EXTRA EXERCISES CONCERNING THE LECTURE

MAXIMILIAN JANKE

In this document I outline certain exercises which somehow relate to selected topics from the lecture Advanced Algorithms held by Professor Albers in the winter term 2018/19: Divide and Conquer, Fibonacci heaps, Greedy algorithms and dynamic programming. I mostly chose these exercises because I personally find them interesting and now upload them as some students asked me to. Note though that these exercise do not aim at representing or preparing for the exam, though thinking about them should not be detrimental. I do not plan on publishing solutions to the exercises, albeit I am willing to give hints, sketches of solutions or reference for chosen exercises if people are interested. Feel free to discuss the exercises and give feedback (especially if you spot mistakes) via e-mail or during my office hours.

1. Graph coloring

Given an undirected graph $G = (V, E)$ where each node has degree at most d outline a linear-time greedy-algorithm that finds a $(d + 1)$-coloring of G, i.e. a function $f\colon V \to \{1, \dots d+1\}$ such that no two neighbouring nodes in G have the same color.

Give an example of a graph on which the greedy algorithm uses the minimum amount of colors possible and an example where it does not.

2. Refueling

We want to take a road-trip along a street of length L. For each distance unit the car needs one unit of fuel. Its tank can contain t fuel units and is full at the beginning of the trip. Along the road there are n gas stations which are described by a pair of numbers $(l_i, p_i) \in \mathbb{N}^2$. Here l_i denotes their respective distance from the start, while p_i is the price at which the gas station sells fuel. Whenever the car reaches a gas station, we can buy as many units of fuel as fit into its tank at a price of p_i per unit. Of course we may buy less. Each unit of fuel allows us to travel one unit of distance. We assume that the gas stations are ordered increasingly by the numbers l_i and that it is possible for the car to make the trip if it fully refuels at every gas station along the way. Of course by doing so it may spend more money than necessary.

Give a linear-time algorithm that outputs the minimum amount of money needed to make the trip.

3. Balancing a binary tree

Let be a binary tree T with n nodes and a number k. On the first exercise sheet you showed that there exists a partition of the nodes of T into one set of size k

Date: January 29, 2019.

and one set of size $n-k$ such that there are at most $3\log_2(n)$ edges between both sets. Now find a polynomial algorithm that, given T and k, finds the minimum number of edges between the two sets of any partition where one set has size k and the other has size $n-k$.

4. Fibonacci heap

Consider a Fibonacci heap that immediately consolidates itself after every operation. Show that all operations but the union-operation still have the same amortized runtime as in the lecture.

5. Travelling Salesman

In the travelling salesman problem, a complete graph $K_n = (V, E)$ with $n \geq 3$ nodes is given as well as edge weights $c\colon E \to \mathbb{R}$. One now wants to find a cycle in K_n of minimum weight that visits every node exactly once.

Find an algorithm with exponential running time for the problem, i.e. running time $O(c^n)$ for some constant c.

6. Longest Path [CM]

6.1. **colored graph.** Let $k \in \mathbb{N}$ be a positive integer. We are given a k-colored graph, that is a graph $G = (V, E)$ with a coloring function $f\colon V \to \{1, \dots, k\}$. We make no assumptions on the colors related to the structure of the graph. In particular we do not assume neighbouring nodes to be distinctly colored. We want to find a path that visits exactly one node of each color or decide that none exists.

Give an algorithm that has running time $O\left(n^{100}\right)$ if k is fixed. (Of course the constant in the O-notation will grow quite fast in k.)

6.2. **Arbitrary graph.** Given a number $k \in \mathbb{N}$ and a graph G we want to decide whether the graph G contains a path of length k that visits no node twice. Use the algorithm for the previous problem to find a randomized algorithm that has running time $O\left(n^{100}\right)$ if k is fixed and outputs such a path with constant probability. (Of course the constant in the O-notation will grow quite fast in k.)

If the number k is part of the input (and not fixed) this problem is NP-hard. In particular no polynomial algorithm is known.

7. Bridges in a graph

Given an undirected, connected graph G an edge $e \in E(G)$ is called a *bridge* if G becomes disconnected if e is removed. In linear time, find a bridge in G or decide that none exists.

References

[CM] Cygan, Marek, et al. *Parameterized algorithms.* Vol. 3. Cham: Springer, 2015. 2

E-mail address: maximilian.janke@in.tum.de