

Graph Traversal - BFS

Graph traversal is a technique used for searching a vertex in a graph. The graph traversal is also used to decide the order of vertices is visited in the search process. A graph traversal finds the edges to be used in the search process without creating loops. That means using graph traversal we visit all the vertices of the graph without getting into looping path.

There are two graph traversal techniques and they are as follows...

1. **DFS (Depth First Search)**
2. **BFS (Breadth First Search)**

BFS (Breadth First Search)

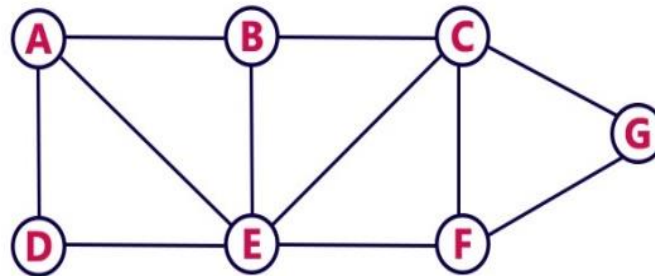
BFS traversal of a graph produces a **spanning tree** as final result. **Spanning Tree** is a graph without loops. We use **Queue data structure** with maximum size of total number of vertices in the graph to implement BFS traversal.

We use the following steps to implement BFS traversal...

- **Step 1** - Define a Queue of size total number of vertices in the graph.
- **Step 2** - Select any vertex as **starting point** for traversal. Visit that vertex and insert it into the Queue.
- **Step 3** - Visit all the non-visited **adjacent** vertices of the vertex which is at front of the Queue and insert them into the Queue.
- **Step 4** - When there is no new vertex to be visited from the vertex which is at front of the Queue then delete that vertex.

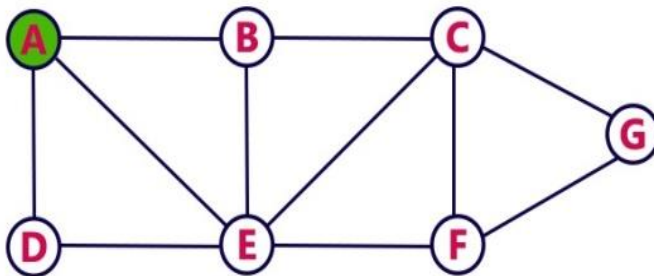
- **Step 5** - Repeat steps 3 and 4 until queue becomes empty.
- **Step 6** - When queue becomes empty, then produce final spanning tree by removing unused edges from the graph

Consider the following example graph to perform BFS traversal



Step 1:

- Select the vertex **A** as starting point (visit **A**).
- Insert **A** into the Queue.

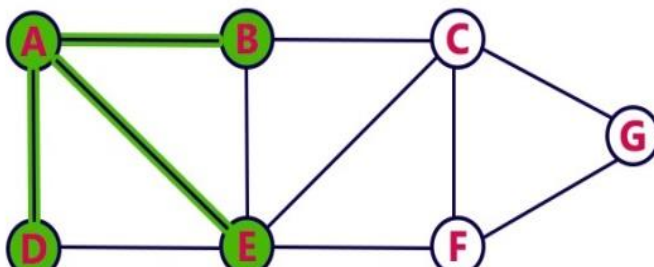


Queue



Step 2:

- Visit all adjacent vertices of **A** which are not visited (**D, E, B**).
- Insert newly visited vertices into the Queue and delete A from the Queue..

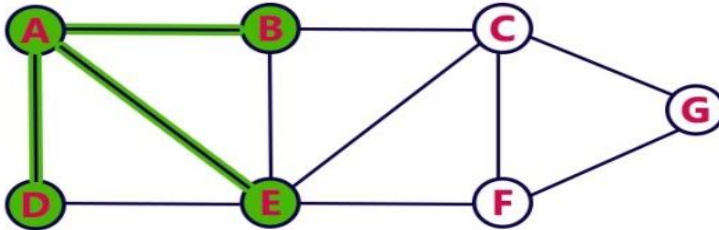


Queue



Step 3:

- Visit all adjacent vertices of **D** which are not visited (there is no vertex).
- Delete D from the Queue.

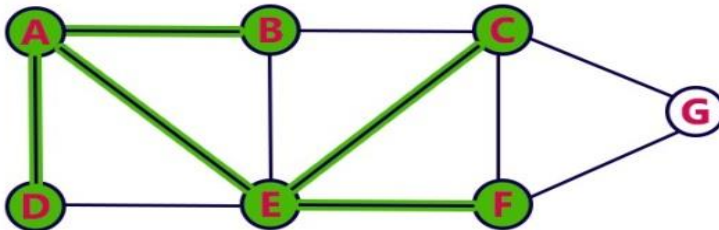


Queue



Step 4:

- Visit all adjacent vertices of **E** which are not visited (**C, F**).
- Insert newly visited vertices into the Queue and delete E from the Queue.

