Depth-first search (DFS) code in python

Asked 3 years, 5 months ago Active 2 months ago Viewed 69k times

8

3

Can you please let me know what is incorrect in below DFS code. It's giving correct result AFAIK, but I don't know when it will fail.

```
graph1 = {
    'A' : ['B','S'],
    'B' : ['A'],
    'C' : ['D','E','F','S'],
    'D' : ['C'],
    'E' : ['C','H'],
    'F' : ['C','G'],
    'G' : ['F','S'],
    'H' : ['E','G'],
    'S' : ['A','C','G']
}

visited = []

def dfs(graph,node):
    global visited
    if node not in visited:
        visited.append(node)
        for n in graph[node]:
            dfs(graph,n)

    dfs(graph1,'A')
    print(visited)
```

Output:

```
['A', 'B', 'S', 'C', 'D', 'E', 'H', 'G', 'F']
```

python depth-first-search Edit tags

edited May 9 '17 at 16:13
Juan Leni
4,835 2 39 61

asked Apr 15 '17 at 19:23
Vicky
141 1 1 5

2 First of all: don't use `global` s avoid them as much as possible!! – Willem Van Onsem Apr 15 '17 at 19:24

1 DFS is a *search* algorithm, but there is *no target* defined you are looking for... – Willem Van Onsem Apr 15 '17 at 19:28

Thanks for the response.. visited = [] def dfs(graph,node,visited): if node not in visited: visited.append(node) for n in graph[node]: dfs(graph,n,visited) dfs(graph1,'A',visited) print(visited) – Vicky Apr 15 '17 at 19:29

I'm also not convinced that the output you show is generated by the data you show - I don't see where S, H and G come from? (But I could possibly be wrong here) – DavidW Apr 15 '17 at 19:32

Hi Willem, if you checkout the link eddmann.com/posts/... there a sample code for the DFS is given but If I change the graph as given in youtube tutorial "youtu.be/iaBEKo5sM7w" I am not getting the result as mentioned. So I thought of writing my own version based on the tutorial. – Vicky Apr 15 '17 at 19:34

|

8 Answers

Active Oldest Votes

0

Here is a more versatile algorithm, the one asked in the question works only for undirected graphs. But this hopefully works for both them.
Check it out

```
graph1= {
'A' : ['B','S'],
'B' : [],
'C' : ['E','S'],
'D' : ['C'],
'E' : ['H'],
'F' : ['C'],
'G' : ['F','S'],
'H' : ['G'],
'S' : []
}
visited = []

def dfs_visit(graph, s):
    global visited
    for v in graph[s]:
        if v not in visited:
            visited.append(v)
            dfs_visit(graph, v)


def dfs(graph):
    global visited
    for v in [*graph]:
        if v not in visited:
            visited.append(v)
            dfs_visit(graph,v)

dfs(graph1)
print(visited)
```

answered Jul 2 at 13:51
Ramesses2
1 3

2

Here's an iterative (non-recursive) implementation of a DFS:

```
def dfs_iterative(graph, start_vertex):
    visited = set()
    traversal = []
    stack = [start_vertex]
    while stack:
        vertex = stack.pop()
        if vertex not in visited:
            visited.add(vertex)
            traversal.append(vertex)
            stack.extend(reversed(graph[vertex]))   # add vertex in the same order as
visited
    return traversal

test_graph = {
    'A' : ['B','S'],
    'B' : ['A'],
    'C' : ['D','E','F','S'],
    'D' : ['C'],
    'E' : ['C','H'],
    'F' : ['C','G'],
    'G' : ['F','S'],
```

```
print(dfs_iterative(test_graph, 'A'))
```

Output:

```
['A', 'B', 'S', 'C', 'D', 'E', 'H', 'G', 'F']
```

answered Mar 15 at 22:38
Saurabh
431 3 15

0

This post is hidden. It was deleted 7 months ago by the post author.

Non recursive way.

```
def dfs(graph, node):
    visited = [node]
    stack = [node]
    while stack:
        tmp = stack.pop()
        print(tmp)
        for adj in graph[tmp]:
            if adj not in visited:
                visited.append(adj)
                stack.append(adj)

graph1 = {
    'A' : ['B','S'],
    'B' : ['A'],
    'C' : ['D','E','F','S'],
    'D' : ['C'],
    'E' : ['C','H'],
    'F' : ['C','G'],
    'G' : ['F','S'],
    'H' : ['E','G'],
    'S' : ['A','C','G']
}

dfs(graph2, 'A')
```

answered Mar 8 at 21:16
Sukeshini
1,081 2 20 46

comments disabled on deleted / locked posts / reviews

7

Without recursion:

```
def dfs(graph, node):
    visited = [node]
    stack = [node]
    while stack:
        node = stack[-1]
        if node not in visited:
            visited.extend(node)
        remove_from_stack = True
        for next in graph[node]:
            if next not in visited:
                stack.extend(next)
                remove_from_stack = False
                break
        if remove_from_stack:
            stack.pop()
    return visited

print (dfs(graph1, 'A'))
```

