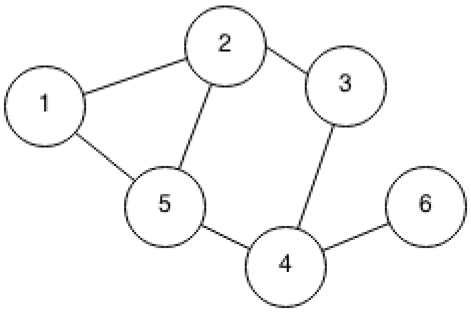
Search Algorithms comparison

Chadi Hani – 17105697

Breadth First Search

BFS refers to the method in which you traverse a graph by visiting all children of a node before moving on to that child's children. You can think in terms of levels. If the root node is Level 1, you visit all of Level 2, then all of Level 3, all of Level 4, and so on and so forth.



Depth First Search

Whereas BFS goes level by level, DFS follows the path of one child as far as it can go, from root to finish, before going back and starting down the path of another child.

	BFS	**DFS**
1.	BFS stands for Breadth First Search.	DFS stands for Depth First Search.
2.	BFS(Breadth First Search) uses Queue data structure for finding the shortest path.	DFS(Depth First Search) uses Stack data structure.

5.

3.	BFS can be used to find single source shortest path in an unweighted graph, because in BFS, we reach a vertex with minimum number of edges from a source vertex.	In DFS, we might traverse through more edges to reach a destination vertex from a source.
3.	**BFS is more suitable for searching vertices which are closer to the given source.**	**DFS is more suitable when there are solutions away from source.**
4.	**BFS considers all neighbors first and therefore not suitable for decision making trees used in games or puzzles.**	**DFS is more suitable for game or puzzle problems. We make a decision, then explore all paths through this decision. And if this decision leads to win situation, we stop.**
5.	The Time complexity of BFS is O(V + E) when Adjacency List is used and O(V^2) when Adjacency Matrix is used, where V stands for vertices and E stands for edges.	The Time complexity of DFS is also O(V + E) when Adjacency List is used and O(V^2) when Adjacency Matrix is used, where V stands for vertices and E stands for edges.

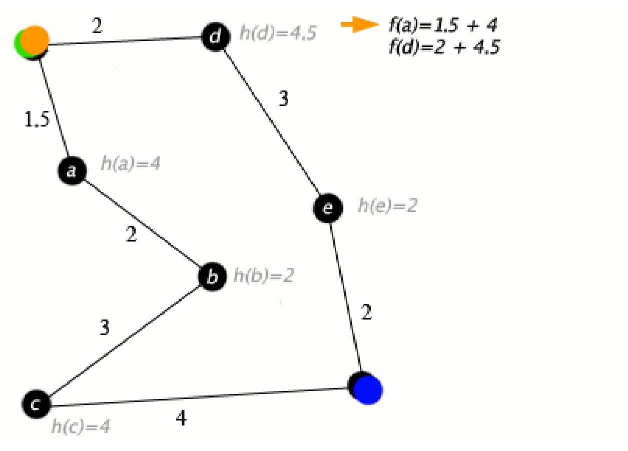
Best First Search

In BFS and DFS, when we are at a node, we can consider any of the adjacent as next node. So both BFS and DFS blindly explore paths without considering any cost function. The idea of Best First Search is to use an evaluation function to decide which adjacent is most promising and then explore. Best First Search falls under the category of Heuristic Search or Informed Search.

We use a priority queue to store costs of nodes. So the implementation is a variation of BFS, we just need to change Queue to PriorityQueue.

A* Search

Informally speaking, A* Search algorithms, unlike other traversal techniques, it has "brains". What it means is that it is really a smart algorithm which separates it from the other conventional algorithms. This fact is cleared in detail in below sections.
And it is also worth mentioning that many games and web-based maps use this algorithm to find the shortest path very efficiently (approximation).

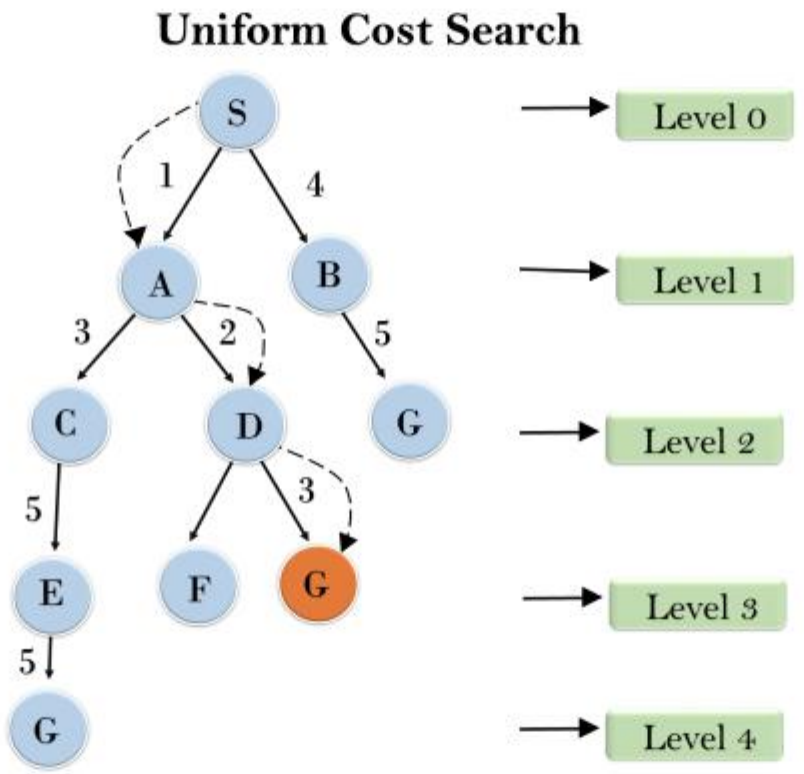


Uniform-cost Search Algorithm:

Uniform-cost search is a searching algorithm used for traversing a weighted tree or graph. This algorithm comes into play when a different cost is available for each edge. The primary goal of the uniform-cost search is to find a path to the goal node which has the lowest cumulative cost. Uniform-cost search expands nodes according to their path costs form the root node. It can be used to solve any graph/tree where the optimal cost is in demand. A uniform-cost search algorithm is implemented by the priority queue. It gives maximum priority to the lowest cumulative cost. Uniform cost search is equivalent to BFS algorithm if the path cost of all edges is the same.

Advantages:

Uniform cost search is optimal because at every state the path with the least cost is chosen.


Uniform Cost Search
S
1
4
A
B
3
2
5
C
D
G
5
3
E
F
G
5
G
Level 0
Level 1
Level 2
Level 3
Level 4