99 questions/Solutions/83

From HaskellWiki

< 99 questions | Solutions

(**) Construct all spanning trees

Write a predicate s_tree(Graph,Tree) to construct (by backtracking) all spanning trees of a given graph.

Here is a working solution that generates all possible subgraphs, then filters out those that meet the criteria for a spanning tree:

```
data Graph a = Graph [a] [(a, a)]
               deriving (Show, Eq)

k4 = Graph ['a', 'b', 'c', 'd']
     [('a', 'b'), ('b', 'c'), ('c', 'd'), ('d', 'a'), ('a', 'c'), ('b', 'd')]

paths' :: (Eq a) => a -> a -> [(a, a)] -> [[a]]
paths' a b xs | a == b = [[a]]
              | otherwise = concat [map (a :) $ paths' d b $ [x | x <- xs, x /= (c, d)]
                                    | (c, d) <- xs, c == a] ++
                                  concat [map (a :) $ paths' c b $ [x | x <- xs, x /= (c, d)]
                                    | (c, d) <- xs, d == a]

cycle' :: (Eq a) => a -> [(a, a)] -> [[a]]
cycle' a xs = [a : path | e <- xs, fst e == a, path <- paths' (snd e) a [x | x <- xs, x /= e]
              [a : path | e <- xs, snd e == a, path <- paths' (fst e) a [x | x <- xs, x /= e]

spantree :: (Eq a) => Graph a -> [Graph a]
spantree (Graph xs ys) = filter (connected) $ filter (not . cycles) $ filter (nodes) alltrees
   where
      alltrees = [Graph (ns edges) edges | edges <- foldr acc [[]] ys]
      acc e es = es ++ (map (e:) es)
      ns e = foldr (\x xs -> if x `elem` xs then xs else x:xs)
             [] $ concat $ map (\(a, b) -> [a, b]) e
      nodes (Graph xs' ys') = length xs == length xs'
      cycles (Graph xs' ys') = any ((/=) 0 . length . flip cycle' ys') xs'
      connected (Graph (x':xs') ys') = not $ any (null) [paths' x' y' ys' | y' <- xs']
```