## BFS:

Breadth first search:

⇒ It produces spanning tree as a final result is graph without loops.

⇒ We use Queue data structure with maximum size of total number of vertices in the graph to implement BFS of a graph:

### Algorithm:

Step 1: Define a Queue of size, ie total number of vertices in the graph.

Step 2: Select any vertex as starting point from traversal. Visit vertex and insert it into the Queue.

Step 3:

Visit all the adjacent vertices of the vertex which is at front of the queue, which is not visited and insert them into the queue.

Step 4:

When there is no new vertex to be visit from the vertex at front of the queue then delete that vertex from the Queue.

Step 5:

Repeat step 3 & 4 until queue becomes empty.

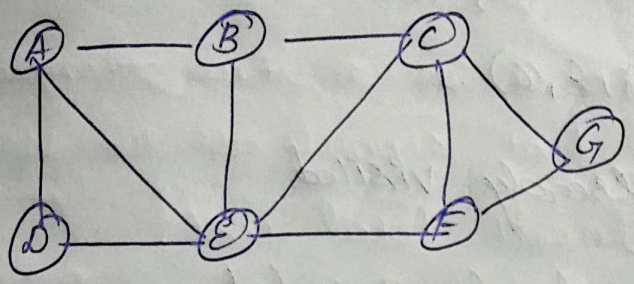
Step 6:

When Queue becomes empty then produce final spanning tree by removing unused edges from the graph.

Routines

```
Void BFT (vertex v)
{
Initialize Queue Q;
Visited [v] = 1;
Enqueue (v, Q);
while ( ! IsEmpty (Q))
{
  u = Dequeue (Q);
  Print u;
for all vertices v adjacent to u
if ( visited [v] = = 0) then
{
  Enqueue (v, Q) ; visited [v] = 1;
}}.
```

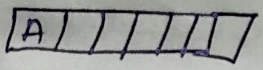
Example:




Step 1:

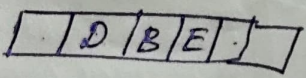
⇒ Select vertex A as starting point

⇒ Insert A into Queue.




Step 2:

* Visit all adj(A) which are not visited (D, B, E).

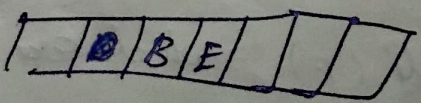



Step 3:

* Visit all adj (D), which are not visited (A, E) ⇒ already visited.

* There is no vertex

* Delete D from Queue.



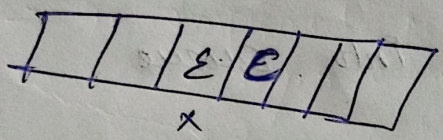
Step 4:

* visit all adj (B), which are not visited (A, E, C).

* A → already visited

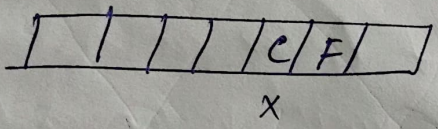
* E → already inserted

C → Inserted



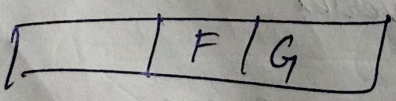

Step 5:

* visit all adj (E), which are not visited (P, B, C, F).



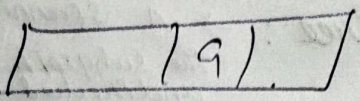

Step 6.

* visit all adj (c), which are not visited (B, E, F, G)



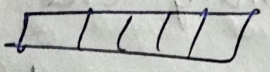

Step 7:

* visit all adj (F), which are not visited (E, C, G)

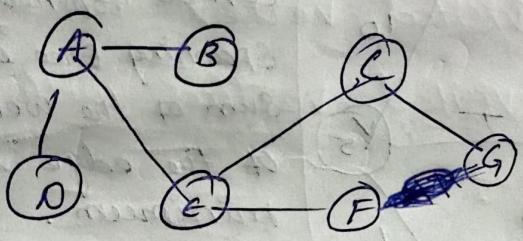



Step 8:

* Adj (G)
* $G = (C, E)$



* Queue is became empty, stop BFS process
* final spanning tree by removing unused edges.




Application of BFS:

1. unweighted Graphs
2. Peer to peer networks.
3. Web crawlers
4. Navigation systems
5. Network Broadcasting.