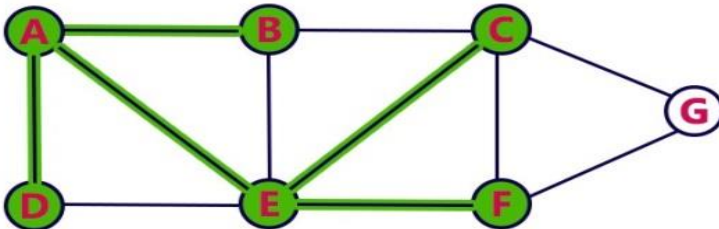


Queue



Step 5:

- Visit all adjacent vertices of **B** which are not visited (**there is no vertex**).
- Delete **B** from the Queue.

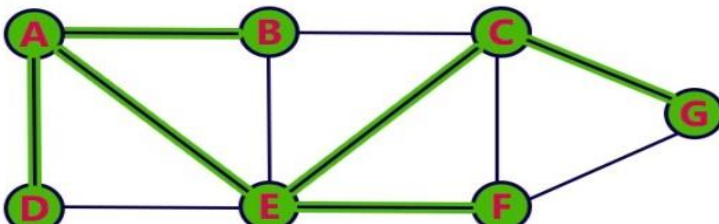


Queue



Step 6:

- Visit all adjacent vertices of **C** which are not visited (**G**).
- Insert newly visited vertex into the Queue and delete **C** from the Queue.

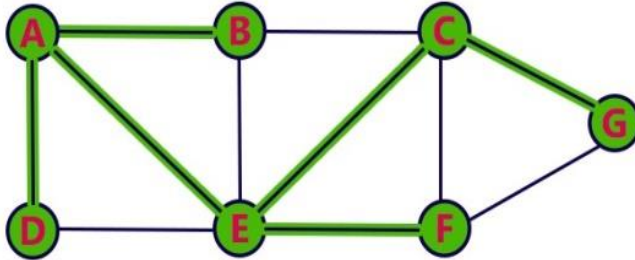


Queue



Step 7:

- Visit all adjacent vertices of **F** which are not visited (**there is no vertex**).
- Delete **F** from the Queue.

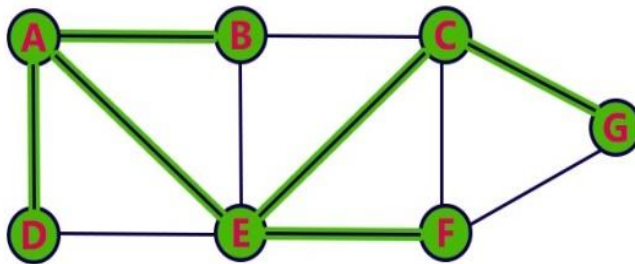


Queue



Step 8:

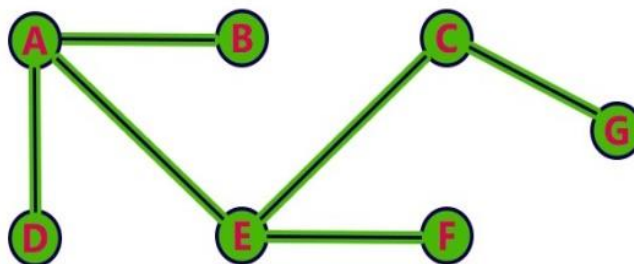
- Visit all adjacent vertices of **G** which are not visited (**there is no vertex**).
- Delete **G** from the Queue.



