99 questions/Solutions/86

From HaskellWiki

< 99 questions | Solutions

(**) Node degree and graph coloration

Use Welch-Powell's algorithm to paint the nodes of a graph in such a way that adjacent nodes have different colors.

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

data Adjacency a = Adj [(a, [a])]
                   deriving (Show, Eq)

petersen = Graph ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']
                 [('a', 'b'), ('a', 'e'), ('a', 'f'), ('b', 'c'), ('b', 'g'),
                  ('c', 'd'), ('c', 'h'), ('d', 'e'), ('d', 'i'), ('e', 'j'),
                  ('f', 'h'), ('f', 'i'), ('g', 'i'), ('g', 'j'), ('h', 'j')]

-- produces graph coloration using Welch-Powell algorithm
kcolor :: (Eq a, Ord a) => Graph a -> [(a, Int)]
kcolor g = kcolor' x [] 1
   where
      Adj x = sortg g
      kcolor' [] ys _ = ys
      kcolor' xs ys n = let ys' = color xs ys n
                        in kcolor' [x | x <- xs, notElem (fst x, n) ys']
                                   ys'
                                   (n + 1)
      color []          ys n = ys
      color ((v, e):xs) ys n = if any (\x -> (x, n) `elem` ys) e
                               then color xs ys n
                               else color xs ((v, n) : ys) n

-- determines chromatic number, given graph coloration
chromatic :: [(a, Int)] -> Int
chromatic x = length $ foldr (\(a, n) xs -> if n `elem` xs then xs else n : xs) [] x

-- converts from graph to adjacency matrix representations
graphToAdj :: (Eq a) => Graph a -> Adjacency a
graphToAdj (Graph [] _)      = Adj []
graphToAdj (Graph (x:xs) ys) = Adj ((x, ys >>= f) : zs)
   where
      f (a, b)
         | a == x = [b]
         | b == x = [a]
         | otherwise = []
      Adj zs = graphToAdj (Graph xs ys)

-- produces graph sorted by node degree
sortg :: (Eq a, Ord a) => Graph a -> Adjacency a
sortg g = Adj $ map (\(a, b) -> (a, sort b 1 maximum)) $ sort x 1 maxv
```

```
where
   Adj x = graphToAdj g
   sort [] _ _ = []
   sort xs n f = let m = f xs in
                 m : sort [x | x <- xs, x /= m] (n + 1) f
   maxv (x:xs) = foldr (\a@(a1, _) b@(b1, _) -> if a1 > b1 then a else b) x xs
```

Retrieved from "https://wiki.haskell.org/index.php?title=99_questions/Solutions/86&oldid=57144"

- This page was last modified on 22 November 2013, at 20:14.
- Recent content is available under a simple permissive license.