Detecting cycles in a graph using DFS: 2 different approaches and what's the difference

Asked 7 years ago Active 1 year, 8 months ago Viewed 80k times

43

35

Note that a graph is represented as an adjacency list.

I've heard of 2 approaches to find a cycle in a graph:

1. Keep an array of boolean values to keep track of whether you visited a node before. If you run out of new nodes to go to (without hitting a node you have already been), then just backtrack and try a different branch.
2. The one from Cormen's CLRS or Skiena: For depth-first search in undirected graphs, there are two types of edges, tree and back. The graph has a cycle if and only if there exists a back edge.

Can somebody explain what are the back edges of a graph and what's the diffirence between the above 2 methods.

Thanks.

Update: Here's the code to detect cycles in both cases. Graph is a simple class that represents all graph-nodes as unique numbers for simplicity, each node has its adjacent neighboring nodes (g.getAdjacentNodes(int)):

```
public class Graph {

  private int[][] nodes; // all nodes; e.g. int[][] nodes = {{1,2,3}, {3,2,1,5,6}...};

  public int[] getAdjacentNodes(int v) {
    return nodes[v];
  }

  // number of vertices in a graph
  public int vSize() {
    return nodes.length;
  }

}
```

Java code to detect cycles in an undirected graph:

```
    public class DFSCycle {

    private boolean marked[];
    private int s;
    private Graph g;
    private boolean hasCycle;

    // s - starting node
    public DFSCycle(Graph g, int s) {
        this.g = g;
        this.s = s;
        marked = new boolean[g.vSize()];
        findCycle(g,s,s);
    }

    public boolean hasCycle() {
        return hasCycle;
    }

    public void findCycle(Graph g, int v, int u) {

        marked[v] = true;

        for (int w : g.getAdjacentNodes(v)) {
            if(!marked[w]) {
                marked[w] = true;
                findCycle(g,w,v);
            } else if (v != u) {
                hasCycle = true;
                return;
            }
        }

    }
}
```

Java code to detect cycles in a directed graph:

```
public class DFSDirectedCycle {

    private boolean marked[];
    private boolean onStack[];
    private int s;
    private Graph g;
    private boolean hasCycle;

    public DFSDirectedCycle(Graph g, int s) {
        this.s = s
        this.g = g;
        marked = new boolean[g.vSize()];
        onStack = new boolean[g.vSize()];
        findCycle(g,s);
    }

    public boolean hasCycle() {
        return hasCycle;
    }

    public void findCycle(Graph g, int v) {

        marked[v] = true;
        onStack[v] = true;

        for (int w : g.adjacentNodes(v)) {
            if(!marked[w]) {
                findCycle(g,w);
            } else if (onStack[w]) {
                hasCycle = true;
                return;
            }
        }

        onStack[v] = false;
    }
}
```

graph cycle depth-first-search adjacency-list Edit tags

edited Nov 2 '17 at 5:44
connect2krish
153 2 9

asked Oct 1 '13 at 9:55
Ivan Voroshilin
4,265 3 25 54

I think there is a mistake in the ordered graph code. In case of traversing a graph in the shape of O, with the root on the top and with all the edges directed to bottom, this algorithm will detect cycle (firstly will traverse the left side of O to bottom and mark all nodes as marked, then the right part of O until I will get to bottom, which is already marked). I would add there marked[v]= false; just after findCycle(g,w); What do you think? – Matej Briškár Apr 6 '15 at 10:41

In `DFSCycle` the condition to check for cycle is incorrect. It should be `if(i != u) // If an adjacent is visited and not parent of current vertex, then there is a cycle. { hasCycle = true; return; }` – akhil_mittal Jan 14 '18 at 9:58

6 Answers

Active Oldest Votes

0

I think the above code works only for a connected digraph since we start dfs from the source node only, for if the digraph is not connected there may be a cycle in the other component which may go unnoticed!

answered Jan 15 '19 at 7:15
Shubham Chaudhary
23 5

I don't know why this comment get downvoted, but he did give an important point, if a digraph is not strongly connected, i.e., there exists some vertices that are not reachable from other vertices, a DFS search for a cycle might fail as the graph might be separated into multiple SCCs, refer stackoverflow.com/a/32893154/10661805 for finding cycles in a digraph. – Holmes Queen May 29 at 17:00

-1

This post is hidden. It was deleted 3 years ago by Aaron Hall ♦.

So what would be in the main method for the waited graph

answered Jul 29 '17 at 20:09
TimothyW553
57 5

This is not an answer. You should post your question as a comment to the original question. – Postlagerkarte Jul 29 '17 at 20:28

comments disabled on deleted / locked posts / reviews

0

This post is hidden. It was deleted 5 years ago by Matt.

Here is the code I've written in C based on DFS to find out whether a given **directed graph** is cyclic or not,i.e is there any back edge or not. with some sample output at the end. Hope it'll be helpful :)

```
#include<stdio.h>
#include<stdlib.h>

/****Global Variables****/
int A[20][20],visited[20],count=0,n;
int seq[20],s,stack[20],ptr;

/****DFS Function Declaration****/
void DFS();

/****DFSearch Function Declaration****/
void DFSearch(int cur);

/****Main Function****/
int main()
   {
    int i,j;

    printf("\nEnter no of Vertices: ");
    scanf("%d",&n);

    printf("\nEnter the Adjacency Matrix(1/0):\n");
    for(i=1;i<=n;i++)
        for(j=1;j<=n;j++)
            scanf("%d",&A[i][j]);

    DFS();

    printf("\nGraph is DAG. No back edge!\n");

    printf("\n\n");
    return 0;
   }

/****DFS Function Definition****/
void DFS()
    {
      int i;
      for(i=1;i<=n;i++)
```

