Farbenie mapy


```
:load PlanarGraphColoring.hs
```

Skúsenosť s farbením mapy som mal z minulého roka na predmete `1-AIN-304/15`. Tam sme tento problém definovali ako `CSP` (Constraint Satisfaction Problem), t.j. prehnali sme to backtrackingom s nejakými doplnkovými heuristikami, ako napríklad spôsob výberu ďalšieho štátu a pod. Generálnu ideu som teda mal, nastavil som si aj testovaciu mapu Austrálie (je menšia) a pokúsil sa pretransformovať moje myšlienky do Haskellu. Najväčším problémom bolo odkloniť myslenie z `mutable` dátových štruktúr, ako napr. `mapa`. Po dni nervov som sa konečne dopracoval k možným riešeniam, hoci bez pokročilých heuristík, aspoň teda niečo.

Najprv nejaké základné funkcie na prístup k jednotlivým dátam a vlastný typ `Assignment`:

```
type Assignment = [(Country, Color)]

countries :: Graph -> [Country]
countries g = [c| (c, _) <- g]

colors :: [Color]
colors = [Red, Green, Blue, Yellow]

neighbors :: Graph -> Country -> Neighbors
neighbors g ctr = [n | (c, neighs) <- g, c == ctr, n <- neighs]
```

Problém, s ktorým som bojoval bol ten, ako zistiť farby susedných štátov v nejakom čiastočnom priradení (`Assignment`). Podarilo sa mi to nasledovnou funkciou v spolupráci s funkciou `neighbors`.

```
getNeighborsColors :: Graph -> Country -> Assignment -> [Color]
getNeighborsColors g ctr as = [col | (c, col) <- as, ctr `elem` neighbors g c]
```

Základná šablóna pre vyhľadávanie **všetkých** farebných priradení bola v skratke takáto:

```
search :: Graph -> [Country] -> [Assignment]
search g [] = [[]]
search g ctrs = [(ctr, col):rem | ctr <- ctrs,
                    rem <- search g (ctrs \\ [ctr]), col <- colors]
```

Samozrejme bolo treba nastaviť nejakú priebežnú kontrolu, primárne na to, aby sa neofarbovali susedné štáty rovnakou farbou. To sa dalo vyriešiť vyberaním farby z nejakej vypočítanej množiny dovolených farieb:

```
possibleColors :: Graph -> Country -> Assignment -> [Color]
possibleColors g ctr as = [col | col <- colors,
                           col `notElem` getNeighborsColors g ctr as]
```

Potom:

```
search :: Graph -> [Country] -> [Assignment]
search g [] = [[]]
search g ctrs = [(ctr, col):rem | ctr <- ctrs,
                    rem <- search g (ctrs \\ [ctr]),
                    col <- possibleColors g ctr rem]
```

Pri malej mape, ako je mapa Austrálie sa mohlo rýchlo stať, že ofarbenie spĺňalo podmienku, že susediace štáty sú rozdielnych farieb, no pre estetiku (a možno aj zadanie?) som doplnil ďalšie obmedzenie výsledku, a to kontrolu na prítomnosť všetkých štyroch farieb:

```
allFourColors :: Graph -> Assignment -> Bool
allFourColors g as | lg == la = length (nub [col | (ctr, col) <- as]) == lc
                   | otherwise = True
                 where lg = length g
                       la = length as
                       lc = length colors
```

Výsledná prehľadávacia funkcia tak vyzerá nasledovne:

```
search :: Graph -> [Country] -> [Assignment]
search g [] = [[]]
search g ctrs = [(ctr, col):rem | ctr <- ctrs,
                        rem <- search g (ctrs \\ [ctr]),
                        col <- possibleColors g ctr rem,
                        allFourColors g ((ctr, col):rem)]
```

