

99 questions/Solutions/87

From HaskellWiki

< 99 questions | Solutions

```
type Node = Int
type Edge = (Node,Node)
type Graph = ([Node],[Edge])

depthfirst :: Graph -> Node -> [Node]
depthfirst (v,e) n
  | [x|x<-v,x==n] == [] = []
  | otherwise = dfrecursive (v,e) [n]

dfrecursive :: Graph -> [Node] -> [Node]
dfrecursive ([],_) _ = []
dfrecursive (_,_) [] = []
dfrecursive (v,e) (top:stack)
  | [x|x<-v,x==top] == [] = dfrecursive (newv, e) stack
  | otherwise = top : dfrecursive (newv, e) (adjacent ++ stack)
where
  adjacent = [x | (x,y)<-e,y==top] ++ [x | (y,x)<-e,y==top]
  newv = [x|x<-v,x/=top]
```

Call it:

```
depthfirst ([1,2,3,4,5],[(1,2),(2,3),(1,4),(3,4),(5,2),(5,4)]) 1
```

Retrieved from "https://wiki.haskell.org/index.php?title=99_questions/Solutions/87&oldid=38924"

- This page was last modified on 5 March 2011, at 23:26.
- Recent content is available under a simple permissive license.