Output:

```
['A', 'B', 'S', 'C', 'D', 'E', 'H', 'G', 'F']
```

edited Nov 25 '19 at 10:00
Stan James
2,177 23 34

answered Aug 15 '18 at 20:24
gaetano
592 9 19

2 Thanks for te non recursive version - checked and works – Mercury May 21 '19 at 7:17

0

DFS implementation in Python

```
from collections import defaultdict

class Graph:
    def __init__(self):
        self.graph = defaultdict(list)

    def addEdge(self, u, v):
        self.graph[u].append(v)

    def DFSUtil(self, v, visited):
        visited[v]=True
        print(v)

        for i in self.graph[v]:
            if visited[i] == False:
                self.DFSUtil(i, visited)

    def DFS(self):
        V = len(self.graph)

        visited = [False]*(V)

        for i in range(V):
            if visited[i] == False:
                self.DFSUtil(i, visited)

# Driver code
# Create a graph given in the above diagram
g = Graph()
g.addEdge(0, 1)
g.addEdge(0, 2)
g.addEdge(1, 2)
g.addEdge(2, 0)
g.addEdge(2, 3)
g.addEdge(3, 3)

print("Following is Depth First Traversal")
g.DFS()
```

edited Feb 4 '19 at 11:44
Community ♦
1 1

answered Jan 12 '18 at 5:01
Akash Kandpal
1,758 16 23

@AssafLivne why not? Would you mind to expand a bit on that? – monkey intern Nov 6 '19 at 7:24

0

```
from collections import defaultdict

class Graph:
    def __init__(self):
        self.graph = defaultdict(list)

    def addEdge(self,u,v):
        self.graph[u].append(v)

    def DFS(self,v,vertex):
        visited = [False]*vertex
        print(self. graph)
        # print(len(self.graph),"+++")
        self.DFSUtil(v,visited)

    def DFSUtil(self,v,visited):
        visited[v]=True
        print(v)

        for i in self.graph[v]:
            if visited[i] == False:
                # print(visited)
                self.DFSUtil(i,visited)
g= Graph()

vertex=7

g.addEdge(0,1)
g.addEdge(0,2)
g.addEdge(0,6)
g.addEdge(0,5)
g.addEdge(5,3)
g.addEdge(5,4)
g.addEdge(4,3)
g.addEdge(6,4)

g.DFS(0,vertex)
```

This is the modification for the above code because that doesn't work with in all cases.
We have to specify the number of vectors and then give edges manually.

edited Oct 25 '18 at 11:55
Yuca
4,710 2 15 36

answered Oct 25 '18 at 11:48
Aslesha Ch
16 1

0

```
graph = {'A': ['B', 'C'],
         'B': ['A', 'D', 'E'],
         'C': ['A', 'F'],
         'D': ['B'],
         'E': ['B', 'F'],
         'F': ['C', 'E']}

  def dfs(s,d):
   def dfs_helper(s,d):
       if s == d:
           return True
       if  s in visited :
           return False
       visited.add(s)
       for c in graph[s]:
           dfs_helper(c,d)
       return False
    visited = set()
    return dfs_helper(s,d)
dfs('A','E') ---- True
dfs('A','M') ---- False
```

answered Sep 11 '18 at 19:28
Sneha Mule
1 1

15

I think you have an indentation problem there. Assuming your code looks like this:

```
graph1 = {
    'A' : ['B','S'],
    'B' : ['A'],
    'C' : ['D','E','F','S'],
    'D' : ['C'],
    'E' : ['C','H'],
    'F' : ['C','G'],
    'G' : ['F','S'],
    'H' : ['E','G'],
    'S' : ['A','C','G']
}

visited = []

def dfs(graph,node):
    global visited
    if node not in visited:
        visited.append(node)
        for n in graph[node]:
            dfs(graph,n)

dfs(graph1,'A')
print(visited)
```

I would say a couple of things:

- Avoid globals if you can
- Instead of a list for visited, use a set

plus:

- This will not work for forests but I assume you already know that
- It will fail if you reference a node that does not exist.

Updated code:

```
graph1 = {
    'A' : ['B','S'],
    'B' : ['A'],
    'C' : ['D','E','F','S'],
    'D' : ['C'],
    'E' : ['C','H'],
```

```
    'S' : ['A','C','G']
}

def dfs(graph, node, visited):
    if node not in visited:
        visited.append(node)
        for n in graph[node]:
            dfs(graph,n, visited)
    return visited

visited = dfs(graph1,'A', [])
print(visited)
```

edited Jul 28 '17 at 15:31

answered May 9 '17 at 11:55

Juan Leni
4,835 2 39 61

3 Should the updated code contain the variable `visited` as `dict` or `list` ? – Sudarshan Jul 23 '17 at 5:26

@Sudarshan since it says `visited.append(node)` the variable visited is a list. – oamandawi Mar 2 at 1:27

@Juan Leni or anyone listening: In the for loop of your code, how is variable n not stuck in values between 'B' and 'A' and how come it jumps to 'S' in the third iteration? – MM Khan Apr 25 at 18:58