Munirah Binte Mohamad

U1822418F

Lab Group: TSP2

ArtiFICIAL iNTELLIGENCE

Lab Assignment 1

# ***Question 1***

Assumptions made:

* The graph is on a plane.
* The Euclidean Distance is set as at arc cost and as the heuristic function.
* Euclidean distance (straight-line distance) as the arc cost and as the heuristic function.
* Nodes and its neighbours are being read from left to right.
* The starting node is S and the ending node is G.

1. **Give a graph where depth-first search (DFS) is much more efficient (expands fewer nodes) than breadth-first search (BFS)**

In graph 1 below, DFS expands five nodes, whereas BFS expands every node.

Diagram

Description automatically generated

Graph 1

DFS search by expanding the deepest unexpanded node which can be implemented by a Last-In-First-Out (LIFO) stack. The search will backtrack when there are no more nodes for expansion.

DFS: S → A → C → H → G

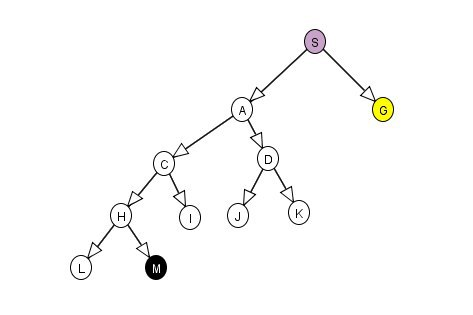
BFS search by expanding the shallowest unexpanded node which ca be implemented by a First-In-First-Out (FIFO) queue.

BFS: S → A → B → C → D → E → F → H → I → J → K → L → M → N → G

Thus, this is a graph where DFS is much more efficient than BFS.

1. **Give a graph where BFS is much better than DFS.**

In graph 2 below, DFS expands every node whereas BFS expands only three nodes.



Graph 2

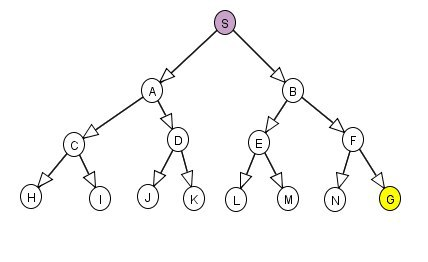
DFS: S → A → C → H → L → M → I → D → J → K → G

BFS: S → A → G

Thus, the graph above shows BFS is much better than DFS.

1. **Give a graph where A\* search is more efficient than either DFS or BFS.**

In graph 3 below, DFS and BFS would expand every node whereas A\* Search expands only three nodes.



Graph 3

A\* search operates by estimating the total cost of path through n to goal. It will calculate the closest path to the goal based on the cost of its neighbouring nodes.

BFS: S → A → B → C → D → E → F → H → I → J → K → L → M → N → G

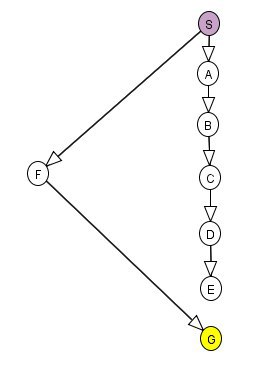
DFS: S → A → C → H → I → D → J → K → B → E → L → M → F → N → G

A\* Search: S → B → F → G

Thus, the graph shows that A\* search is more efficient than either BFS or BFS.

1. **Give a graph where DFS and BFS are much more efficient than A\* Search.**

In graph 4 below, DFS expand three nodes, BFS expand four nodes and A\* search expands every node.



Graph 4

BFS: S → F → A → G

DFS: S → F → G

A\* Search: S → A → B → C → D → E → F → G

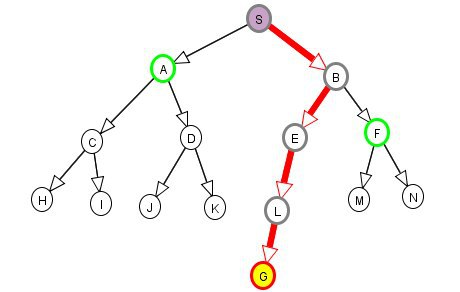
Thus, this graph shows that DFS and BFS are much more efficient than A\* Search.

