Experiment 1

Foundations of AI

Path traversal using Depth First Search

By- Shivansh Jain,Vit Chennai

AIM:-

To find the path traversed using DFS (Depth First Search) given the graph and the starting node

CONCEPT:-

Depth-first search is an algorithm for traversing or searching tree or graph data structures. The algorithm starts at the root node (selecting some arbitrary node as the root node in the case of a graph) and explores as far as possible along each branch before backtracking. So the basic idea is to start from the root or any arbitrary node and mark the node and move to the adjacent unmarked node and continue this loop until there is no unmarked adjacent node. Then backtrack and check for other unmarked nodes and traverse them. Finally print the nodes in the path.

ALGORITHM:-

1. Created a stack of nodes and visited array.
2. Insert the root in the stack. Run a loop till the stack is not empty.
3. Pop the element from the stack and print the element.
4. For every adjacent and unvisited node of current node, mark the node and insert it in the stack.

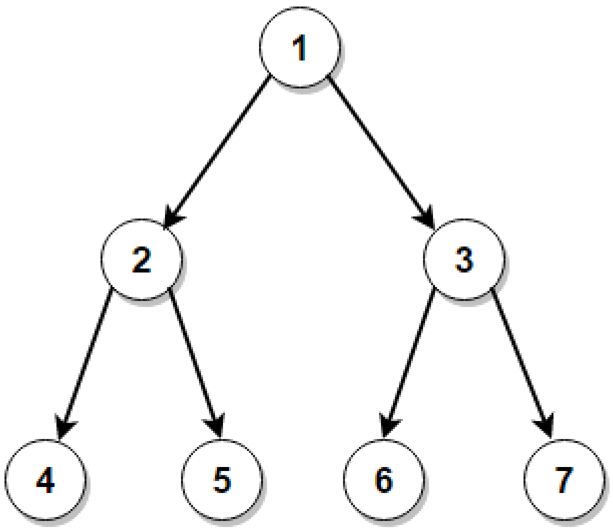
STEP BY STEP IMPLEMENTATION:-

```
#R version 4.1.0
#RStudio version 1.4.1717

rm(list = ls())
#to ensure a clean environment before executing the code

dfs <-function(graph,start) {
# function dfs with arguments graph and start
# graph is an adjacency-matrix-representation of the graph where (x,y) is TRUE if there is an edge
between nodes x and y
# start the node to start from.
# returns an array containing te path from the given start node till it traverses every node in the graph


  #using a stack to manage the nodes that have yet to be visited, intialized with the start node
  stack = c(start)

  #array path to store the path
  path=c()
  # A boolean array indicating whether we have already visited a node
  # the start node is already visited
  visited = rep(FALSE, nrow(graph))
  visited[start] = TRUE


  # while there are nodes yet to visit
  while(length(stack) > 0){

    # if there is only one element in the stack then that element is set to be the node and the stack is
emptied
    if(length(stack)==1){
      node=stack[1]
      stack=c()
    }

    # if there are more than one elements then the last element in the stack is set to be the node and that
element is removed from the stack
    else{
      node=tail(stack,-(length(stack)-1))
      stack=head(stack,-1)
    }


    # the node is added to the path
    # then we check all the neighbouring elements of the node which are yet to be visited and add them to
the stack
    path = c(path,node)
    for(i in seq_along(graph[node,])) {
      if(graph[node,i] && !visited[i]){
        visited[i] = TRUE
        stack=c(stack,i)
      }
    }
  }
  # return path
  return(path)
}

#driver code

#initialise the graph here
#here 1 is the starting node

path=dfs(graph,1)
cat("Final path: ", path)
```

