March 13, 2019

Homework 3: IsBipartite Algorithm
50 Points

A $k - coloring$ of a graph $G = (V, E)$ is a mapping $color : V \to \{1, \ldots, k\}$ such that for any edge $e = [v, w] \in E$ $color(v) \neq color(w)$. G is said to be *bipartite* iff G has a 2-coloring.

The idea of a k-coloring is that the vertices can be "colored" using k distinct colors so that the vertices of any edge have different colors. A bipartite graph is one that can be colored with two colors.

Part 1. Invent an algorithm named **IsBipartite** with these properties:

(1) IsBipartite operates on any undirected graph $G = (V, E)$
(2) IsBipartite returns true iff G is bipartite
(3) If IsBipartite returns true then a supplied vector will be populated with a 2-coloring of the vertices of G
(4) The runtime of IsBipartite is $\leq \mathcal{O}(|V| + |E|)$

Part 2. Code up the algorithm in C++ conformant with the stub below. Test the implementation on small graphs that can be hand verified and on some large graphs (such as the "Kevin Bacon" actor-movie abstract graph) and some very large graphs generated at random. Include some random maze graphs, and report any discoveries.

Part 3. Provide a proof that your algorithm is correct.

Part 4. Provide a proof that your algorithm has runtime $\leq \mathcal{O}(|V| + |E|)$

Here is C++ code stub in which to code your algorithm. Note that the graph `g` and the vector `color` are passed by const reference and non-const reference, respectively.

```
template < class G >
bool IsBipartite ( const G& g , fsu::Vector <char>& color )
{
  // code goes here
}

template < class G >
bool IsBipartite ( const G& g )
{
  fsu::Vector<char> color (g.VrtxSize());
  return IsBipartite (g,color);
}
```

Cite your sources!