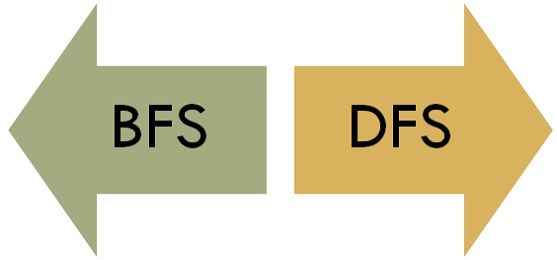
Difference Between BFS and DFS

The major difference between BFS and DFS is that BFS proceeds level by level while DFS follows first a path form the starting to the ending node (vertex), then another path from the start to end, and so on until all nodes are visited. Furthermore, BFS uses the queue for storing the nodes whereas DFS uses the stack for traversal of the nodes.

BFS and DFS are the traversing methods used in searching a graph. Graph traversal is the process of visiting all the nodes of the graph. A graph is a group of Vertices ‘V’ and Edges ‘E’ connecting to the vertices.

Content: BFS Vs DFS

1. [Comparison Chart](https://techdifferences.com/difference-between-bfs-and-dfs.html#ComparisonChart)
2. [Definition](https://techdifferences.com/difference-between-bfs-and-dfs.html#Definition)
3. [Key Differences](https://techdifferences.com/difference-between-bfs-and-dfs.html#KeyDifferences)
4. [Conclusion](https://techdifferences.com/difference-between-bfs-and-dfs.html#Conclusion)

Comparison Chart

| **BASIS FOR COMPARISON** | **BFS** | **DFS** |
| --- | --- | --- |
| Basic | Vertex-based algorithm | Edge-based algorithm |
| Data structure used to store the nodes | Queue | Stack |
| Memory consumption | Inefficient | Efficient |
| Structure of the constructed tree | Wide and short | Narrow and long |
| Traversing fashion | Oldest unvisited vertices are explored at first. | Vertices along the edge are explored in the beginning. |
| Optimality | Optimal for finding the shortest distance, not in cost. | Not optimal |
| Application | Examines bipartite graph, connected component and shortest path present in a graph. | Examines two-edge connected graph, strongly connected graph, acyclic graph and topological order. |

Definition of BFS

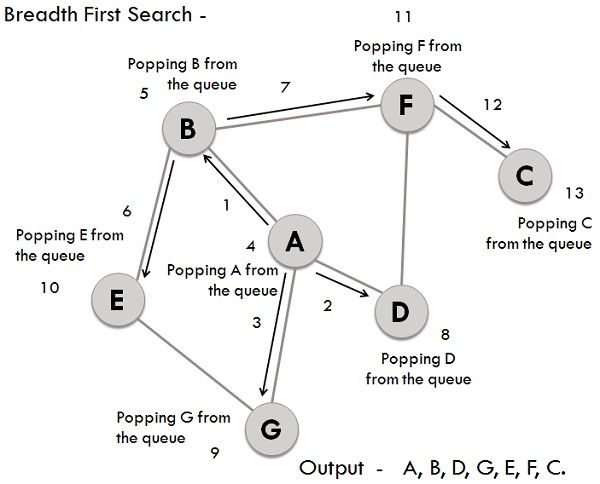
**Breadth First Search (BFS)** is the traversing method used in graphs. It uses a queue for storing the visited vertices. In this method the emphasize is on the vertices of the graph, one vertex is selected at first then it is visited and marked. The vertices adjacent to the visited vertex are then visited and stored in the queue sequentially.

Similarly, the stored vertices are then treated one by one, and their adjacent vertices are visited. A node is fully explored before visiting any other node in the graph, in other words, it traverses shallowest unexplored nodes first.

Example

We have a graph whose vertices are A, B, C, D, E, F, G. Considering A as starting point. The steps involved in the process are:

* Vertex A is expanded and stored in the queue.
* Vertices B, D and G successors of A, are expanded and stored in the queue meanwhile Vertex A removed.
* Now B at the front end of the queue is removed along with storing its successor vertices E and F.
* Vertex D is at the front end of the queue is removed, and its connected node F is already visited.
* Vertex G is removed from the queue, and it has successor E which is already visited.
* Now E and F are removed from the queue, and its successor vertex C is traversed and stored in the queue.
* At last C is also removed and the queue is empty which means we are done.
* The generated Output is – A, B, D, G, E, F, C.