Sample Output:

```
majid@majid-K53SC:~./a.out

Enter no of Vertices: 5

Enter the Adjacency Matrix(1/0):
0 0 1 0 0
0 0 1 0 1
0 0 0 1 1
0 0 0 0 1
0 0 0 0 0

Graph is DAG. No back edge!


majid@majid-K53SC:~./a.out

Enter no of Vertices: 5

Enter the Adjacency Matrix(1/0):
0 0 1 0 0
0 0 1 0 0
0 0 0 1 1
0 0 0 0 1
0 1 0 0 0

Graph is not a DAG. There is back edge!
```

answered May 17 '15 at 17:51
Majid NK
191 1 1 13

Please do not duplicate your posts, and please do not beg for upvotes. Both are noise. – Matt Aug 6 '15 at 11:01

comments disabled on deleted / locked posts / reviews

1

Here is the code I've written in C based on DFS to find out whether a given **undirected graph** is connected/cyclic or not. with some sample output at the end. Hope it'll be helpful :)

```
#include<stdio.h>
#include<stdlib.h>

/****Global Variables****/
```

```
/****DFS Function Declaration****/
void DFS();

/****DFSearch Function Declaration****/
void DFSearch(int cur);

/****Main Function****/
int main()
   {
    int i,j;

    printf("\nEnter no of Vertices: ");
    scanf("%d",&n);

    printf("\nEnter the Adjacency Matrix(1/0):\n");
    for(i=1;i<=n;i++)
            for(j=1;j<=n;j++)
        scanf("%d",&A[i][j]);

    printf("\nThe Depth First Search Traversal:\n");

    DFS();

    for(i=1;i<=n;i++)
        printf("%c,%d\t",'a'+seq[i]-1,i);

    if(connected && acyclic)    printf("\n\nIt is a Connected, Acyclic Graph!");
    if(!connected && acyclic)   printf("\n\nIt is a Not-Connected, Acyclic Graph!");
    if(connected && !acyclic)   printf("\n\nGraph is a Connected, Cyclic Graph!");
    if(!connected && !acyclic)  printf("\n\nIt is a Not-Connected, Cyclic Graph!");
```

Sample Output:

```
majid@majid-K53SC:~/Desktop$ gcc BFS.c

majid@majid-K53SC:~/Desktop$ ./a.out
************************************

Enter no of Vertices: 10

Enter the Adjacency Matrix(1/0):

0 0 1 1 1 0 0 0 0 0
0 0 0 0 1 0 0 0 0 0
0 0 0 1 0 1 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 1 0 0 1 0 0 0 0 0
0 0 0 0 0 0 0 1 0 0
0 0 0 0 0 0 0 0 1 0
0 0 0 0 0 0 0 0 0 1
0 0 0 0 0 0 1 0 0 0

The Depdth First Search Traversal:
a,1 c,2 d,3 f,4 b,5 e,6 g,7 h,8 i,9 j,10

It is a Not-Connected, Cyclic Graph!


majid@majid-K53SC:~/Desktop$ ./a.out
************************************

Enter no of Vertices: 4

Enter the Adjacency Matrix(1/0):
0 0 1 1
0 0 1 0
1 1 0 0
0 0 0 1

The Depth First Search Traversal:
a,1 c,2 b,3 d,4
```

edited May 17 '15 at 17:22

answered May 17 '15 at 0:42
Majid NK
191 1 1 13

10

For the sake of completion, it is possible to find cycles in a directed graph using DFS (from wikipedia):

```
 L ← Empty list that will contain the sorted nodes
while there are unmarked nodes do
    select an unmarked node n
    visit(n)
function visit(node n)
    if n has a temporary mark then stop (not a DAG)
    if n is not marked (i.e. has not been visited yet) then
        mark n temporarily
        for each node m with an edge from n to m do
            visit(m)
        mark n permanently
        unmark n temporarily
        add n to head of L
```

edited Mar 10 '15 at 16:13

answered Dec 18 '14 at 17:45
Igor Zelaya
3,977 4 32 49

60

✓

Answering my question:

The graph has a cycle if and only if there exists a back edge. A back edge is an edge that is from a node to itself (selfloop) or one of its ancestor in the tree produced by DFS forming a cycle.

Both approaches above actually mean the same. However, this method can be applied only to **undirected graphs**.

The reason why this algorithm doesn't work for directed graphs is that in a directed graph 2 different paths to the same vertex don't make a cycle. For example: A-->B, B-->C, A-->C - don't make a cycle whereas in undirected ones: A--B, B--C, C--A does.

Find a cycle in undirected graphs

An undirected graph has a cycle if and only if a depth-first search (DFS) finds an edge that points to an already-visited vertex (a back edge).

Find a cycle in directed graphs

In addition to visited vertices we need to keep track of vertices currently in recursion stack of function for DFS traversal. If we reach a vertex that is already in the recursion stack, then there is a cycle in the tree.

Update: Working code is in the question section above.

edited Nov 13 '14 at 8:53

answered Oct 1 '13 at 11:34
Ivan Voroshilin
4,265 3 25 54

11 In undirected graph there exists a cycle only if there is a back edge excluding the parent of the edge from where the back edge is found.Else every undirected graph has a cycle by default if we don't exclude the parent edge when finding a back edge. – crackerplace Jan 11 '15 at 16:51

"If we reach a vertex that is already in the recursion stack" I have a confusion for this part for finding a cycle in directed graphs. Your 1st line says in addition to visited vertices we have to consider recursion stack. But next line you said just about the recursion stack. Can you clarify the confusion? – avijit bhattacharjee Aug 24 at 5:32