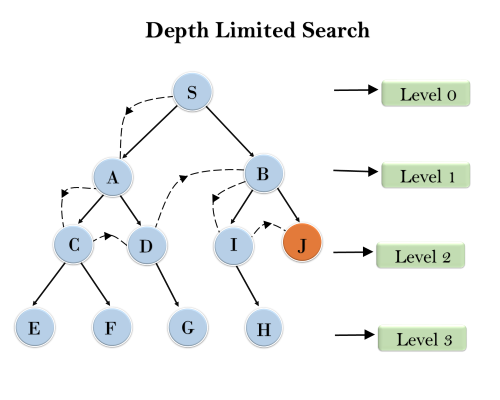
**Depth-Limited Search Algorithm:**

A depth-limited search algorithm is similar to depth-first search with a predetermined limit. Depth-limited search can solve the drawback of the infinite path in the Depth-first search. In this algorithm, the node at the depth limit will treat as it has no successor nodes further.

Depth-limited search can be terminated with two Conditions of failure:

* Standard failure value: It indicates that problem does not have any solution.
* Cutoff failure value: It defines no solution for the problem within a given depth limit.

Example:



**Completeness:** DLS search algorithm is complete if the solution is above the depth-limit.

**Time Complexity:** Time complexity of DLS algorithm is **O(bℓ)**.

**Space Complexity:** Space complexity of DLS algorithm is O**(b×ℓ)**.

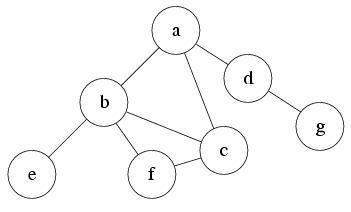
**Optimal:** Depth-limited search can be viewed as a special case of DFS, and it is also not optimal even if ℓ>d.

## Depth-Limited Search (Example)

The depth-limited search (DLS) method is almost equal to depth-first search (DFS), but DLS can work on the infinite state space problem because it bounds the depth of the search tree with a predetermined limit L. Nodes at this depth limit are treated as if they had no successors.

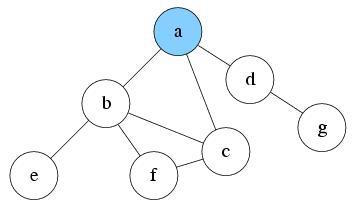
Now use the example in DFS to see what will happen if we use DLS with .

Below is the graph we will traverse. Same as DFS, we use the stack data structure S1 to record the node we’ve explored. Suppose the source node is node a.



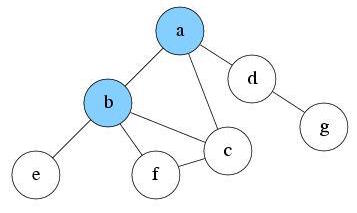
###### S1:

At first, the only reachable node is a. So push it into S1 and mark as visited. Current level is 0.



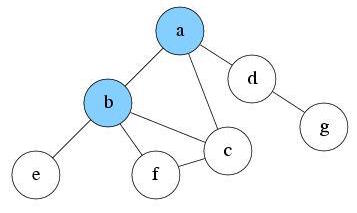
###### S1: a

After exploring a, now there are three nodes reachable: node b, c and d. Suppose we pick node b to explore first. Push b into S1 and mark it as visited. Current level is 1.



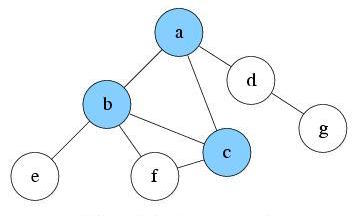
###### S1: b, a

Since current level is already the max depth L. Node b will be treated as having no successor. So there is nothing reachable. Pop b from S1. Current level is 0.



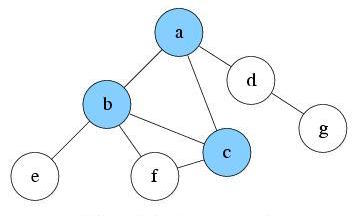
###### S1: a

Explore a again. There are two unvisited nodes c and d that are reachable. Suppose we pick node c to explore first. Push c into S1 and mark it as visited. Current level is 1.