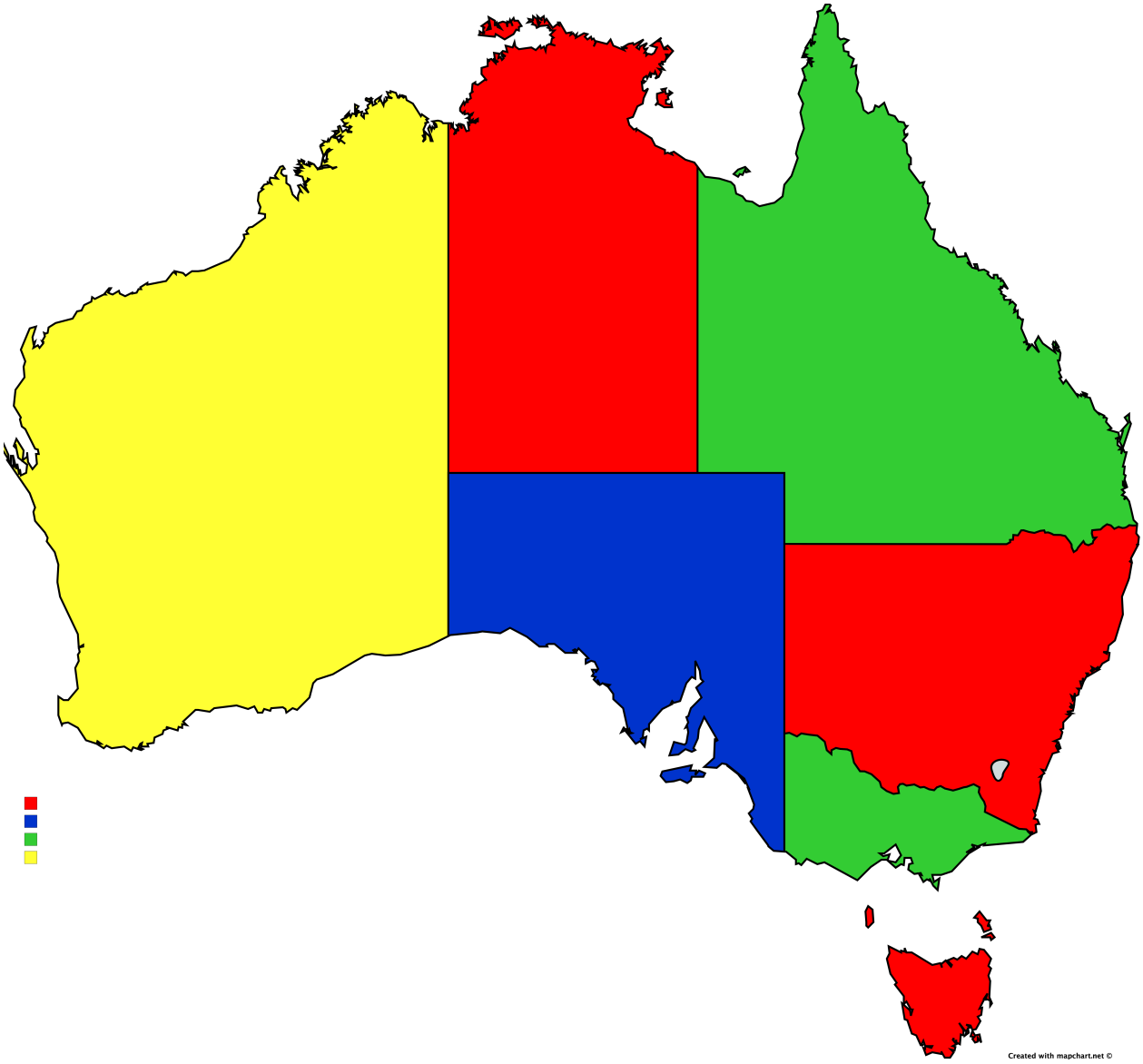
Takto sa nám vypočítajú všetky možné korektné ofarbenia, je ich však pomerne veľa a nám v praxi stačí jedno, tak si obmedzme výpis:

```
coloring :: Graph -> Assignment
coloring g = head $ search g (countries g)
```

Farbenie mapy

Pozrime sa na jedno z možných ofarbení mapy Austrálie:

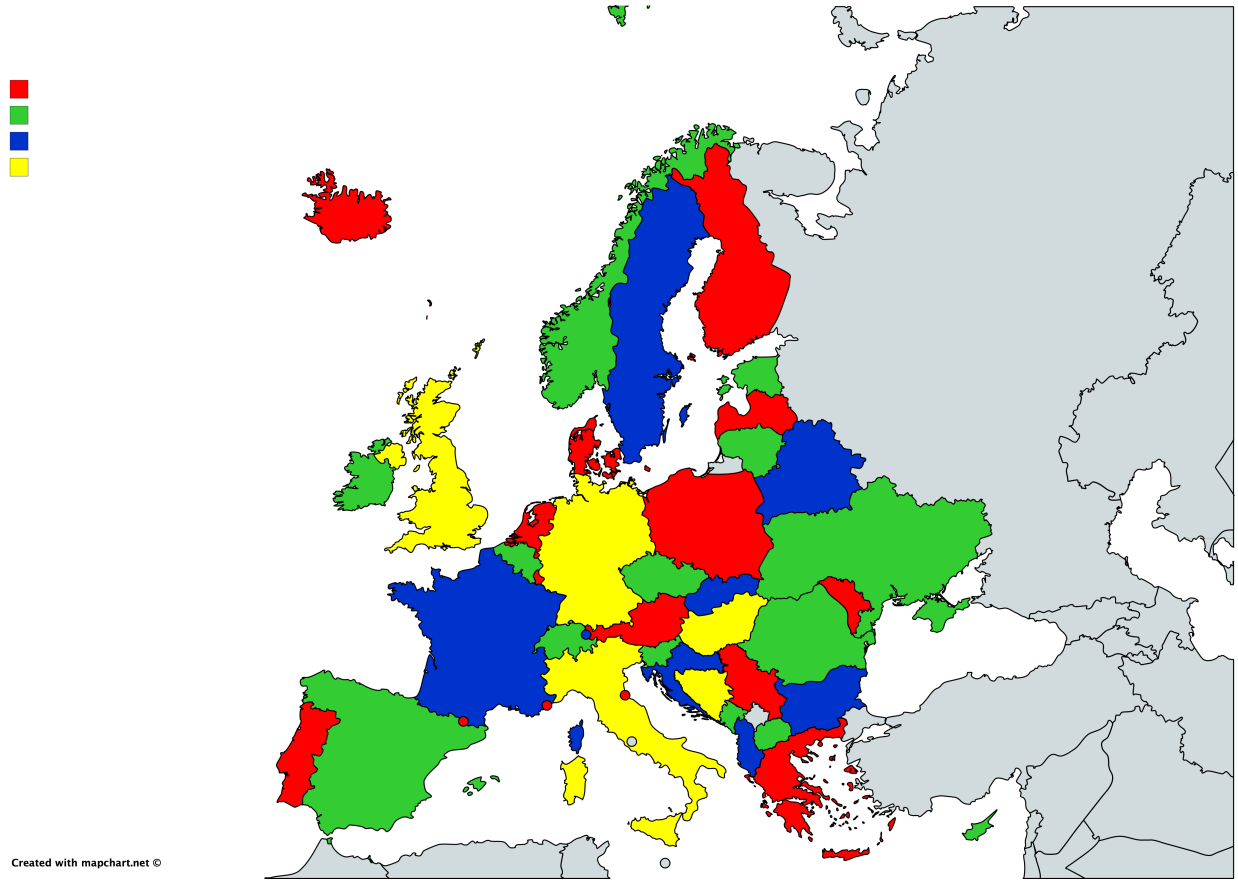
```
*PlanarGraphColoring> coloring australia
[("Western Australia",Yellow),("Northern Territory",Red),("South Australia",Blue),
("Queensland",Green),("New South Wales",Red),("Victoria",Green),("Tasmania",Red)]
(0.01 secs, 302,272 bytes)
```



