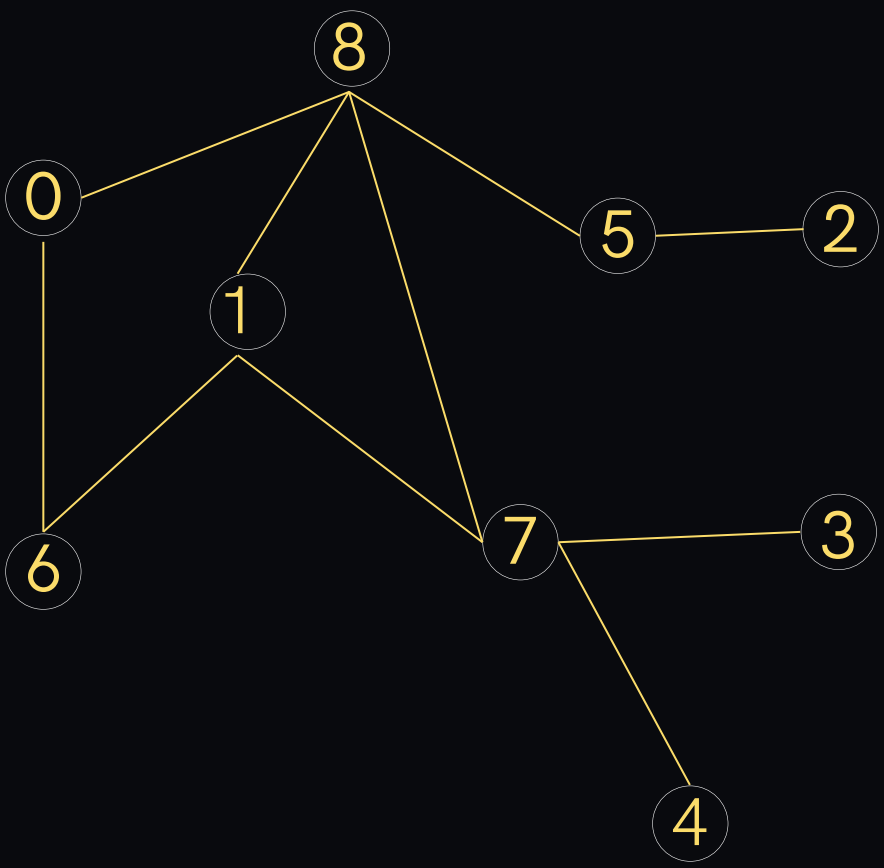
# DEPTH FIRST SEARCH

```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 dfs(int vertex,unordered_map<int,set<int>>&adj
,vector<int>&DFS,int node,unordered_map<int,bool>&visit)
{
    DFS.push_back(node);
    visit[node]=1;

    for(auto it:adj[node])
    {
        if(!visit[it])
        {
            dfs(vertex,adj,DFS,it,visit);
        }
    }

    // for(auto x:DFS) 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)
    {
        cin>>x;
        cin>>y;
        if(x!=-1 && y!=-1)
        edges.push_back(make_pair(x,y));
    }

    unordered_map<int,set<int>>adj;
    cout<<"\n\n";
```

```
    vector<int>DFS;
    unordered_map<int,bool>visit;
    for(int i=0;i<vertex;i++)
    {
        DFS.clear();
        visit.clear();
    cout<<"\nDFS Traversal for root as "<< i << "-->";
    dfs(vertex,adj,DFS,i,visit);


    for(auto x:DFS) cout<<x<<" ";


    }
}
```

# INPUT GRAPH



# OUTPUT

```
enter no. of vertices-->9
enter edges , press -1 -1 to quit
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

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