**CS 6364.001 - Artificial Intelligence project report**

**Minimum distance path search**

**Bohao Chen**

**UTD-ID: 2021271333**

**NET-ID: bxc151130**

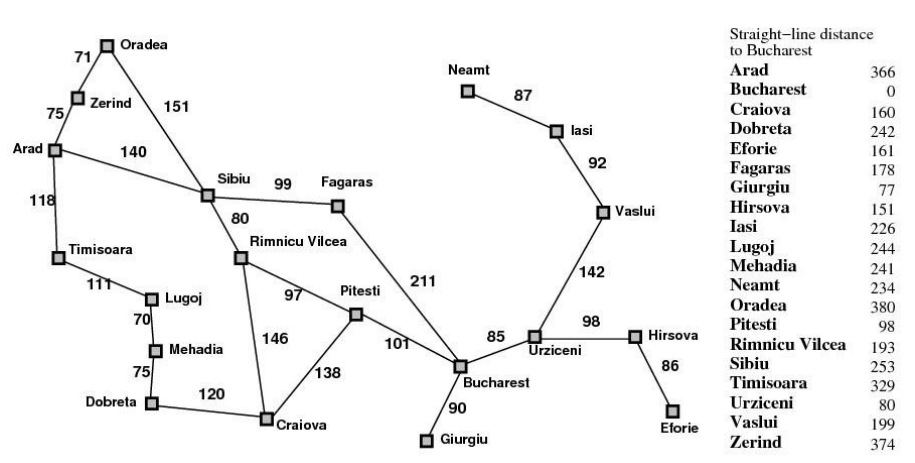
1. **Introduction**

There are several ways to find the minimum distance path between two places in a searched graph. In this project, I want to compare the performance between blind search and heuristic search, and compare the performance several search algorithm. The performance includes nodes that expands, total distance of the path, number of node visit, and cost (number of node that expand / number of node in result path).

1. **Graphs to be used**

In this project, I choose 4 graphs for comparing searching performance of each algorithms. These graphs include big size graph (number of nodes is not less than 20), middle size graph (number of nodes is between 10 and 20) and small size graph (number of nodes is less than 10). So that these graphs can simulate the real world problems, and make the result more accurate.

**First Graph: Romania** (big size graph)



This graph contains 20 nodes, which is a complex graph relatively. I choose Zerind as the start point, Bucharest as the end point. For reading convenience, I use number id 19 as start point, and number id 1 as end point.

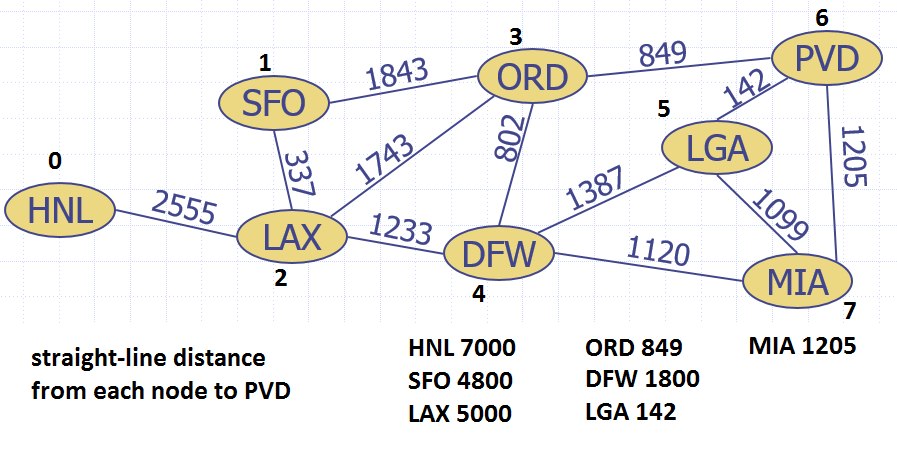
The shortest path from start point 19 to end point 1 is:

19 – 0 – 15 – 14 – 13 – 1

Zerind – Arad – Sibiu – Rimnicu Vilcea – Pitesti - Bucharest

The shortest length is 493.

**Second Graph: Airport** (small size graph)



This graph contains 8 nodes, which is a small and simple graph relatively. I choose HNL (node 0) as start point, MIA (node 7) as end point.

