Sairam INSTITUTIONS

Sri SAI RAM ENGINEERING COLLEGE
INSTITUTE OF TECHNOLOGY
West Tambaram, Chennai - 44

YEAR II | SEM III

CS8391

DATA STRUCTURES (COMMON TO CSE & IT)

UNIT No. 4

4.2 BREADTH FIRST TRAVERSAL

Version 1.00

4 QUALITY EDUCATION

BREADTH FIRST TRAVERSAL

GRAPH TRAVERSAL: Traversing the graph means examining all the nodes and vertices of the graph. There are two standard methods by using which, we can traverse the graphs.

- Breadth First Search
- Depth First Search

BREADTH FIRST SEARCH: Breadth first search is a graph traversal algorithm that starts traversing the graph from root node and explores all the neighbouring nodes.

It selects the nearest node and explore all the unexplored nodes. The algorithm follows the same process for each of the nearest node until all the nodes in the graph visited**.**

Algorithm:

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.

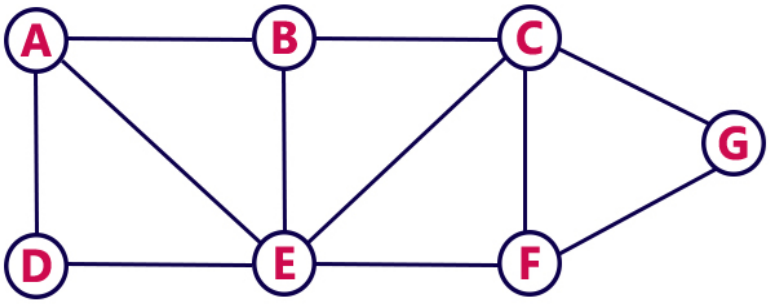
Step 4 - When there is no new vertex to be visited from the vertex which is at front of the Queue then delete that vertex.

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

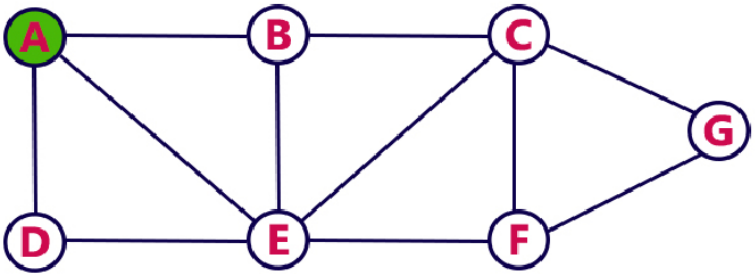
Example:

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.

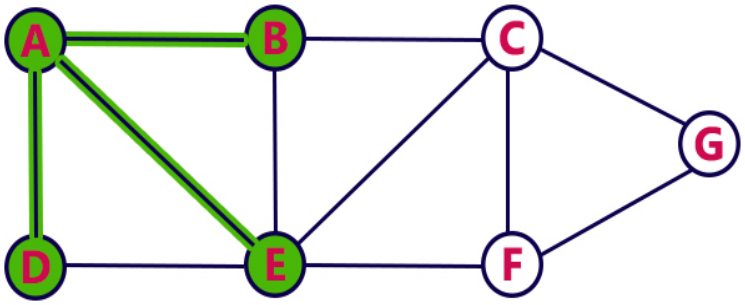


Queue

A					

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..

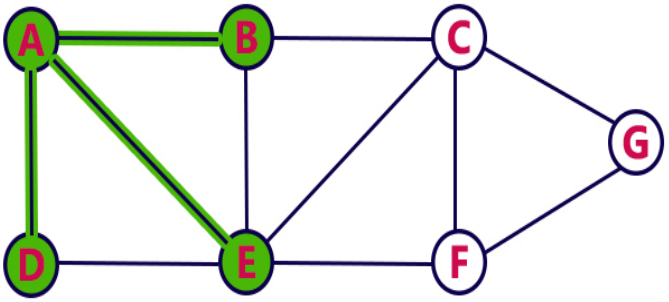


Queue

	D	E	B		

Step 3:

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

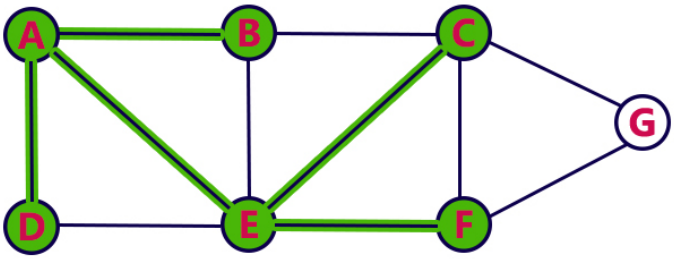


Queue

		E	B			

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.