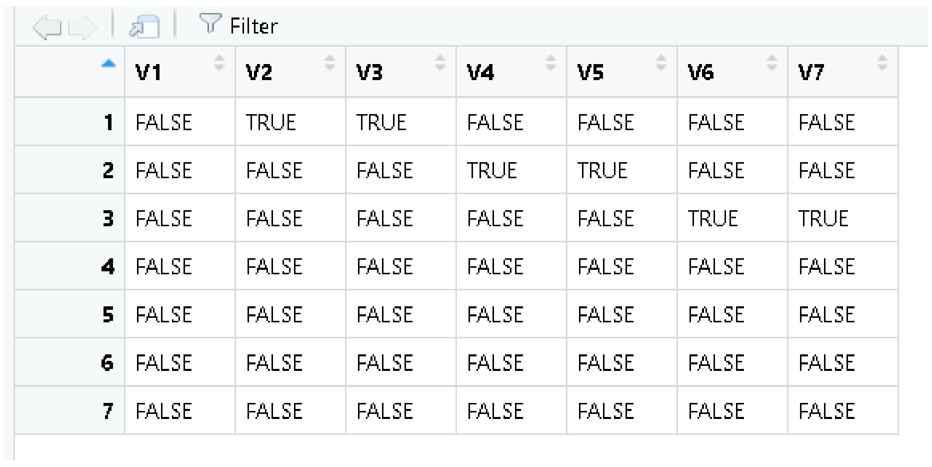
RESULTS AND OUTPUT:-

→

Test case 1:-



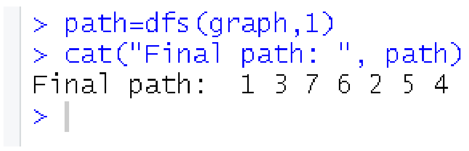
Graph:-



	V1	V2	V3	V4	V5	V6	V7
1	FALSE	TRUE	TRUE	FALSE	FALSE	FALSE	FALSE
2	FALSE	FALSE	FALSE	TRUE	TRUE	FALSE	FALSE
3	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	TRUE
4	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
5	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
6	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
7	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE

Output :-

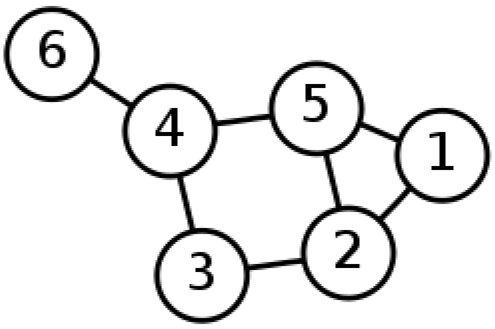
Starting node:- 1



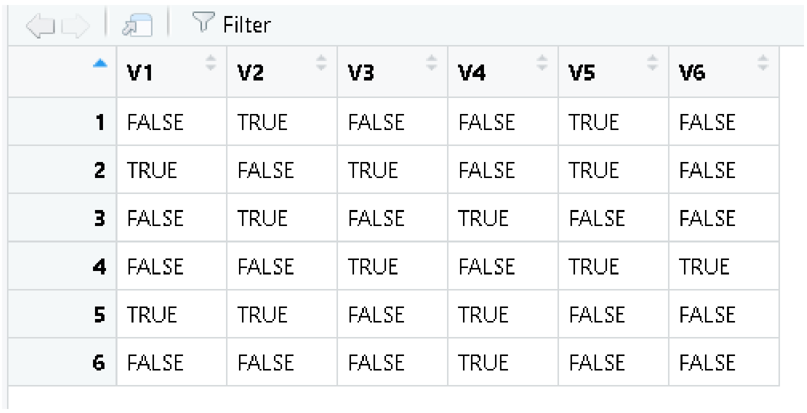
```
> path=dfs(graph,1)
> cat("Final path: ", path)
Final path:  1 3 7 6 2 5 4
> 
```

→

Test case 2:-



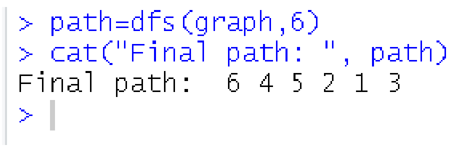
Graph



	V1	V2	V3	V4	V5	V6
1	FALSE	TRUE	FALSE	FALSE	TRUE	FALSE
2	TRUE	FALSE	TRUE	FALSE	TRUE	FALSE
3	FALSE	TRUE	FALSE	TRUE	FALSE	FALSE
4	FALSE	FALSE	TRUE	FALSE	TRUE	TRUE
5	TRUE	TRUE	FALSE	TRUE	FALSE	FALSE
6	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE

OUTPUT

Starting node:- 6



```
> path=dfs(graph,6)
> cat("Final path: ", path)
Final path:  6 4 5 2 1 3
> 
```