University of Freiburg
Dept. of Computer Science
Prof. Dr. F. Kuhn
Marc Fuchs

Algorithm Theory
Exercise Sheet 6

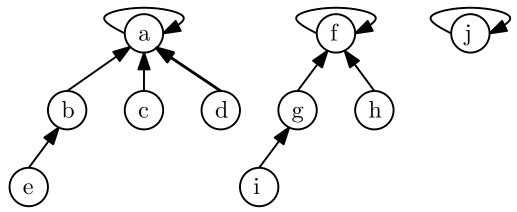
Due: Friday, 28th of November 2025, 10:00 am

Exercise 1: Union-Find *(11 Points)*

In the lecture we have seen two heuristics (i.e., the **union-by-size** and the **union-by-rank** heuristic) to implement the **union-find** data structure. In this exercise we will focus on the **union-by-rank** heuristic only! To solve the following tasks consider the **union-find** data structure implemented by disjoint forest using union-by-rank heuristic and path compression.

(a) Give the pseudocode for `union(x, y)`. *(2 Points)*
Remark: Use $x.parent$ to access the parent of some node x and use $x.rank$ to get its rank. The `find(x)` operation is implemented as stated in the lecture using path compression.

(b) Consider the following state of such a union-find data structure represented as disjoint-set forest where the *current* rank of each node equals the height of the tree rooted at it.



Conduct the following operations *sequentially* on the union-find data structure (the result of the prior operation is the input of the next). Give the state of the data structure after each operation.

(i) `union`(f, j) using the *union-by-rank* heuristic *(1 Point)*

(ii) `union`(b, g) using the *union-by-rank* heuristic *(1 Point)*

(iii) `find`(i) using *path compression* *(1 Point)*

(c) Show that the height of each tree (in the disjoint forest) is at most $O(\log n)$ where n is the number of nodes. *(3 Points)*
Remark: First show that the maximum rank of a tree is at most $O(\log n)$, and then explain why this maximum rank is an upper bound on the height of the tree.

(d) Show that the above's bound is tight, i.e., give an example execution (of `makeSet`'s and `union`'s) that creates a tree of height $\Theta(\log n)$. Proof your statement! *(3 Points)*

Exercise 2: Coming Home *(9 Points)*

Imagine we are in a city with N houses and N people. Each house is numbered from 1 to N and each person is numbered from 1 to N as well. We say that person i's home is house i. However, due to some unexplainable occurrence, each person wakes up in an arbitrary house (random permutation over the numbers 1 to N) and wants to go back to his own. Furthermore, they can not just walk home outside, they need to use hidden tunnels between the houses. These M tunnels are bidirectional connections that connect two distinct houses. The big question here is *if anyone can get home through the tunnels.*

(a) What (graph based) property needs to be fulfilled that each person located at house p_i can come back home to house i? Imagine that each house represents a node in a graph and two nodes are connected by an edge if there exists a tunnel between the representative houses. *(1 Point)*

To make the question more interesting, assume each tunnel t has a capacity $c_t \in \mathbb{N}$ and persons prefer to go through tunnels with large capacities.

(b) Given a fixed threshold $W > 0$, give an algorithm that decides if every person can get home by only taking tunnels with capacity at least W. Your algorithm should run in time $O((N + M) \cdot \log^* N)$. Argue why this is the case. *(5 Points)*

(b) Not given a fixed W, can you find the largest possible W such that a solution exists (i.e., every person can get home)? What is the runtime? Try to make your algorithm as efficient as possible! *(3 Points)*