algunas situaciones, estas decisiones pueden llevar a un camino sin salida o a una solución no óptima. Por lo tanto, la completitud no está garantizada.

¿Es óptimo?

El algoritmo de búsqueda voraz no garantiza encontrar siempre la solución óptima. Debido a su naturaleza de tomar decisiones basadas en la mejor opción local de acuerdo con la función heuristica en cada paso, podría llegar a una solución que no es la más óptima en términos globales y en términos de los costos reales. En algunas ocasiones, puede tener éxito en encontrar la solución óptima.

Complejidad en tiempo:

Dado que el algoritmo de búsqueda voraz toma decisiones óptimas localmente basadas en heurísticas, es importante considerar que, en el peor de los casos, este enfoque no garantiza completitud ni optimización global. En tales situaciones, el algoritmo puede requerir explorar la totalidad de los nodos presentes en la estructura de datos, lo que lleva a una complejidad temporal expresada como $O(b^m)$. Aquí, b representa el factor de ramificación, mientras que m corresponde a la profundidad máxima alcanzada.

Es relevante señalar que el rendimiento en el peor escenario de este algoritmo de búsqueda voraz puede equipararse al peor caso de la estrategia de búsqueda en profundidad, aunque con la diferencia de estar guiado por heurísticas. Sin embargo, es crucial destacar que la eficiencia de este método puede ser significativamente mejorada mediante la aplicación de heurísticas adecuadas. Estas heurísticas permiten optimizar los tiempos de ejecución y mitigar los inconvenientes presentes en los casos menos favorables.

Complejidad en espacio:

Debido a la naturaleza secuencial en la que el algoritmo explora los nodos, la complejidad espacial en el peor escenario se establece en $O(b^m)$. No obstante, es importante notar que esta complejidad puede reducirse significativamente a través de la incorporación de heurísticas sólidas.

1 Parte 4: Problema de la mochila

El código se encuentra en el archivo llamado `Knapsack.py`

```
    Tarea_1/bin/python3 /home/max/maestria/AI-Class-Homework/Tarea_1/Mochila.py
Algoritmo usado: Busqueda Voraz
Cantidad de Objetos en la Mochila: 4
Capacidad de la mochila: 31181 g
Peso total conseguido: 30162 g
Capacidad sin utilizar: 1019 g
Mochila: [(3878, 9656 g), (4136, 10372 g), (2945, 7390 g), (1022, 2744 g)]
```

Parte 5: Problema de la mochila (Bonus)

El código se encuentra en el archivo llamado `Knapsack.py`

```
Tarea_1  /bin/python3 /home/max/maestria/AI-Class-Homework/Tarea_1/Knapsack.py

Algoritmo usado: Busqueda Voraz
Cantidad de Objetos en la Mochila: 18
Capacidad de la mochila: 1000000 g
Peso total conseguido: 1000000 g
Capacidad sin utilizar: 0
Mochila: [(92264, 83877 g), (76140, 69221 g), (109027, 99123 g), (177323, 161215 g),
(69271, 62979 g), (179749, 163423 g),
(61650, 56051 g), (108657, 98789 g),
(181206, 164754 g), (41755, 37964 g), (327, 298 g), (437, 399 g), (706, 649 g),
(244, 225 g), (945, 873 g), (87, 81 g), (74, 70 g), (8, 9 g)]
```