dfs.pl

File Edit Browse Compile Prolog Pce Help

dfs.pl

```
dfs(Start, Goal) :-
    dfs(Start, Goal, [], Path),
    reverse(Path, PathReversed),
    write('Path: '), write(PathReversed).
dfs(Node, Node, Visited, [Node | Visited]).
dfs(CurrentNode, Goal, Visited, Path) :-
    connected(CurrentNode, NextNode),
    \+ member(NextNode, Visited),
    dfs(NextNode, Goal, [CurrentNode | Visited], Path).
connected(a, b).
connected(a, c).
connected(b, d).
connected(b, e).
connected(c, f).
```

d:/prolog/dfs.pl compiled

SWI-Prolog (AMD64, Multi-threaded, version 9.0.4)

File Edit Settings Run Debug Help

```
% d:/prolog/dfs compiled 0.00 sec, 0 clauses
?- dfs(a, e).
Path: [a,b,e]
true
```

```python
def dfs(graph,start,end):
    visited=set()
    stack=[start]
    while stack:
        node=stack.pop()
        if node==end:
            print(node,end=" ")
            break
        if node not in visited:
            visited.add(node)
            print(node,end=" ")
            for neighbour in graph[node]:
                if neighbour not in visited:
                    stack.append(neighbour)
graph={}
n=int(input("Enter number of nodes : "))
for _ in range(n):
    node=input("Enter name of node : ")
    neighbours=input("Enter space seperated neighbour node name : ").split()
    graph[node]=neighbours
start=input("Enter start node : ")
end=input("Enter end node : ")
print("DFS Path: ",end=" ")
dfs(graph,start,end)
```

OUTPUT:

```
Enter number of nodes : 5
Enter name of node : a
Enter space seperated neighbour node name : b c
Enter name of node : b
Enter space seperated neighbour node name : a d
Enter name of node : c
Enter space seperated neighbour node name : a
Enter name of node : d
Enter space seperated neighbour node name : b e
Enter name of node : e
Enter space seperated neighbour node name : d
Enter start node : a
Enter end node : e
DFS Path:  a c b d e
```