The shortest path from HNL to MIA is:

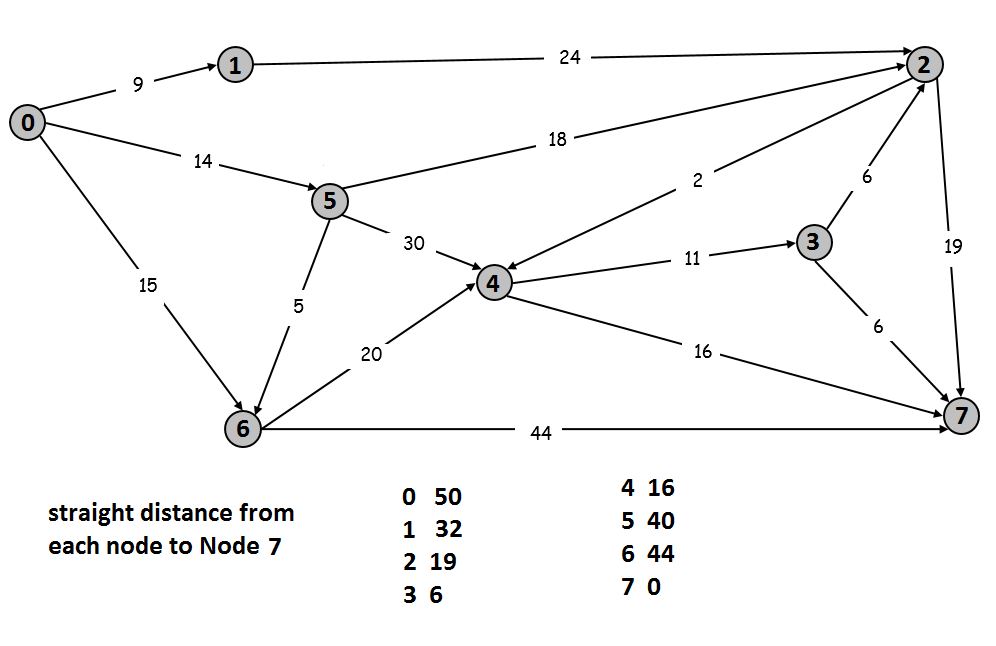
0 – 2 – 3 – 6

NHL – LAX – ORD - PVD

The shortest path length from HNL to MIA is:

5147

**Third Graph: number graph** (small graph)



This graph contains 8 nodes, which is a small and simple graph relatively. I choose node 0 as start point and node 7 as the end point.

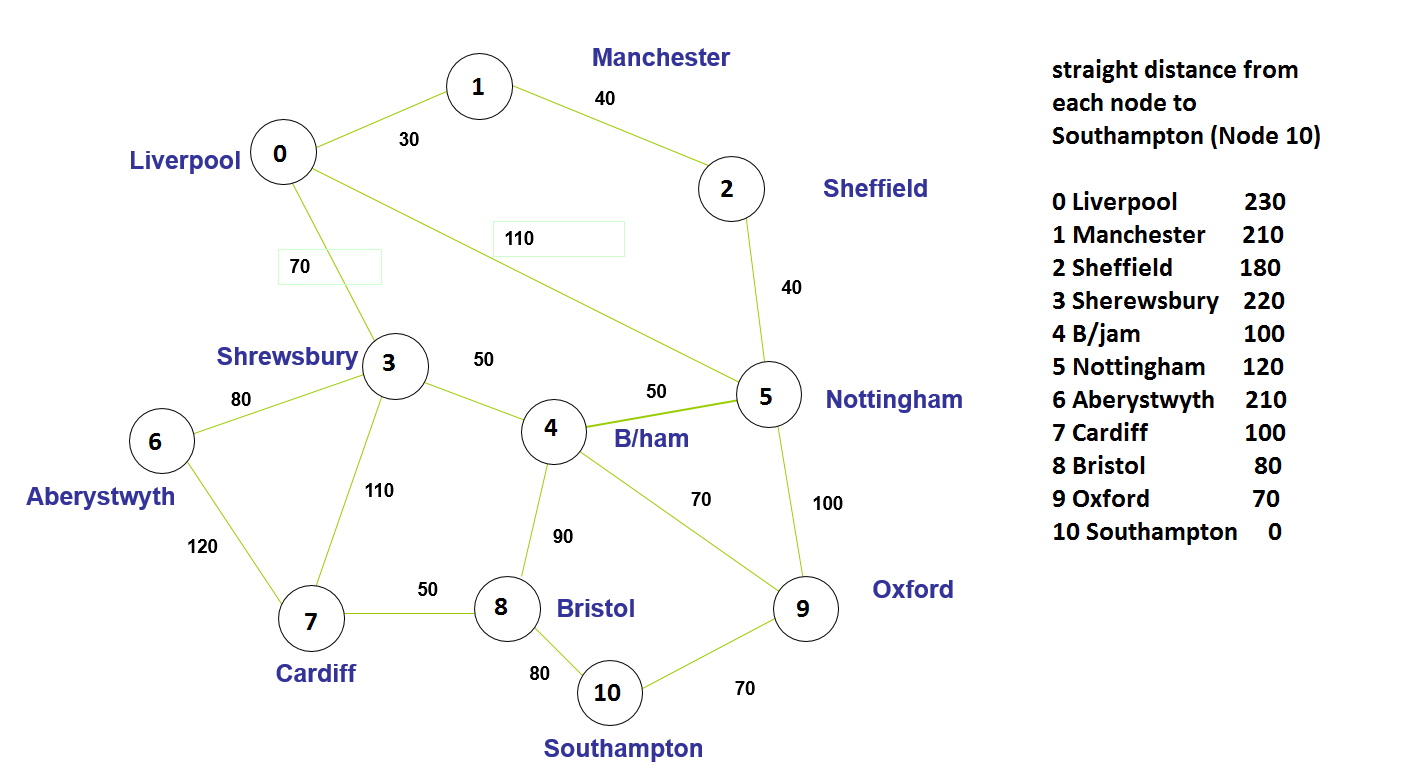
The shortest path from node 0 to node 7 is:

