Technische Universität München
Fakultät für Informatik
Lehrstuhl für Algorithmen und Komplexität
Prof. Dr. Harald Räcke
Richard Stotz

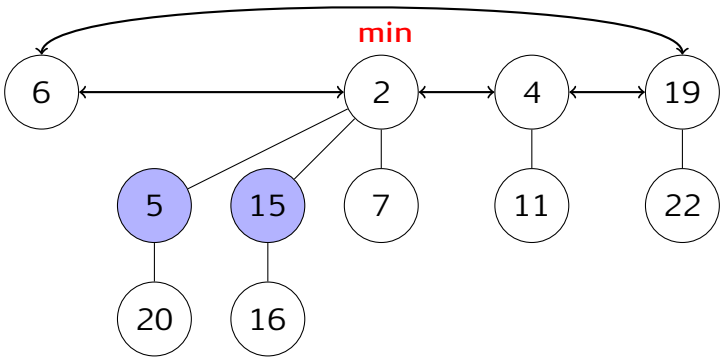
Winter term 2019/20
Assignment 11
January 7, 2020

Efficient Algorithms and Data Structures I

*Deadline: January 13, 10:15 am in the **Efficient Algorithms** mailbox.*

Homework 1 (6 Points)

Perform the following operations sequentially on the Fibonacci Heap shown below so that it remains a Fibonacci Heap. Show what the heap looks like after each operation (always carry out each operation on the result of the previous operation).
Nodes that are marked appear in blue.



1. Insert(1)
2. DeleteMin()
3. Delete(19)
4. DecreaseKey(20,3)

Homework 2 (8 Points)

The `Binomial Arrays` data structure supports the operations INSERT and FIND, defined as usual. It consists of $k = \lceil \log(n+1) \rceil$ sorted arrays $A_0, A_1, \ldots, A_{k-1}$, where n is the number of elements in the data structure. The length of array A_i is 2^i. Let $\langle n_{k-1} n_{k-2} \ldots n_1 n_0 \rangle$ denote the binary representation of n. Array A_i is full if $n_i = 1$ and empty otherwise.

1. Describe how to perform a FIND operation in Binomial Arrays. Prove that your algorithm has worst-case running time $\mathcal{O}((\log n)^2)$.

2. Describe how to perform an INSERT operation in Binomial Arrays. Give a tight upper bound on the worst-case running time of your algorithm; you don't have to show a lower bound. Using the accounting method, prove that your algorithm has amortized running time $\mathcal{O}(\log n)$.

You do not need to prove correctness of your algorithms. All logarithms in this exercise are binary.
Hints: Your implementation of FIND should be inspired by a binary search in an array. For the amortized analysis of INSERT, analyze how many operations are performed on a given element in its lifetime.

Homework 3 (6 Points)

Consider the following sequence of $2n-1$ operations on a disjoint-set data structure with list implementation:

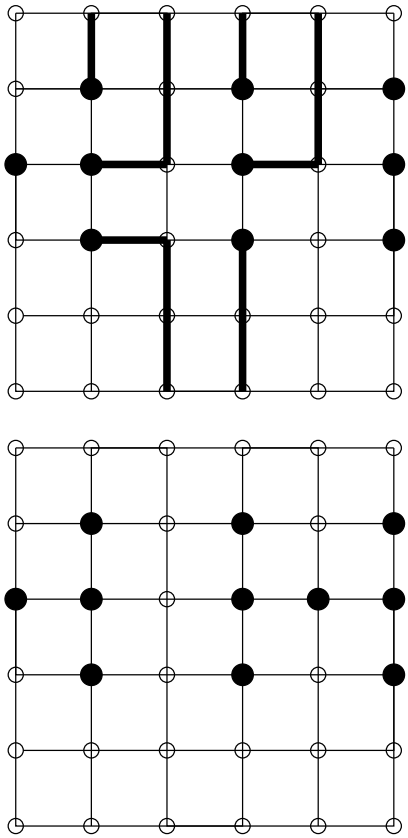
> Makeset(x_1), Makeset(x_2),..., Makeset(x_n),
> Union(x_2, x_1), Union(x_3, x_2), ..., Union(x_n, x_{n-1}).

1. Analyze the asymptotic running time of the sequence if the Union operation is implemented carelessly, appending the longer list to the shorter one.

2. Analyze the asymptotic running time of the sequence if the Union operation is implemented as shown in the lecture, appending the shorter list to the larger one.

Tutorial Exercise 1

An $n \times n$ grid is an undirected graph consisting of n rows and n columns of vertices, as shown in the figures below. We denote the vertex in the ith row and the jth column by (i, j). All vertices in a grid have exactly four neighbors, except for the boundary vertices, which are the points (i, j) for which $i = 1$, $i = \text{n}$, $j = 1$, or $j = \text{n}$. Given $m \leq n^2$ starting points $(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)$ in the grid, the escape problem is to determine whether or not there are m vertex-disjoint paths from the starting points to any m different points on the boundary. For example, the first grid below has an escape, but the second grid does not. Starting points are shown in black.



Describe an efficient algorithm to solve the escape problem, and analyze its running time.

Hint: You might need capacities at both nodes and edges. Argue that this is not a problem.

> In any such war in Europe the rail network of Eastern Europe would be an important target. It therefore appears reasonable to illustrate the method [of using a flow network] by applying it to the Eastern European rail net.
>
> - T.E. Harris, F.S. Ross