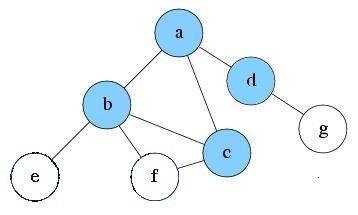
###### S1: c, a

Since current level is already the max depth L. Node c will be treated as having no successor. So there is nothing reachable. Pop c from S1. Current level is 0.



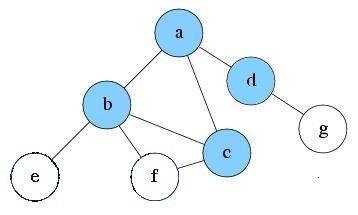
###### S1: a

Explore a again. There is only one unvisited node d reachable. Push d into S1 and mark it as visited. Current level is 1.



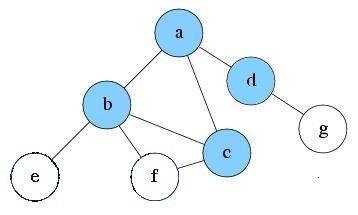
###### S1: d, a

Explore d and find no new node is reachable. Pop d from S1. Current level is 0.



###### S1: a

Explore a again. No new reachable node. Pop a from S1



###### S1:

S1 is empty now. DLS will be finished.

### Algorithm

This algorithm essentially follows a similar set of steps as in the DFS algorithm.

1. The start node or node 1 is added to the beginning of the stack.
2. Then it is marked as visited, and if node 1 is not the goal node in the search, then we push second node 2 on top of the stack.
3. Next, we mark it as visited and check if node 2 is the goal node or not.
4. If node 2 is not found to be the goal node, then we push node 4 on top of the stack.
5. Now we search in the same depth limit and move along depth-wise to check for the goal nodes.
6. If Node 4 is also not found to be the goal node and depth limit is found to be reached, then we retrace back to nearest nodes that remain unvisited or unexplored.
7. Then we push them into the stack and mark them visited.
8. We continue to perform these steps in iterative ways unless the goal node is reached or until all nodes within depth limit have been explored for the goal.

When we compare the above steps with DFS, we may found that DLS can also be implemented using the queue data structure. In addition to each level of the node needs to be computed to check the finiteness and reach of the goal node from the source node.

Depth-limited search is found to terminate under these two clauses:

1. When the goal node is found to exist.
2. When there is no solution within the given depth limit domain.

#### Algorithm of the example

1. We start with finding and fixing a start node.
2. Then we search along with the depth using the DFS algorithm.
3. Then we keep checking if the current node is the goal node or not.

**If the answer is no:** then we do nothing.

**If the answer is yes:** then we return.

1. Now we will check whether the current node is lying under the depth limit specified earlier or not.

**If the answer is not:** then we do nothing.

**If the answer is yes, we** will explore the node further and save all of its successors into a stack.

1. Now we call the function of DLS iteratively or recursively for all the nodes of the stack and go back to step 2.

Thus we have successfully explored all the nodes in the given depth limit and found the goal node if it exists within a specified depth limit.

#### C program

**Code:**

#include <stdio.h>

#include <stdlib.h>

/\*ADJACENCY MATRIX\*/

int source,X,Y,time,visited[20],Z[20][20];

void DFS(int p)

{

int q;

visited[p]=1;

printf(" %d->",p+1);

for(q=0;q<X;q++)

{

if(Z[p][q]==1&&visited[q]==0)

DFS(q);

}

}

int main()

{

int p,q,x1,x2;

printf("\t\t\tGraphs\n");

printf("Enter the required number of edges:");

scanf("%d",&Y);

printf("Enter the required number of vertices:");

scanf("%d",&X);

for(p=0;p<X;p++)

{

for(q=0;q<X;q++)

