Recall that a legal coloring of a graph $G$ assigns colors to the vertices such that adjacent vertices never receive the same color. The minimum number of colors needed for this is the chromatic number $\chi(G)$ of the graph. The graph $G$ is bipartite if $\chi(G) \leq 2$.

Let $G = (V, E)$ be a graph with $n$ vertices. We assume $V = \{1, 2, \ldots, n\}$.

The greedy coloring algorithm assigns a color (non-negative integer) $c(x)$ to each vertex $x$ in a greedy manner as follows. The variable $k$ stores the number of colors used; this will be the output. Notation: $\text{adj}(i)$ is the list of vertices adjacent to vertex $i$.

0 $k := 0$
1 for $i = 1$ to $n$ do
2 let $c(i)$ be the smallest positive integer such that
3 $c(i) \notin \{c(j) \mid j < i, j \in \text{adj}(i)\}$
4 if $c(i) > k$ then $k := c(i)$
5 end(for)
6 return $k$

It should be clear that the assignment $c(.)$ defined by the algorithm is a legal coloring of $G$. Observe that the colors used are exactly the numbers $\{1, \ldots, k\}$.

**Problem.** (a) (“Greedy coloring is not so bad”) Prove: the number of colors used is at most $1 + \deg_{\max}$. ($\deg_{\max}$ is the maximum degree.)

(b) (“Greedy coloring is terrible”) Let $n$ be even. Construct a bipartite graph with $n$ vertices so that the greedy coloring algorithm will use a whopping $n/2$ colors. (You need to state for all $i$ and $j$ whether or not $i$ and $j$ are adjacent. Just giving the graph up to isomorphism does not determine what the greedy coloring does.)

(c) (“Greedy coloring can be optimal”) Given a graph, prove that one can relabel it (permute the vertex labels) such that the greedy coloring algorithm gives an optimal coloring (uses $k = \chi(G)$ colors, where $\chi(G)$ is the chromatic number). (Catch: we cannot efficiently find this relabeling. But it exists.)

(d) Implement the greedy coloring algorithm in linear time ($O(n + m)$ where $m$ is the number of edges). $G$ is given in the adjacency array representation (array of adjacency lists). “Implementation” refers to a detailed description of how you execute Line 2. Prove that your algorithm runs in linear time.