0 – 5 – 2 – 3 – 7

The shortest path length from node 0 to node 7 is:

45

**Fourth Graph: England** (middle size)



This graph contains 11 nodes, which is a middle size graph relatively. I choose Liverpool (node 0) as start point and Southampton (node 10) as end point.

The shortest path from Liverpool (node 0) to Southampton (node 10) is:

0 – 3 – 4 – 9 – 10

Liverpool – Sherewsbury – B/jam – Oxford - Southampton

The shortest path length from Liverpool (node 0) to Southampton (node 10) is:

260

1. **Algorithm performance**

The performance includes nodes that expands, total distance of the path, number of node visit, and cost (number of node that expand / number of node in result path).

The cost means: in order to find a node in the result path, the number of nodes that need to be expand. If the value of cost is high, it means this search algorithm has lower performance.

1. **BFS**

Breadth-first search is a blind search that search node from top level to bottom level.

The search result of BFS is shown below:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Nodes that expand | Nodes in result path | Path length | Cost | Is the shortest |
| First Graph | 14 | 5 | 525 | 2.8 | No |
| Second Graph | 9 | 4 | 5147 | 2.25 | Yes |
| Third Graph | 11 | 3 | 59 | 3.67 | No |
| Fourth Graph | 24 | 4 | 280 | 6 | No |

From the form, we find that the cost of BFS is high. Algorithm needs to pay more than 2 times effort in order to find the path.

Also, BFS only find the shortest path in small graph, and failed to find shortest path in other 3 graphs. The accuracy is low.

1. **Beam Search**

Beam Search is a heuristic search. The search result of Beam Search is shown below:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Nodes that expand | Nodes in result path | Path length | Cost | Is the shortest |
| First Graph | 9 | 5 | 525 | 1.8 | No |
| Second Graph | 6 | 4 | 5147 | 1.5 | Yes |
| Third Graph | 7 | 3 | 59 | 2.33 | No |
| Fourth Graph | 11 | 4 | 280 | 2.75 | No |

From the form, we find the cost of Beam Search is lower than BFS. Beam Search needs to pay around 2, sometimes lower than 2 times effort in order to find the path.

However, Beam Search only finds shortest path in second graph, which is a small graph, and failed to find shortest path in other 3 graphs. The accuracy is low.

1. **DFS**

Depth-first search is a blind search. The search result of DFS is shown below:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Nodes that expand | Nodes in result path | Path length | Cost | Is the shortest |
| First Graph | 12 | 5 | 525 | 2.4 | No |
| Second Graph | 5 | 5 | 6113 | 1 | No |
| Third Graph | 3 | 3 | 59 | 1 | No |
| Fourth Graph | 4 | 4 | 280 | 1 | No |

From the form, we can find the cost of DFS is low. In 3 graphs the cost is 1. It means using DFS on these 4 graphs doesn’t need to pay too much effort in order to find the path.

However, DFS can’t find shortest path for any graph, the accuracy is 0.

1. **Hill Climbing Search**

Hill Climbing Search is a heuristic search algorithm. The search result of Hill Climbing Search is shown below.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Nodes that expand | Nodes in result path | Path length | Cost | Is the shortest |
| First Graph | 5 | 5 | 525 | 1 | No |
| Second Graph | 4 | 4 | 5147 | 1 | Yes |
| Third Graph | 3 | 3 | 52 | 1 | No |
| Fourth Graph | 4 | 4 | 280 | 1 | No |

From the form, we can find the cost of Hill Climbing Search are all 1, which means the efficiency of finding path is really high. Every step to expand a node is finding the path.

