

# 99 questions/Solutions/88

## From HaskellWiki

< 99 questions | Solutions

```
import Data.List
```

```
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]
```

```
connectedcomponents :: Graph -> [[Node]]
```

```
connectedcomponents ([],_) = []
```

```
connectedcomponents (top:v,e)
  | remaining == [] = [connected]
  | otherwise = connected : connectedcomponents (remaining, e)
where
  connected = depthfirst (top:v,e) top
  remaining = (top:v) \\ connected
```

You can call it:

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

Retrieved from "[https://wiki.haskell.org/index.php?title=99\\_questions/Solutions/88&oldid=38925](https://wiki.haskell.org/index.php?title=99_questions/Solutions/88&oldid=38925)"

- 
- This page was last modified on 5 March 2011, at 23:30.
  - Recent content is available under a simple permissive license.