

School of Computer Science and Engineering

CZ3005: ARTIFICIAL INTELLIGENCE

Semester 2 2020/2021

Lab 1

U1922414E

AIDE ISKANDAR BIN ABDUL RAHIM

C190147@e.ntu.edu.sg

Tutorial Group: TS4

Q1a) Give a graph where Depth-First search (DFS) is much more efficient (expands fewer nodes) than breadth-first search (BFS)

|  |  |
| --- | --- |
| Fig 1 |  |
| Notes | DFS (Node Expanded = 4): |
| Fig 2 |  |
| Notes | BFS (Node Expanded = 8): |

Q1b) Give a graph where BFS is much better than DFS.

|  |  |
| --- | --- |
| Fig 3 |  |
| Notes | BFS (Node Expanded = 5): |
| Fig 4 |  |
| Notes | DFS (Node Expanded = 6) |

Q1c) Give a graph where A\* search is more efficient than either DFS or BFS.

|  |  |
| --- | --- |
| Fig 5 |  |
| Notes | DFS (Node Expanded = 13) |
| Fig 6 |  |
| Notes | BFS (Node Expanded = 7) |
| Fig 7 |  |
| Notes | A\* Search (Node expanded = 4) |

Q1d) Give a graph where DFS and BFS are both more efficient than A\* search.

|  |  |
| --- | --- |
| Fig 8 |  |
| Notes | DFS (Node expanded = 7) |
| Fig 9 |  |
| Notes | BFS (Node expanded = 10) |
| Fig 10 |  |
| Notes | A\* Search (Node Expanded = 14) |
| Learning Points | For DFS and BFS search algorithm, only metric is used to find the goal node is just the number of nodes being expanded. When A\* search algorithm is added, Heuristic function h(n) is added into the metric to calculate to total cost function: f(n) = g(n) + h(n).  In this observation, we assume the DFS algorithm will expand from the left part of binary of tree, although, we can configure the algorithm to expand from the right. From question 1a and 1b observation, DFS search is faster when the goal node is at the deepest level (left leaf node) of the binary tree. However, this is not an optimal solution if the goal node is on the right side of the binary tree. On the other hand, BFS is will expand lesser node when the goal is not on the deepest level of the binary.  When A\* search is involved, we must set the heuristic level higher to be higher than other nodes for A\* search to less efficient than DFS and BFS. Another example would set heuristic level to lower than rest of the other nodes for A\* search to be more efficient than BFS and DFS. |

Question 2:

What is the effect of reducing h(n) when h(n) is already an underestimate?  
There would 3 cases will happen when h(n) has the lowest value

1. When the heuristic value h(n) is reduced on all neighboring nodes, the algorithm will expand same number of nodes and the optimal path still have the lowest heuristic value.
2. When the starting node and destination nodes ends up with the highest heuristic value h(n), the algorithm would search through all the nodes to get the optimal path
3. When iterating on a random graph, the algorithm would still be able to find a path, but it might not be the optimal path for the random graph.

How does A\* perform when h(n) is the exact distance from n to a goal?  
It depends on the edge cost [g(n)] by calculating f(n) = g(n) + h(n), taking the path that has the lowest f(n).

* In situation where all nodes have the same heuristic value h(n) and equal edge cost g(n), the algorithm will just behave similarly to a Breadth First Search (BFS) algorithm.
* In another situation where all nodes have the same heuristic value h(n) and equal edge cost g(n), except for edges that cost lesser. The algorithm will proceed to expand the node that cost lesser.

What happens if h(n) is not an underestimate? You can give an example to justify your answer.  
In the case of using a binary graph, When h(n) is overestimated, the calculated f(n) of A\* will be the longest path which resulting on expanding more node to find the optimal path.  
  
In another case of using a random graph, The algorithm’s selected path might be not be the optimal path, which speculates that the optimal path might be “overlooked” by the current selected path.