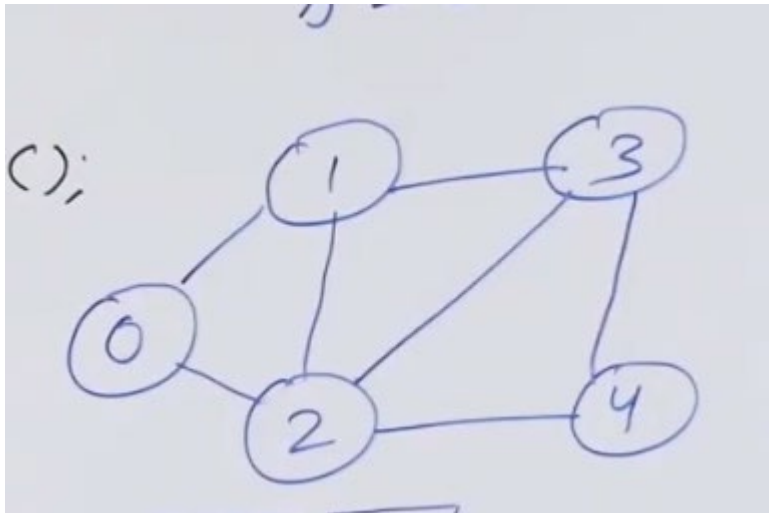


## BREATH FIRST SEARCH IN GRAPH:



Create this graph and perform bfs

```
1  /*graph creation using adjacency list for a undirected graph*/
2  #include<bits/stdc++.h>
3  using namespace std;
4  typedef long long int lli;
5  addedge(vector<int>a[],int u,int v)
6  {
7      a[u].push_back(v);
8      a[v].push_back(u);
9  }
10 printgraph(vector<int>a[],int noofvertex )
11 {
12     cout<<"the adjacency list is:\n";
13     for(int i=0;i<noofvertex;i++)
14     {
15         for(int j=0;j<a[i].size();j++)
16             cout<<a[i][j]<<" ";
17         cout<<endl;
18     }
19 }
20 bfs(vector<int>a[],int noofvertex,int s)
21 {
22     bool visited[noofvertex+1];//visited array to track we are not visiting the same node again and again
23     for(int i=0;i<noofvertex;i++)
24         visited[i]=false;
25     queue<int>q;
26     q.push(s);
27     visited[s]=true;
28     cout<<"doing bfs:\n";
29     while(!q.empty())
```

```

29 while(!q.empty())
30 {
31     int k=q.front();
32     q.pop();
33     cout<<k<<" ";
34     for(int x:a[k])
35     {
36         if(visited[x]==false)
37         {
38             visited[x]=true;
39             q.push(x);
40         }
41     }
42 }
43 }
44 }
45 int main()
46 {
47     int noofvertex=5;
48     vector<int>a[5]; //here a is the adjacency list
49     addedge(a,0,1);
50     addedge(a,0,2);
51     addedge(a,1,2);
52     addedge(a,1,3);
53     addedge(a,2,3);
54     addedge(a,2,4);
55     addedge(a,3,4);
56     printgraph(a,noofvertex);
57     int sourcevertex=0;//start bfs from the source vertex
58     bfs(a,noofvertex,sourcevertex);
59     return 0;
60 }
61

```

/\*graph creation using adjacency list for a undirected graph\*/

#include<bits/stdc++.h>

using namespace std;

typedef long long int lli;

addege(vector<int>a[],int u,int v)

{

a[u].push\_back(v);

a[v].push\_back(u);

}

printgraph(vector<int>a[],int noofvertex )

{

```

        cout<<"the adjacency list is:\n";
        for(int i=0;i<noofvertex;i++)
        {
            for(int j=0;j<a[i].size();j++)
                cout<<a[i][j]<<" ";
            cout<<endl;
        }
    }

    bfs(vector<int>a[],int noofvertex,int s)
    {
        bool visited[noofvertex+1];//visited array to track we are not visiting
        the same node again and again
        for(int i=0;i<noofvertex;i++)
            visited[i]=false;
        queue<int>q;
        q.push(s);
        visited[s]=true;
        cout<<"doing bfs:\n";
        while(!q.empty())
        {
            int k=q.front();
            q.pop();
            cout<<k<<" ";
            for(int x:a[k])
            {

```

```

        if(visited[x]==false)
        {
            visited[x]=true;
            q.push(x);
        }

    }
}

int main()
{
    int noofvertex=5;
    vector<int>a[5]; //here a is the adjacency list
    addedge(a,0,1);
    addedge(a,0,2);
    addedge(a,1,2);
    addedge(a,1,3);
    addedge(a,2,3);
    addedge(a,2,4);
    addedge(a,3,4);
    printgraph(a,noofvertex);
    int sourcevertex=0;//start bfs from the source vertex
    bfs(a,noofvertex,sourcevertex);
    return 0;
}

```