**Searching Algorithm**

**Notes by (Abu sufyan)**

**Breadth First Search (BFS) Algorithm with EXAMPLE**

**What is BFS Algorithm (Breadth-First Search)?**

Breadth-first search (BFS) is an algorithm that is used to graph data or searching tree or traversing structures. The full form of BFS is the Breadth-first search.

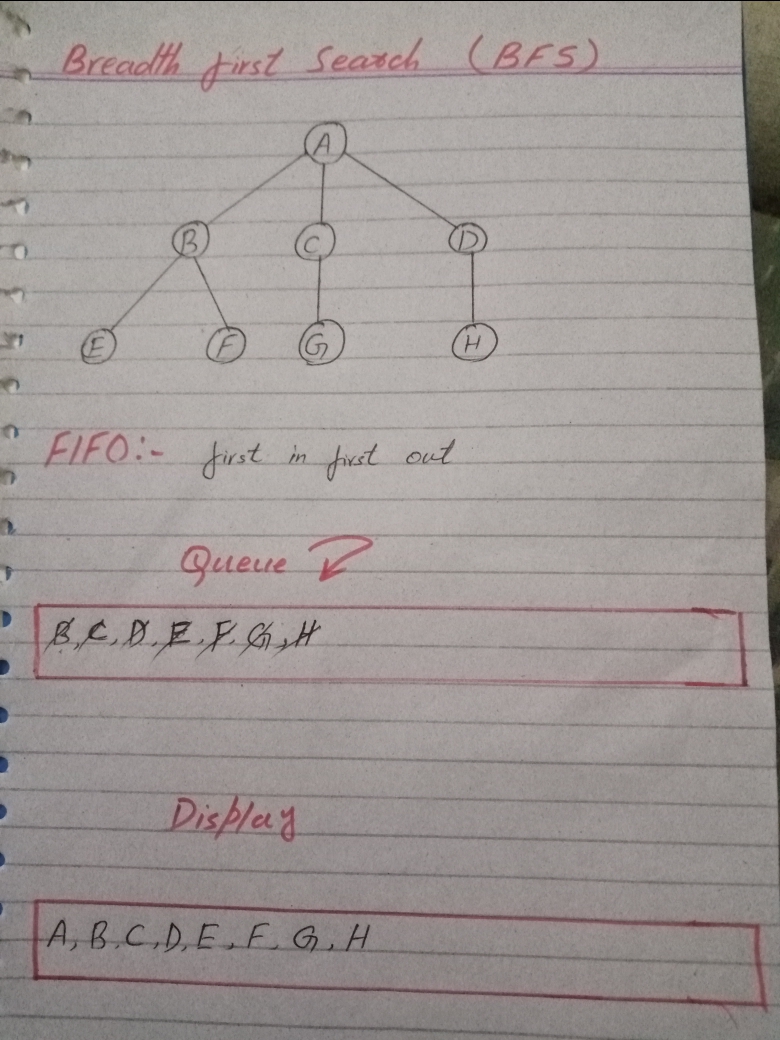
The algorithm efficiently visits and marks all the key nodes in a graph in an accurate breadthwise fashion. This algorithm selects a single node (initial or source point) in a graph and then visits all the nodes adjacent to the selected node. Remember, BFS accesses these nodes one by one.

Once the algorithm visits and marks the starting node, then it moves towards the nearest unvisited nodes and analyses them. Once visited, all nodes are marked. These iterations continue until all the nodes of the graph have been successfully visited and marked.

**What is Graph traversals?**

A graph traversal is a commonly used methodology for locating the vertex position in the graph. It is an advanced search algorithm that can analyze the graph with speed and precision along with marking the sequence of the visited vertices. This process enables you to quickly visit each node in a graph without being locked in an infinite loop.

**BFS example given below:**

****

**DFS example:**

**Depth First Search (DFS)**

The DFS algorithm is a recursive algorithm that uses the idea of backtracking. It involves exhaustive searches of all the nodes by going ahead, if possible, else by backtracking.

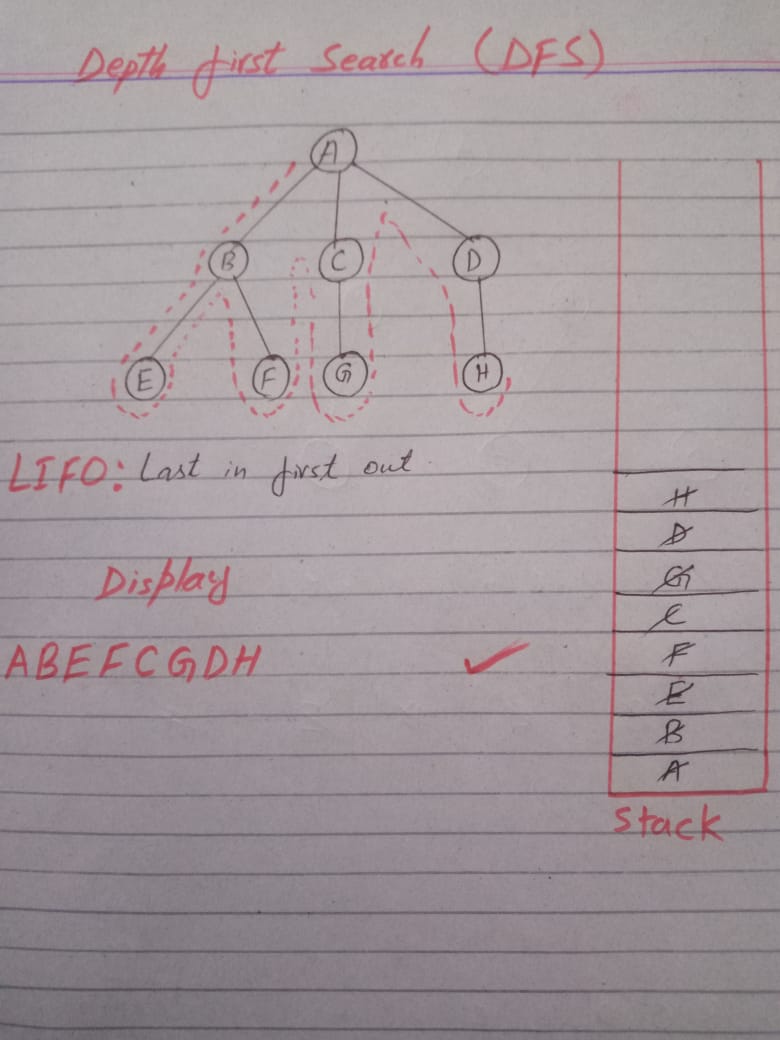
Here, the word backtrack means that when you are moving forward and there are no more nodes along the current path, you move backwards on the same path to find nodes to traverse. All the nodes will be visited on the current path till all the unvisited nodes have been traversed after which the next path will be selected.