Z[p][q]=0;

}

/\*creating edges : \*/

for(p=0;p<Y;p++)

{

printf("Enter the format of the edges (format: x1 x2) : ");

scanf("%d%d",&x1,&x2);

Z[x1-1][x2-1]=1;

}

for(p=0;p<X;p++)

{

for(q=0;q<X;q++)

printf(" %d ",Z[p][q]);

printf("\n");

}

printf("Enter the source of the DFS: ");

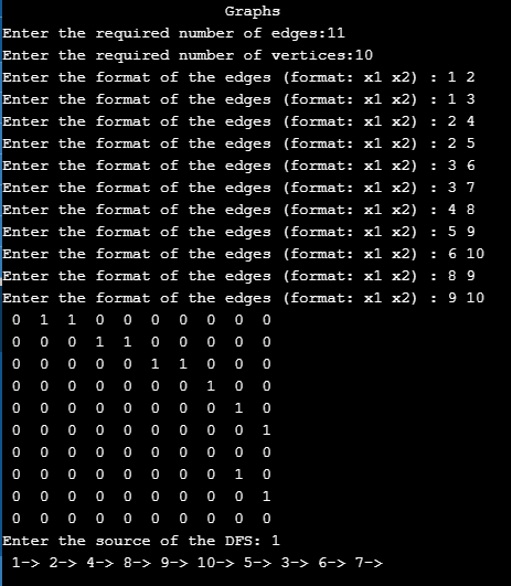
scanf("%d",&source);

DFS(source-1);

return 0;

}

**Output:**



### Advantages and Disadvantages of Depth Limited Search

Below are the advantages and disadvantages are below:

#### Advantages of Depth Limited Search

* Depth limited search is better than DFS and requires less time and memory space.
* DFS assures that the solution will be found if it exists infinite time.
* There are applications of DLS in graph theory particularly similar to the DFS.
* To combat the disadvantages of DFS, we add a limit to the depth, and our search strategy performs recursively down the search tree.

#### Disadvantages of Depth Limited Search

* The depth limit is compulsory for this algorithm to execute.
* The goal node may not exist in the depth limit set earlier, which will push the user to iterate further adding execution time.
* The goal node will not be found if it does not exist in the desired limit.

### Performance Measures

* **Completeness:** The DLS is a complete algorithm in general except the case when the goal node is the shallowest node, and it is beyond the depth limit, i.e. l < d, and in this case, we never reach the goal node.
* **Optimality:** The DLS is a non-optimal algorithm since the depth that is chosen can be greater than d (l>d). Thus DLS is not optimal if l > d
* **Time complexity is expressed as**: It is similar to the DFS, i.e. O(bl), where 1 is the set depth limit.
* **Space Complexity is expressed as:** It is similar to DFSe. O(bl), where 1 is specified depth limit.

**Properties of depth-limited search**

Although DLS is same as DFS with a depth limit, its property is different from DFS’. First, let’s assume the given graph is a tree and:

* **Branching factor** is B
* The depth of the solution is D
* The maximum depth of the tree is Dm

First of all DLS is **not complete**i n general. If the optimal solution node is still deeper than the depth limit L, DLS will never find the solution. Since DLS is not complete, it is **not optimal**, too.

The time complexity of DLS is which is much better than DFS’ if L is much smaller than . The space complexity of DLS is which is linear because same as DFS, DLS only needs to record the route it currently searching on. Compare to DFS, DLS is doing better on time complexity. However, DLS may also be unable to find the solution. If a search algorithm can’t find the solution even though the solution exists, then that algorithm is very unlikely to be preferred. However, understanding DLS is crucial to understanding and dealing with the disadvantages of BFS, and leads to an algorithm called **iterative deepening search**.

### Conclusion – Depth Limited Search

DLS algorithm is used when we know the search domain, and there exists a prior knowledge of the problem and its domain while this is not the case for uninformed search strategy. Typically, we have little idea of the goal node’s depth unless one has tried to solve it before and has found the solution.