

Topological Sort

```
-- A TOPOLOGICAL SORT of a directed acyclic graph is a list of all its nodes --
-- in which the start node of each edge occurs earlier than its finish node --
```

```
import GRAPH
```

```
-- topSort g : a topological sort of the directed acyclic graph 'g'
```

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

```
topSort g = topSort' g ( nodes g )
```

```
-- topSort' g ns : a topological sort of the subgraph of 'g'
-- induced by the nodes in the list 'ns'
```

```
topSort' :: Graph -> [ Node ] -> [ Node ]
```

```
topSort' _ [ ] = [ ]
```

```
topSort' g ns = scs ++ topSort' g ( remList ns scs ) where scs = sources g ns
```

```
-- sources g ns : a list of those nodes in graph 'g' from list 'ns'
-- which have no incoming edges from nodes in 'ns'
```

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

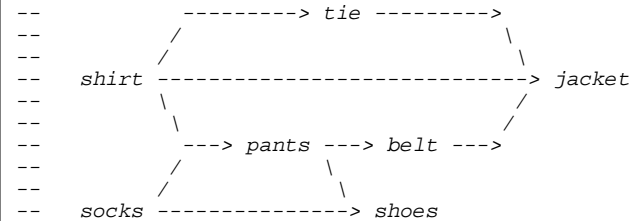
```
sources g ns = [ n | n <- ns, null [ e | e <- edges g,
                                         finish e == n, elem ( start e ) ns ] ]
```

```
-- remList xs ys : the list of items in list 'xs' but not in list 'ys'
```

```
remList :: Eq a => [ a ] -> [ a ] -> [ a ]
```

```
remList xs ys = [ x | x <- xs, notElem x ys ]
```

```
g = graph [ "belt", "jacket", "pants", "shirt", "socks", "shoes", "tie" ]
      [ ( "belt", "jacket" ),
        ( "pants", "belt" ),
        ( "pants", "shoes" ),
        ( "shirt", "jacket" ),
        ( "shirt", "pants" ),
        ( "shirt", "tie" ),
        ( "socks", "pants" ),
        ( "socks", "shoes" ),
        ( "tie", "jacket" ) ]
```



```
> map n2s ( topSort g )
["shirt","socks","pants","tie","belt","shoes","jacket"]
```

```
> map n2s ( sources g ( nodes g ) )
["shirt","socks"]
```