However, Hill Climbing Search only finds the shortest path once. The accuracy is low.

1. **Branch and Bound Search**

Branch and Bound Search is a heuristic search algorithm. The result of Branch and Bound Search is shown below:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Nodes that expand | Nodes in result path | Path length | Cost | Is the shortest |
| First Graph | 199 | 6 | 493 | 33.17 | Yes |
| Second Graph | 76 | 4 | 5147 | 19 | Yes |
| Third Graph | 44 | 5 | 44 | 8.8 | Yes |
| Fourth Graph | 203 | 5 | 260 | 40.6 | Yes |

From the form, we find that the cost of Branch and Bound Search is so high. So many duplicated nodes have been visited again and again. In order to find one step of path, Branch and Bound Search need to pay more than 10 times effort.

However, Branch and Bound Search must return the shortest path.

1. **Redundant Path Search**

Redundant Path Search is a blind search algorithm. The result of Redundant Path Search is shown below:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Nodes that expand | Nodes in result path | Path length | Cost | Is the shortest |
| First Graph | 13 | 6 | 493 | 2.17 | Yes |
| Second Graph | 7 | 4 | 5147 | 1.75 | Yes |
| Third Graph | 7 | 5 | 45 | 1.4 | No |
| Fourth Graph | 10 | 4 | 280 | 2.5 | No |

From the form, Redundant Path Search has a not very high cost and 50% percentages of accuracy. Performance is in the middle level among these algorithms.

1. **ASTAR Algorithm**

ASTAR algorithm is a heuristic algorithm. It’s a good algorithm to find minimum path. The result form is shown below:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Nodes that expand | Nodes in result path | Path length | Cost | Is the shortest |
| First Graph | 9 | 6 | 493 | 1.5 | Yes |
| Second Graph | 2 | 2 | 5147 | 1 | Yes |
| Third Graph | 5 | 5 | 45 | 1 | No |
| Fourth Graph | 10 | 4 | 280 | 2.5 | No |

From the form, ASTAR algorithm has a little bit lower cost than Redundant Path Search, the accuracy is 50% which is not very high.

1. **Analyze**

Combine the result in part 3. I made the form to compare all algorithm together.

Focus on **Cost**:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | First Graph | Second Graph | Third Graph | Fourth Graph |
| BFS | 2.8 | 2.25 | 3.67 | 6 |
| Beam Search | 1.8 | 1.5 | 2.33 | 2.75 |
| DFS | 2.4 | 1 | 1 | 1 |
| Hill Climbing Search | 1 | 1 | 1 | 1 |
| Branch and Bound Search | 33.17 | 19 | 8.8 | 40.6 |
| Redundant Path Search | 2.17 | 1.75 | 1.4 | 2.5 |
| ASTAR Algorithm | 1.5 | 1 | 1 | 2.5 |

From above, we can find that Branch and Bound Search has the highest cost. And the cost is much higher than any other algorithms. Hill Climbing Search has the lowest cost but is not so much lower than other algorithms.

Now let’s focus on **accuracy**:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | First Graph | Second Graph | Third Graph | Fourth Graph | Number of Yes | Number of No |
| BFS | No | Yes | No | No | 1 | 3 |
| Beam Search | No | Yes | No | No | 1 | 3 |
| DFS | No | No | No | No | 0 | 4 |
| Hill Climbing Search | No | Yes | No | No | 1 | 3 |
| Branch and Bound Search | Yes | Yes | Yes | Yes | 4 | 0 |
| Redundant Path Search | Yes | Yes | No | No | 2 | 2 |
| ASTAR Algorithm | Yes | Yes | No | No | 2 | 2 |

From above, we find Branch and Bound Search has the top accuracy, which is 100%. DFS has the lowest accuracy which is 0. BFS and Beam Search have accuracies that lower than 50%. Redundant Path Search and ASTAR Algorithm have 50% accuracies.

1. **Result**

If we want to find the result that must be the shortest path, we should use Branch and Bound Search algorithm. This algorithm can return the shortest path in any graph. However, it will spend more running time and more space. The situation we choose Branch and Bound Search algorithm is only when user want to find the shortest path no matter how much time cost.

If user want to save time, Redundant Path Search and ASTAR Algorithm would be good to use. Because these algorithms don’t spend too much time and have a 50% accuracy. Hill Climbing Search is not a good choice because the running time is not save too much when we choose it, however, the accuracy is much lower than Redundant Path algorithm and ASTAR Algorithm.