# ***Question 2***

1. **What is the effect of reducing h(n) when h(n) is already an underestimate?**

When h(n) is reduced when h(n) is already an underestimate, there are two results that could take place. Firstly, the same number of nodes (optimal path) are expanded. Secondly, a graph that contains multiple paths to the goal node will produce a path that expands extra nodes. This is because different optimal paths may be found. Thus, the lower h(n) is, the mode node A\* search expands which makes it slower.

Conjecture: When h(n) is reduced when h(n) is already an underestimate, it will produce a path that expands extra nodes. To prove this, we set heuristic of S to be lower than its Euclidean distance from the goal node.



Graph 5

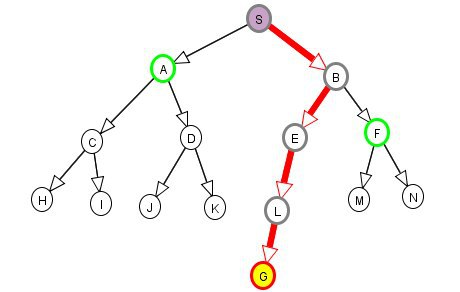
Observation: The optimal path was found without expanding unnecessary nodes (Only expanding five nodes) in graph 5.

Conclusion: When h(n) is reduced when h(n) is already an underestimate, it will sometimes find the graph’s optimal paths without expanding extra nodes.

1. **How does A\* perform when h(n) is the exact distance from n to a goal?**

The performance of A\* search when h(n) is the exact distance for n to a goal is dependent on the numbers of different optimal paths present. If there is a line of nodes with equal f-values, it will reach the goal through a single optimal path (best path) without expanding any unnecessary nodes. However, it does not happen all the time as it requires the perfect given information for A\* search to behave perfectly.

Conjecture: When h(n) is the exact distance from n to a goal, it will always produce the optimal path while expanding unnecessary nodes. To prove this, we set h(n) to be the exact distance from n to a goal.



Graph 6

Observations : Graph 6 found its optimal path by expanding only 5 nodes, which are the necessary nodes.

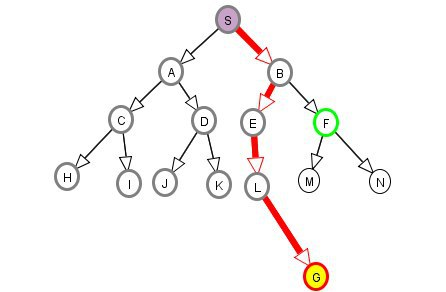
Conclusion: When h(n) is the exact distance from n to the goal node, it produces an optimal path without expanding extra nodes. Thus, it does not always expand unnecessary nodes to find an optimal path.

1. **What happens if h(n) is not an underestimate? You can give an example to justify your answer.**

If h(n) is not an underestimate, it is not guaranteed that it will find a shortest path, but it can run faster. Moreover, the algorithm would do unnecessary work on examining the paths it should be ignoring. This happens because the path cost is overestimated with the estimate cost, which would provide the algorithm with wrong ideas about the paths.

If h(n) happens to be relatively high to g(n), the A\* search turns into Greedy BFS.

Conjecture: When h(n) is not underestimate, it will always find an optimal path without expanding extra nodes. To prove this, Node B heuristic will be overestimated to check the graph’s response.



Graph 7

Observations: Graph 7 shows that the graph has expanded 7 extra nodes ( on the left side of the graph) before finding the optimal path. This is due to the overestimation of the heuristic in Node B. Thus, the total number of nodes that were expanded 12 nodes.

Conclusion: When h(n) is not an underestimate, it does sometimes expand extra nodes before finding an optimal path. Thus, it does not always find the best optimal bath when h(n) is an onverestimate.