Albert-Ludwigs-Universität, Institut für Informatik
Prof. Dr. Fabian Kuhn
M. Ahmadi, P. Schneider, J. Uitto

November 30, 2017

Algorithm Theory - Winter Term 2017/2018
Exercise Sheet 4

Hand in by Thursday 10:15, December 14, 2017

Exercise 1: Data Structures - Heapsort (2+2+2+2+2 Points)

We can employ a binary min-heap to sort a sequence of n distinct keys in an ascending order as follows. First, we insert all keys one by one with the `Insert` operation. Then, we obtain the sorted sequence by removing all n keys with `Delete-Min`.

(a) Show that `Heapsort` takes $\Theta(n \log n)$ if the input sequence is sorted *descending*.

(b) Show that `Heapsort` takes $\Theta(n \log n)$ if the input sequence is sorted *ascending*.

Assume that you have two binary min-heaps that only allow manipulation via the operations `Get-Min`, `Insert`, `Delete-Min`, `Decrease-Key`, `Get-Size` and `Is-Empty` and may have duplicate keys.

(c) Give a protocol that merges two heaps of size n and m, where $m < n$, in time $\mathcal{O}(m \log n)$ and prove the runtime.

(d) For arbitrary m and n, where $m < n$, give two binary min-heaps of size m and n where the `Merge` protocol takes $\Omega(m \log n)$.

(e) For arbitrary m and n, where $m < n$, give two binary min-heaps of size m and n where the `Merge` protocol takes $\mathcal{O}(m)$.

Exercise 2: Union-Find (5+5 Points)

(a) In the lecture the union-by-size heuristic was introduced to guarantee shallow trees when implementing a Union-Find data structure. Another heuristic that can be used for `union(x,y)` is the union-by-rank heuristic. For the heuristic, the rank of a tree is defined as follows:

(1) The rank $r(T)$ of a tree T consisting of only one node is 0.

(2) When joining trees T_1 and T_2 by attaching the root of tree T_2 as a new child of the root of tree T_1, the rank of the new combined tree T is defined as $r(T) := \max\{r(T_1), r(T_2) + 1\}$.

When applying the union-by-rank heuristic, whenever combining two trees into one tree (as the result of a union operation), we attach the tree of smaller rank to the tree of larger rank (if both trees have the same rank, it does not matter which tree is attached to the other tree). Provide pseudo-code for the `union(x,y)` operation when using the union-by-rank heuristic.

Show that when implementing a Union-Find data structure by using disjoint-set forests with the union-by-rank heuristic, the height of each tree is at most $O(\log n)$.

(b) Demonstrate that the above analysis is tight by giving an example execution (of merging n elements in that data structure) that creates a tree of height $\Theta(\log n)$. Can you even get a tree of height $\lfloor \log_2 n \rfloor$?

Exercise 3: Max Flow (6 Points)

You are given a (connected) directed graph $G = (V, E)$, with positive integer capacities on each edge, a designated source $s \in V$, and a designated sink $t \in V$. Additionally you are given a current maximum $s - t$ flow $f : E \to \mathbb{N}$.
Now suppose we increase the capacity of one specific edge $e_0 \in E$ by one unit. Show how to find a maximum flow in the resulting graph with the manipulated capacity in time $\mathcal{O}(|E|)$.

Exercise 4: Network Flow (4+4+6 Points)

Professor Adam has two children who, unfortunately, dislike each other. The problem is so severe that not only do they refuse to walk to school together, but in fact each one refuses to walk on any street that the other child has used. The children have no problem with their paths crossing at street intersections. Fortunately both the professor's house and *the only school* in town are on intersections, but beyond that he is not sure if it is going to be possible to send both of his children to the only school in town. The professor has a map of his town given as a graph $G = (V, E)$, where E are the streets and V are the intersections.

(a) Show how to formulate the problem of determining whether both his children can go to the school in town as a maximum flow problem.

(b) What algorithm from the lecture would you use to solve the problem? Assuming that for a street network $G = (V, E)$, we have $|E| \leq 3|V|$, what is the asymptotic running time of your algorithm as a function of the number of nodes $n = |V|$?

(c) Now, assume that the two children start disliking each other even more and they now only accept to go to the school if their walks also avoid crossing some of the intersections. Assume that we are given a subset $U \subseteq V$ of intersections and we now need to find two paths from the professor's home to the school such that every intersection in U and every street is part of at most one of the two paths. You can of course assume that the professor's home and the school are not in U.

How can you now reduce the problem to a maximum flow problem?