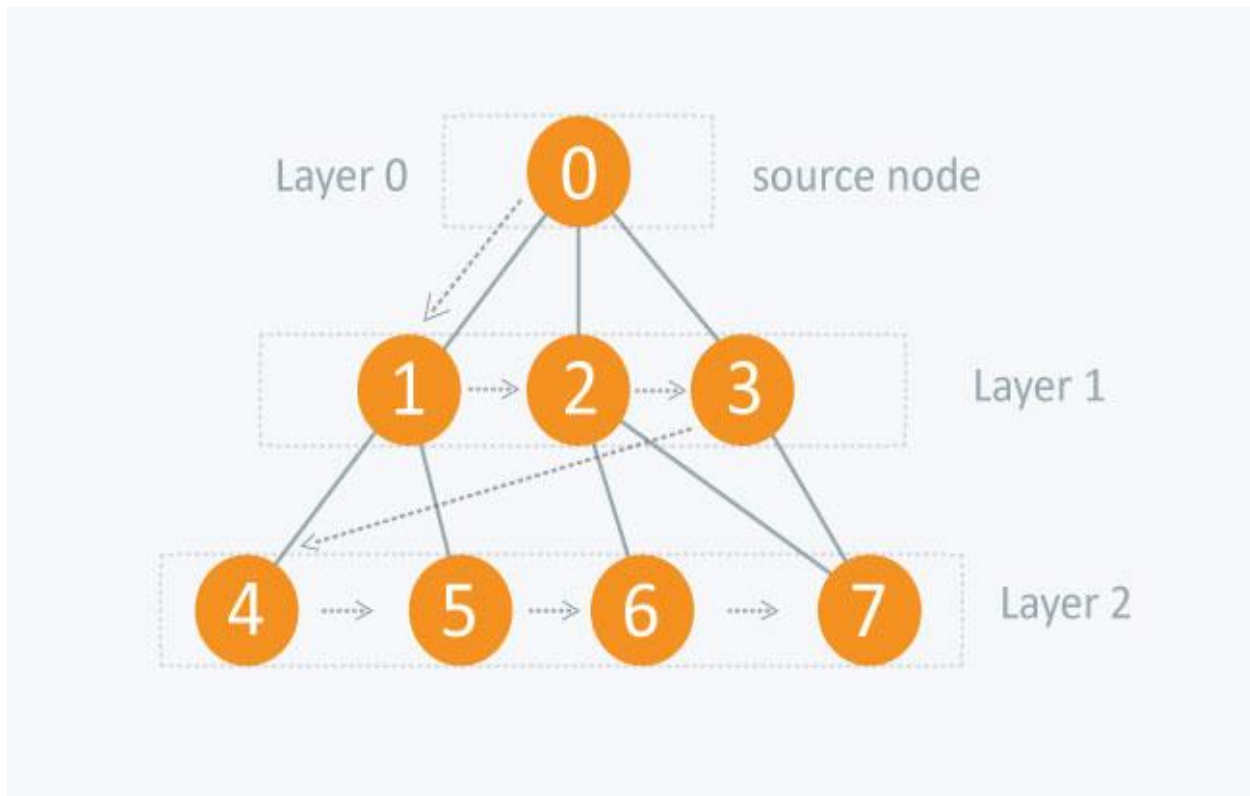
Queue



-
- Queue became Empty. So, stop the BFS process.
 - Final result of BFS is a Spanning Tree as shown below...



Example 2:



Applications of Breadth First Traversal

1) Shortest Path and Minimum Spanning Tree for unweighted graph In an unweighted graph, the shortest path is the path with least number of edges. With Breadth First, we always reach a vertex from given source using the minimum number of edges. Also, in case of unweighted graphs, any spanning tree is Minimum Spanning Tree and we can use either Depth or Breadth first traversal for finding a spanning tree.

2) Peer to Peer Networks. In Peer to Peer Networks like [BitTorrent](#), Breadth First Search is used to find all neighbor nodes.

3) Crawlers in Search Engines: Crawlers build index using Breadth First. The idea is to start from source page and follow all links from source and keep doing same. Depth First Traversal can also be used for crawlers, but the advantage with Breadth First Traversal is, depth or levels of the built tree can be limited.

4) Social Networking Websites: In social networks, we can find people within a given distance 'k' from a person using Breadth First Search till 'k' levels.

5) GPS Navigation systems: Breadth First Search is used to find all neighboring locations.

6) Broadcasting in Network: In networks, a broadcasted packet follows Breadth First Search to reach all nodes.

7) In Garbage Collection: Breadth First Search is used in copying garbage collection using **Cheney's algorithm**. Refer **this** and for details. Breadth First Search is preferred over Depth First Search because of better locality of reference:

8) Cycle detection in undirected graph: In undirected graphs, either Breadth First Search or Depth First Search can be used to detect cycle. We can use **BFS to detect cycle in a directed graph** also,

9) Ford-Fulkerson algorithm In Ford-Fulkerson algorithm, we can either use Breadth First or Depth First Traversal to find the maximum flow. Breadth First Traversal is preferred as it reduces worst case time complexity to $O(VE^2)$.

10) To test if a graph is Bipartite We can either use Breadth First or Depth First Traversal.

11) Path Finding We can either use Breadth First or Depth First Traversal to find if there is a path between two vertices.

12) Finding all nodes within one connected component: We can either use Breadth First or Depth First Traversal to find all nodes reachable from a given node.

Many algorithms like **Prim's Minimum Spanning Tree** and **Dijkstra's Single Source Shortest Path** use structure similar to Breadth First Search.

There can be many more applications as Breadth First Search is one of the core algorithms for Graphs.