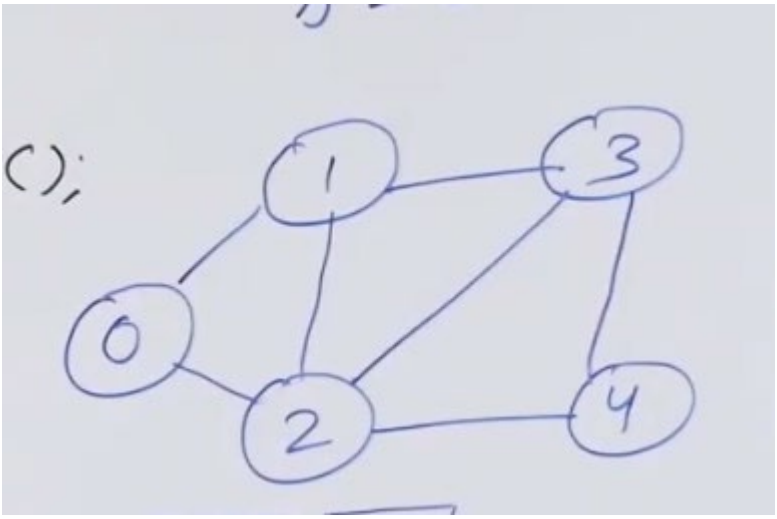
BREATH FIRST SEARCH IN GRAPH:



Create this graph and perform bfs

```
 /*graph creation using adjacency list for a undirected graph*/
#include<bits/stdc++.h>
using namespace std;
typedef long long int lli;
addedge(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())
```

```
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;
}

```

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

#include<bits/stdc++.h>

using namespace std;

typedef long long int lli;

addedge(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;
}
```