

## Assignment No:-01

### Parallel BFS(Breath First Search):-

```
#include<iostream>
#include<bits/stdc++.h>
#include<omp.h>
using namespace std;

vector<bool> v;
vector<vector<int>>> g;

void bfsTraversal(int b)
{
    queue<int> q; //Declare a queue to store all the nodes connected to b
    q.push(b); //Insert b to queue
    v[b]=true; //mark b as visited

    cout<<"\nThe BFS Traversal is: ";

    double start=omp_get_wtime();
    while(!q.empty())
    {
        int a = q.front();
        q.pop(); //delete the first element form queue
        #pragma omp parallel
        for(auto j=g[a].begin();j!=g[a].end();j++)
        {
            if (!v[*j])
            {
                v[*j] = true;
                q.push(*j);
            }
        }
        cout<<a<<" ";
    }
    double end=omp_get_wtime();
    double time=end-start;
    cout<<"\n\nTime taken => "<<time<<endl;
}

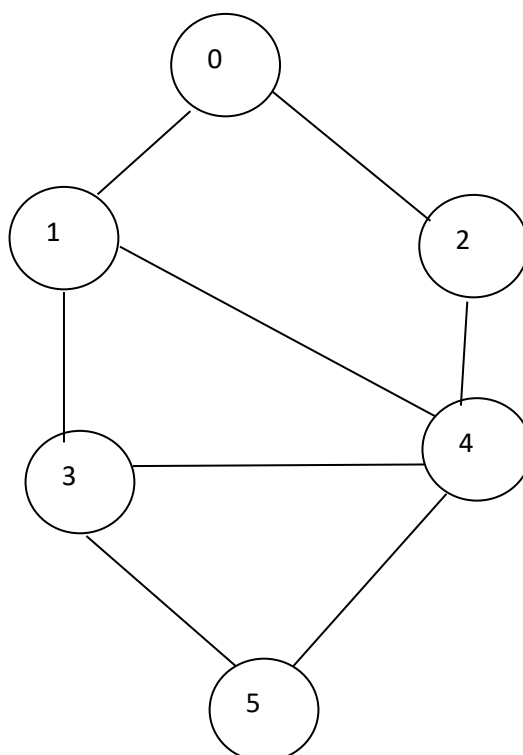
void makeEdge(int a, int b)
{
    g[a].push_back(b); //an edge from a to b (directed graph)
}

int main()
```

```

{
    omp_set_num_threads(4);
    int n,e;
    cout<<"Consider first vertex => 0"<<endl;
    cout<<"\nEnter the number of vertices: ";
    cin >> n;
    cout<<"\nEnter the number of edges: ";
    cin>>e;
    v.assign(n, false);
    g.assign(n, vector<int>());
    int a, b, i;
    cout << "\nEnter the edges with source and target vetex: "<<endl;
    for(i=0;i<e;i++)
    {
        cin>>a>>b;
        makeEdge(a, b);
    }
    for (i=0;i<n;i++)
    {
        if (!v[i]) //if the node i is unvisited
        {
            bfsTraversal(i);
        }
    }
    return 0;
}

```



## **OUTPUT:-**

**Consider first vertex => 0**

**Enter the number of vertices: 6**

**Enter the number of edges: 8**

**Enter the edges with source and target vetex:**

**0 1**

**0 2**

**1 3**

**1 4**

**2 4**

**3 5**

**4 5**

**3 4**

**The BFS Traversal is: 0 1 2 3 4 5**

**Time taken => 0.017**

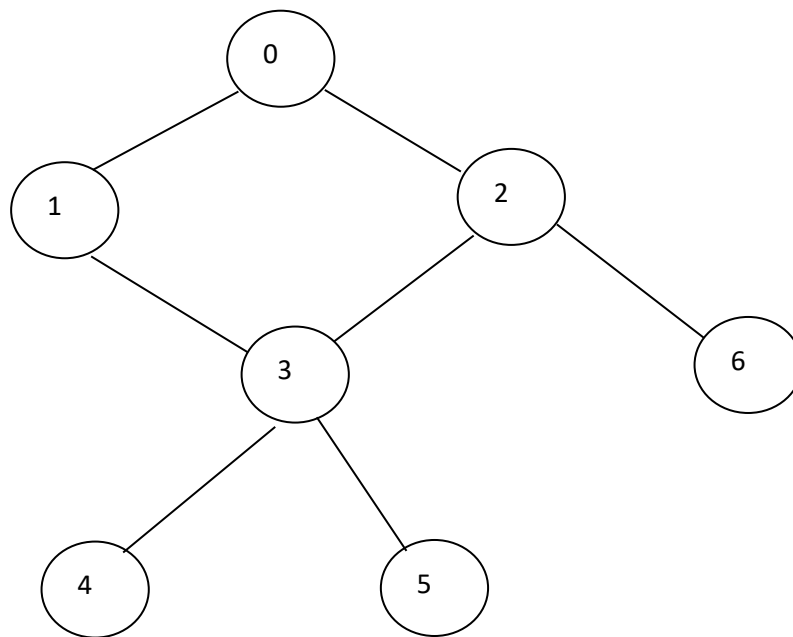
### Parallel DFS(Depth First Search):-

```
#include<bits/stdc++.h>
#include<omp.h>
using namespace std;
class Graph {
public:
map<int, bool> visited;
map<int, list<int> > adj;
// function to add an edge to graph
void addEdge(int v, int w);
// DFS traversal of the vertices reachable from v
void DFS(int v);
};
void Graph::addEdge(int v, int w)
{
adj[v].push_back(w); // Add w to v's list.
}
void Graph::DFS(int v)
{
#pragma omp parallel
// Mark the current node as visited and print it
visited[v] = true;
cout<<v<<" ";
list<int>::iterator i; // Recur for all the vertices adjacent to this vertex
for(i=adj[v].begin();i!=adj[v].end();++i)
{
if(!visited[*i])
DFS(*i);
}
}
int main()
{
omp_set_num_threads(4);
```

```

int z;
Graph g;
g.addEdge(0,1);
g.addEdge(0,2);
g.addEdge(1,3);
g.addEdge(2,3);
g.addEdge(3,4);
g.addEdge(3,5);
g.addEdge(2,6);
cout<<"Enter the vertex to start the DFS traversal with: "<<endl;
cin>>z;
cout<<"\nDepth First Traversal: \n";
g.DFS(z);
cout<<endl;
return 0;
}

```



### OUTPUT:-

Enter the vertex to start the DFS traversal with:

0

Depth First Traversal:

0 1 3 4 5 2 6