Applications

BFS can be useful in finding whether the graph has **connected components** or not. And also it can be used in detecting a **bipartite graph**.

A graph is bipartite when the graph vertices are parted into two disjoint sets; no two adjacent vertices would reside in the same set. Another method of checking a bipartite graph is to check the occurrence of an odd cycle in the graph. A bipartite graph must not contain odd cycle.

BFS is also better at finding the **shortest path** in the graph could be seen as a network.

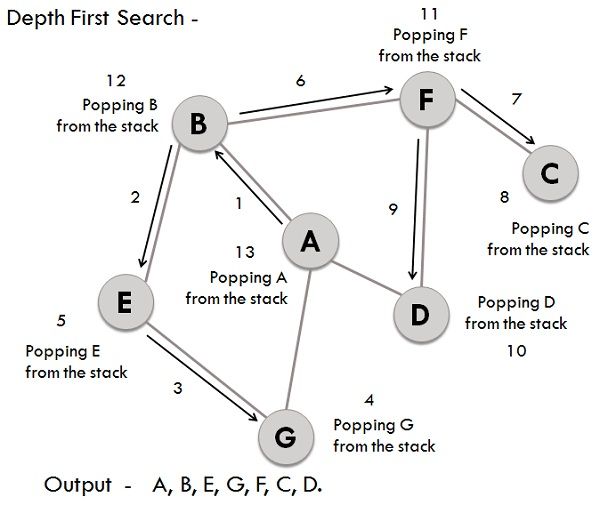
Definition of DFS

**Depth First Search (DFS)** traversing method uses the stack for storing the visited vertices. DFS is the edge based method and works in the recursive fashion where the vertices are explored along a path (edge). The exploration of a node is suspended as soon as another unexplored node is found and the deepest unexplored nodes are traversed at foremost. DFS traverse/visit each vertex exactly once and each edge is inspected exactly twice.

Example

Similar to BFS lets take the same graph for performing DFS operations, and the involved steps are:

* Considering A as the starting vertex which is explored and stored in the stack.
* B successor vertex of A is stored in the stack.
* Vertex B have two successors E and F, among them alphabetically E is explored first and stored in the stack.
* The successor of vertex E, i.e., G is stored in the stack.
* Vertex G have two connected vertices, and both are already visited, so G is popped out from the stack.
* Similarly, E s also removed.
* Now, vertex B is at the top of the stack, its another node(vertex) F is explored and stored in the stack.
* Vertex F has two successor C and D, between them C is traversed first and stored in the stack.
* Vertex C only have one predecessor which is already visited, so it is removed from the stack.
* Now vertex D connected to F is visited and stored in the stack.
* As vertex D doesn’t have any unvisited nodes, therefore D is removed.
* Similarly, F, B and A are also popped.
* The generated output is – A, B, E, G, F, C, D.

Application

The applications of DFS includes the inspection of **two edge connected** graph, **strongly connected** graph, **acyclic graph**, and **topological order**.

A graph is called two edges connected if and only if it remains connected even if one of its edges is removed. This application is very useful, in computer networks where the failure of one link in the network will not affect the remaining network, and it would be still connected.

Strongly connected graph is a graph in which there must exist a path between ordered pair of vertices. DFS is used in the directed graph for searching the path between every ordered pair of vertices. DFS can easily solve connectivity problems.

Key Differences Between BFS and DFS

1. BFS is vertex-based algorithm while DFS is an edge-based algorithm.
2. Queue data structure is used in BFS. On the other hand, DFS uses stack or recursion.
3. Memory space is efficiently utilized in DFS while space utilization in BFS is not effective.
4. BFS is optimal algorithm while DFS is not optimal.
5. DFS constructs narrow and long trees. As against, BFS constructs wide and short tree.

Conclusion

BFS and DFS, both of the graph searching techniques have similar running time but different space consumption, DFS takes linear space because we have to remember single path with unexplored nodes, while BFS keeps every node in memory.

DFS yields deeper solutions and is not optimal, but it works well when the solution is dense whereas BFS is optimal which searches the optimal goal at first.