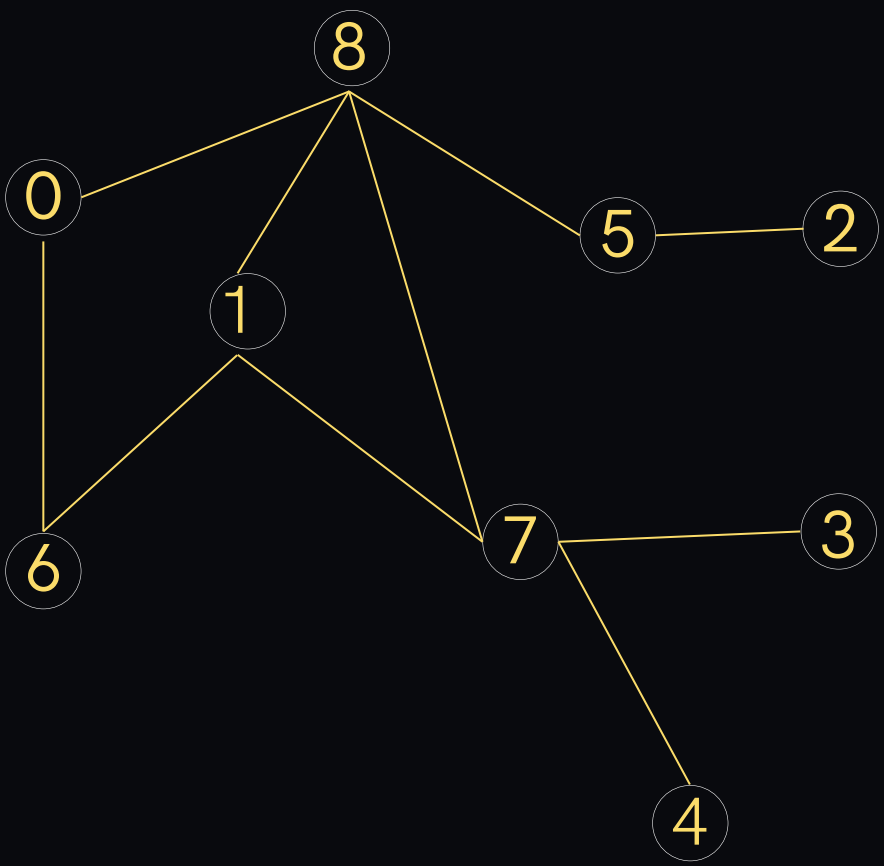
BREATH
FIRST
SEARCH

```cpp
#include<bits/stdc++.h>
using namespace std;
void create_adjacency_list(int vertex,vector<pair<int,int>>&edges
,unordered_map<int,set<int>>&adj)
{
    cout<<"\nadjancency list for graph\n";
    for(auto it:edges)
    {
        adj[it.first].insert(it.second);
        adj[it.second].insert(it.first);
    }

    for(auto it:adj)
    {
        cout<<it.first<<"->";
        for(auto x:it.second)
        {
            cout<<x<<" ";
        }
        cout<<endl;
    }
}

void bfs(int vertex,unordered_map<int,set<int>>&adj
,vector<int>&BFS,int node,unordered_map<int,bool>visit)
{
    BFS.clear();
    queue<int>q;
    q.push(node);
    visit[node]=1;
    while(!q.empty())
    {
        int front=q.front();
        q.pop();
        BFS.push_back(front);

        for(auto it:adj[front])
        {
        {
            if(!visit[it])
            {
                q.push(it);
                visit[it]=1;
            }
        }
    }

    for(auto x:BFS) cout<<x<<" ";
}

int main()
{
    int vertex;
    cout<<"enter no. of vertices-->";cin>>vertex;
    vector<pair<int,int>>edges;
    int x,y;
    cout<<"enter edges , press -1 -1 to quit\t";
    cout<<"\t-- USE 0 BASED INDEXING ---\n";
    while(x!=-1 && y!=-1)
    {
```

```
62          if(x!=-1 && y!=-1)
63          edges.push_back(make_pair(x,y));
64      }
65
66      unordered_map<int,set<int>>adj;
67      cout<<"\n\n";
68      create_adjacency_list(vertex,edges,adj);
69
70      vector<int>BFS;
71      unordered_map<int,bool>visit;
72      for(int i=0;i<vertex;i++)
73      {
74
75      cout<<"\nBFS Traversal for root as "<< i << "-->";
76      bfs(vertex,adj,BFS,i,visit);
77
78      }
79  }
```

INPUT GRAPH



OUTPUT

```
9enter no. of vertices-->
enter edges , press -1 -1 to quit                -- USE 0 BASED INDEXING ---
0 8
1 6
1 7
1 8
5 8
6 0
7 3
7 4
8 7
2 5
-1 -1



adjancency list for graph
2->5
4->7
3->7
5->2 8
7->1 3 4 8
6->0 1
1->6 7 8
8->0 1 5 7
0->6 8

BFS Traversal for root as 0-->0 6 8 1 5 7 2 3 4
BFS Traversal for root as 1-->1 6 7 8 0 3 4 5 2
BFS Traversal for root as 2-->2 5 8 0 1 7 6 3 4
BFS Traversal for root as 3-->3 7 1 4 8 6 0 5 2
BFS Traversal for root as 4-->4 7 1 3 8 6 0 5 2
BFS Traversal for root as 5-->5 2 8 0 1 7 6 3 4
BFS Traversal for root as 6-->6 0 1 8 7 5 3 4 2
BFS Traversal for root as 7-->7 1 3 4 8 6 0 5 2
BFS Traversal for root as 8-->8 0 1 5 7 6 2 3 4 %
jatin@Jatins-Air Algos %
```