1. Consider the following questions.

(a) Give an example of a graph for which \textsc{Approx-Vertex-Cover} always yields a suboptimal solution.

(b) Why is it important to find tight examples for approximation algorithms? (Hint: For example, suppose we have a 2-approximation algorithm for some problem. Why do we care about finding an example where the algorithm returns a solution that is exactly twice as costly as the optimal solution?)

2. Let $G = (V, E)$ be an undirected graph. A \textit{k-coloring} of $G$ is an assignment of $k$ colors to its vertices. A $k$-coloring is said to be \textit{proper} if no two adjacent vertices have the same color. The minimum $k$ for which the graph $G$ has a proper $k$-coloring is known as its \textit{chromatic number}, denoted by $\chi(G)$. Computing $\chi(G)$ is computationally difficult, i.e., \textsc{NP}-hard. Below is an example of (a) a graph $G$ along with (b) a 4-coloring for it.

(a) (For a definition of the $n$-vertex complete graph, $K_n$, see HW 10).

- What is $\chi(K_n)$, i.e., the chromatic number of the $n$-vertex complete graph?
- What is the chromatic number of a path?
- What is the chromatic number of a cycle?
- What is the chromatic number of the above depicted graph $G$?

(b) Recall the \textit{degree} of a vertex is the number of edges incident to it. There is a simple greedy algorithm that finds a proper vertex-coloring for a graph using $\Delta + 1$ colors, where $\Delta$ is the maximum degree of the graph, i.e., the largest degree over all vertices. Give a tight example for this algorithm, i.e., show that
there is a graph $G$ such that $\chi(G) = \Delta + 1$.

(c) Determine the chromatic number of the following graph, and prove your answer is indeed optimal.

3.

In the maximum clique problem, we are given a graph $G = (V, E)$ and an integer $k$. The goal is to decide if there is a clique on $k$ vertices, that is, a complete subgraph on $k$ vertices.

(a) Give an efficient algorithm for solving the maximum clique problem, when the input graph is a tree. Prove that your algorithm is correct, and runs in polynomial time.

(b) Let $G$ be an arbitrary connected undirected graph.

- Can we efficiently determine if $G$ contains a triangle, that is, a complete subgraph on 3 vertices?
- Can we efficiently determine if $G$ contains a complete subgraph on $2^4$ vertices?

4.

Suppose we have a problem we can solve in polynomial time. Does it make sense to consider an approximation algorithm for such a problem? Why or why not?