

SH

hi give me the Best question to test my understanding of dfs



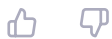
Here's a challenging question to test your understanding of depth-first search (DFS):

Given a directed graph represented by an adjacency matrix and a starting vertex, implement a DFS algorithm to find all the vertices reachable from the starting vertex, and return them in the order they were visited.

Here are some additional considerations to keep in mind:

- The graph may have cycles, so you need to handle cases where you revisit vertices.
- You need to keep track of the vertices you have already visited to avoid visiting them multiple times.
- You need to implement the DFS algorithm using a stack to keep track of the vertices you need to visit next.

If you can successfully implement this question, you should have a solid understanding of DFS and how to implement it using a stack.



SH

ok i will remember this Question .
but before this explain me how to implement dfs using stack



Sure! Here's an overview of how to implement depth-first search (DFS) using a stack:

1. Initialize a stack and push the starting vertex onto the stack.
2. Repeat the following steps until the stack is empty:
 - a. Pop a vertex from the stack and mark it as visited.
 - b. For each unvisited neighbor of the current vertex, push it onto the stack.
3. The DFS algorithm terminates when the stack is empty, at which point all



Regenerate
response

Generate
PNG

Download
PDF

Share
Link



c

 Copy code

```
def DFS(graph, start):  
    stack = [start]  
    visited = set()  
  
    while stack:  
        vertex = stack.pop()  
        if vertex not in visited:  
            visited.add(vertex)  
            for neighbor in graph[vertex]:  
                if neighbor not in visited:  
                    stack.append(neighbor)  
  
    return visited
```

In this implementation, `graph` is represented as an adjacency list, where `graph[vertex]` returns a list of neighbors for the given vertex. The `visited` set is used to keep track of the vertices that have already been visited to avoid visiting them again. The `stack` is used to keep track of the vertices that need to be visited next, and the DFS algorithm terminates when the stack is empty.