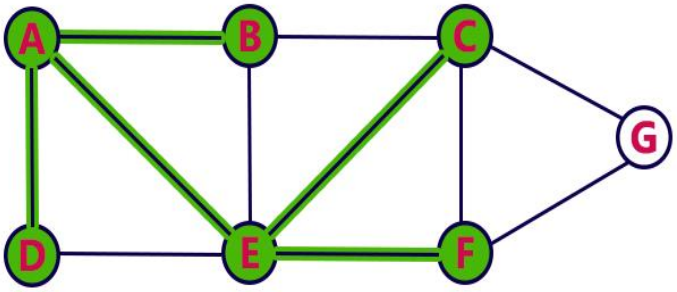


Queue

			B	C	F	

Step 5:

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

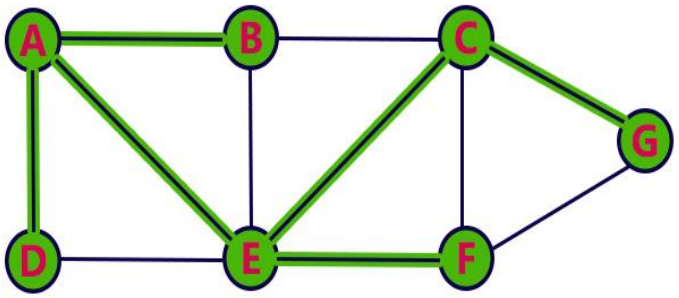


Queue

				C	F	

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.

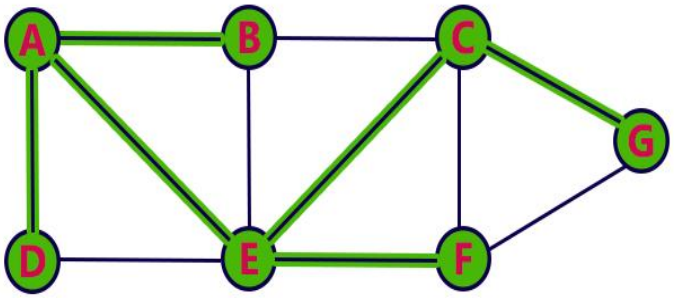


Queue

					F	G

Step 7:

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

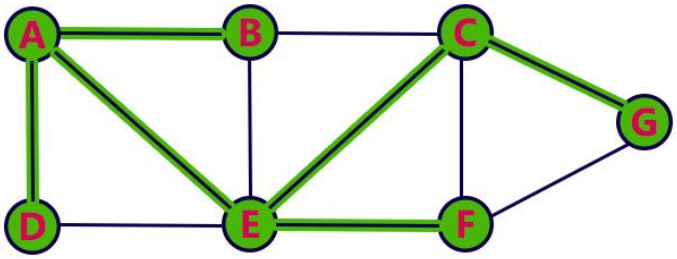


Queue

						G

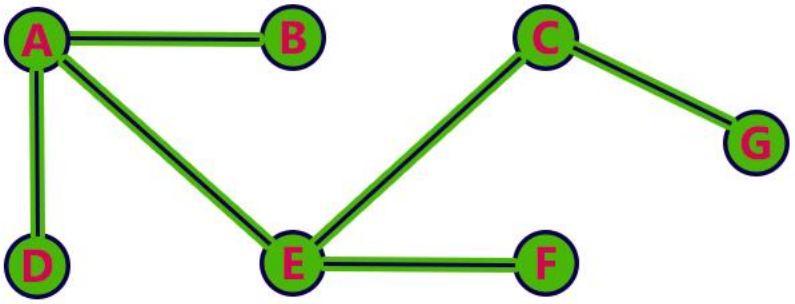
Step 8:

- 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...



Application of BFS

- Un-weighted Graphs: BFS algorithm can easily create the shortest path and a minimum spanning tree to visit all the vertices of the graph in the shortest time possible with high accuracy.
- P2P Networks: BFS can be implemented to locate all the nearest or neighboring nodes in a peer to peer network. This will find the required data faster.
- Web Crawlers: Search engines or web crawlers can easily build multiple levels of indexes by employing BFS. BFS implementation starts from the source, which is the web page, and then it visits all the links from that source.
- Navigation Systems: BFS can help find all the neighboring locations from the main or source location.
- Network Broadcasting: A broadcasted packet is guided by the BFS algorithm to find and reach all the nodes it has the address for.