This recursive nature of DFS can be implemented using stacks. The basic idea is as follows:  
Pick a starting node and push all its adjacent nodes into a stack.  
Pop a node from stack to select the next node to visit and push all its adjacent nodes into a stack.  
Repeat this process until the stack is empty. However, ensure that the nodes that are visited are marked. This will prevent you from visiting the same node more than once. If you do not mark the nodes that are visited and you visit the same node more than once, you may end up in an infinite loop.

***How to find connected components using DFS?***

A graph is said to be disconnected if it is not connected, i.e. if two nodes exist in the graph such that there is no edge in between those nodes. In an undirected graph, a connected component is a set of vertices in a graph that are linked to each other by paths

DFS example given below.

****

**A\* algorithm example:**

**A \* algorithm** is a searching algorithm that searches for the shortest path between the *initial and the final state.* It is used in various applications, such as *maps*.

In *maps* the A\* algorithm is used to calculate the shortest distance between the source (initial state) and the destination (final state).

**How it works**

Imagine a square grid which possesses many obstacles, scattered randomly. The initial and the final cell is provided. The aim is to reach the final cell in the shortest amount of time.

Here A\* Search Algorithm comes to the rescue:

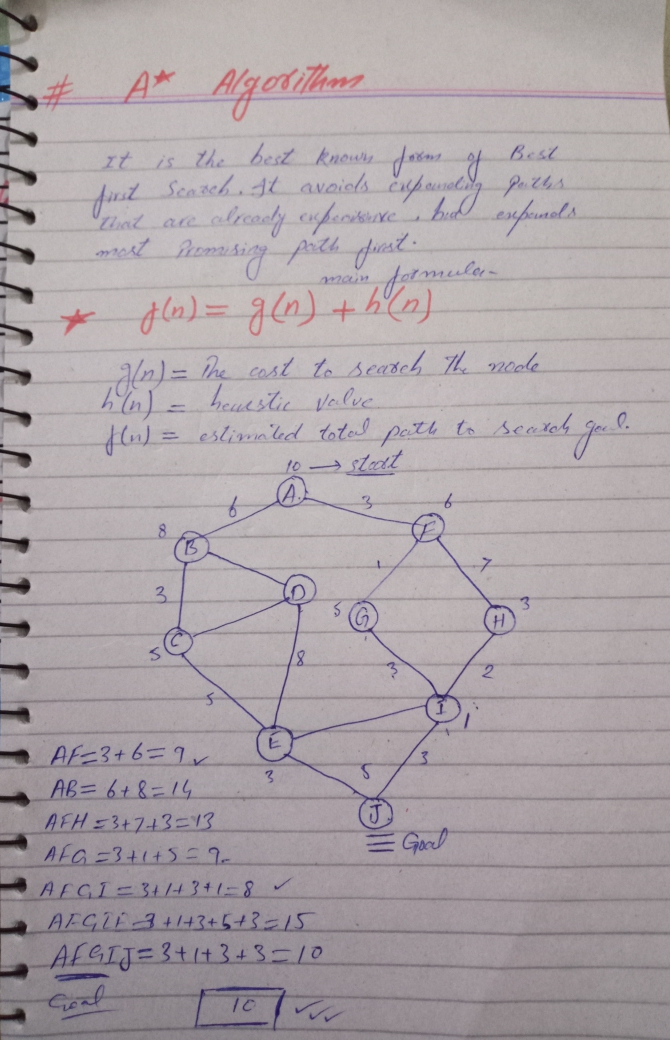
**Explanation**

A\* algorithm has 3 parameters:

* **g :** the cost of moving from the initial cell to the current cell. Basically, it is the sum of all the cells that have been visited since leaving the first cell.
* **h :** also known as the *heuristic value,* it is the **estimated** cost of moving from the current cell to the final cell. The actual cost cannot be calculated until the final cell is reached. Hence, h is the estimated cost. We **must** make sure that there is **never** an over estimation of the cost.
* **f :** it is the sum of g and h. So, **f = g + h**

The way that the algorithm makes its decisions is by taking the f-value into account. The algorithm selects the *smallest f-valued cell* and moves to that cell. This process continues until the algorithm reaches its goal cell.

A\* example given below.

****