CS-E3190 Principles of Algorithmic Techniques

05. Greedy Algorithms – Graded Exercise

Please read the following **rules** very carefully.

- Do not consciously search for the solution on the internet.
- You are allowed to discuss the problems with your classmates but you should **write the solutions yourself**.
- Be aware that **if plagiarism is suspected**, you could be asked to have an interview with teaching staff.
- Each week the second exercise is an **individual exercise**, and the teaching staff will not give hints or help with them. You are allowed to ask for hints for the first exercise.
- In order to ease grading, we want the solution of each problem and subproblem to start on a **new page**. If this requirement is not met, **points will be reduced**.

1. **Greedy graph coloring.** Let $G = (V, E)$ be a graph with n nodes. We want to give a proper coloring $c : V \to \mathbb{N}$ of the vertices. Consider the following naive greedy algorithm.

    1. Pick an arbitrary vertex.
    2. Give it the smallest possible value.

    This algorithm has an upper bound of $\Delta + 1$ colors.

    (a) (2p.) Design a new greedy algorithm such that the number of colors is bounded by $\max_{1\leq i\leq n} \min(d_i + 1, i)$ where d_i is the degree of the node v_i that gets colored in iteration i. Prove the correctness of the algorithm.
    Hint: Think about ordering the nodes.

    (b) (2p.) An interval graph is a graph that corresponds to a family of intervals $\{I_u\}_{u\in V}$ of $[0, 1]$ such that $\{u, v\} \in E \Leftrightarrow I_u \cap I_v \neq \emptyset$. Design an optimal graph coloring algorithm for interval graphs using a greedy approach and prove that it is correct. You can assume that the intervals corresponding to the interval graph are given as input.
    Hint: Think about ordering the nodes again.

    (c) (1p.) Show the first greedy algorithm with an arbitrary ordering is not optimum, not even for interval graphs.

2. **Individual exercise: Greedy coloring of bipartite graphs.** A greedy algorithm for graph coloring of bipartite graphs uses the *color-degree* of each node i.e. the number of already colored neighbors. The algorithm is the following:

    1. The color-degree of each node is initialized to 0.
    2. Choose a node v with maximum color-degree and give it the smallest possible color.
    3. Update the color-degree of its neighbors.

    (a) (2p.) Show that this algorithm is optimum for bipartite graphs.
    Hint: you can use the fact that a graph is bipartite iff it has no odd cycles.

(b) (2p.) Show that the algorithm does not necessarily output an optimum coloring for general graphs.

(c) (1p.) Analyze the runtime of the algorithm.