**Breadth First Search or BFS for a Graph**

Breadth First Traversal (or Search) for a graph is similar to Breadth First Traversal of a tree (See method 2 of this post). The only catch here is, unlike trees, graphs may contain cycles, so we may come to the same node again. To avoid processing a node more than once, we use a boolean visited array. For simplicity, it is assumed that all vertices are reachable from the starting vertex.

For example, in the following graph, we start traversal from vertex 2. When we come to vertex 0, we look for all adjacent vertices of it. 2 is also an adjacent vertex of 0. If we don’t mark visited vertices, then 2 will be processed again and it will become a non-terminating process. A Breadth First Traversal of the following graph is 2, 0, 3, 1.



**Following are the implementations of simple Breadth First Traversal from a given source.**

**The implementation uses adjacency list representation of graphs.**

code :

#include<stdio.h>

int main()

{

int v,e,i,j,v1,v2,k=-1,root;

printf("enter no of (vertices,edges) : ");

scanf("%d%d",&v,&e);

printf("enter edges( , ) between (0-%d) : ",v-1);

int visited[v]={0},queued[v],A[v][v];

for(i=0;i<e;i++)

{

scanf("%d%d",&v1,&v2);

A[v1][v2]=1;

}

printf("select root node :");

scanf("%d",&root);

visited[root]=1;

queued[++k]=root;

for(j=0;j<=k;j++)

{

for(i=0;i<v;i++)

{

if(A[queued[j]][i]==1&&visited[i]==0)

{

queued[++k]=i;

visited[i]=1;

}

}

printf("%d ",queued[j]);

}

return 0;

}