`Pozn:` Tasmánia (ostrov) a Viktória (najjužnejší štát pevninskej časti) v pravom slova zmysle spolu nesusedia, no aby som sa vyhol neestetickému ofarbeniu rovnakou farbou, túto susednosť som explicitne pridal.

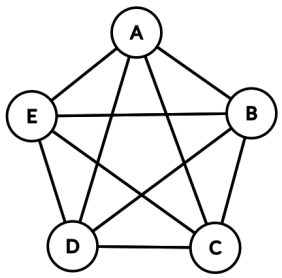
A teraz jedno z možných ofarbení mapy Európy:

```
*PlanarGraphColoring> coloring europe
[("portugal",Red),("spain",Green),("andorra",Red),("france",Blue),
("united_kingdom",Yellow),("ireland",Green),("monaco",Red),("italy",Yellow),
("san_marino",Red),("switzerland",Green),("liechtenstein",Blue),("germany",Yellow),
("belgium",Green),("netherlands",Red),("luxembourg",Red),("austria",Red),
("slovenia",Green),("croatia",Blue),("bosnia",Yellow),("montenegro",Green),
("albania",Blue),("greece",Red),("cyprus",Green),("macedonia",Green),
("bulgaria",Blue),("serbia",Red),("romania",Green),("hungary",Yellow),
("slovakia",Blue),("czech_republic",Green),("poland",Red),("denmark",Red),
("sweden",Blue),("norway",Green),("finland",Red),("iceland",Red),
("ukraine",Green),("moldova",Red),("belarus",Blue),("lithuania",Green),
("estonia",Green),("latvia",Red)]
(22.30 secs, 5,047,617,680 bytes)
```



`Alert`: Vďaka tejto vizualizácii si môžeme všimnúť, že je chyba v grafom súbore, keďže na oko jasní susedia Ukrajina a Rumunsku boli o tento vzťah ochudobnení. Na druhú stranu, mapa očividne obsahuje aj korektné informácie na úkor tých skutočných, viď Krym :).

Ako sa spomína v zadaní, graf, ktorý nedokážeme ofarbiť len 4 farbami už zrejme nebude rovinný. Z definície rovinného grafu sa dá pomerne ľahko vymyslieť príklad jednoduchého nerovinného grafu - 'pentagonu':



Vyskúšajme ho teda ofarbiť 4 farbami:

```
pentagon :: Graph
pentagon = [
    ("A", ["B", "C", "D", "E"]),
    ("B", ["A", "C", "D", "E"]),
    ("C", ["A", "B", "D", "E"]),
    ("D", ["A", "B", "C", "E"]),
    ("E", ["A", "B", "C", "D"])
    ]
```

```
*PlanarGraphColoring> coloring pentagon
*** Exception: Prelude.head: empty list
```

Vidíme, že funkcia `coloring` sa podľa svojej definície snaží vybrať prvé riešenie, ktoré jej poskytla funkcia `search`. Také riešenie však nie je ani jedno, tak to padne na chybe, čo potvrdzuje správnosť riešenia.

Pridajme si ďalšiu farbu:

```
data Color = Red | Green | Blue | Yellow | Purple deriving (Show, Eq)
```

```
*PlanarGraphColoring> coloring pentagon
[("A",Purple),("B",Yellow),("C",Blue),("D",Green),("E",Red)]
(0.01 secs, 196,424 bytes)
```

A podarilo sa: