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

Winter term 2018/19
Repeat exam
April 8, 2019

Efficient Algorithms and Data Structures I

Last Name	First Name	Matrikel No.	
.....	.....	.....	
Hall	**Row**	**Seat No.**	**Signature**
.....	.....	.....	.....

General Information for the Examination

- Please keep your identity card readily available.
- Do not use pencils. Do not write in red or green ink.
- You are not allowed to use anything except a handwritten A4 sheet.
- Verify that you have received 8 pages, each printed on one side.
- You have 150 minutes to answer the questions.
- Unless noted otherwise, you must justify all your answers.
- It is best to explain algorithms by words. Otherwise use pseudocode.
- Do not write programs as these will cost you points.

Left Examination Hall from ..... to ..... / from ..... to .....
Submitted Early at .....
Special Notes:

	A1	A2	A3	A4	A5	A6	A7	Σ	Examiner
Max.	10	5	3	6	4	7	7	42	
1st									
2nd									

Question 1 (2+2+2+2+2 Points)

True or False? Justify your answer! Without justification, no points are awarded.

(a) $n^{1/\ln(n)} \in \Theta(\ln \ln n)$.

(b) A FIND-operation on a splay tree may increase the height of the tree.

(c) A binomial heap of 63 items is always composed of 6 distinct binomial trees.

(d) A class $\mathcal{H}$ of hash functions $U \to \{0, 1, \ldots, n-1\}$ with $|U| = |\mathcal{H}| = n$ is never universal.

(e) An iteration of the Ford-Fulkerson algorithm never decreases the number of saturated edges in a network.

Question 2 (3+2 Points)

(a) The recursion $T(n)$ is given by

$$T(n) = T(n-2) + 2\ln n \ .$$

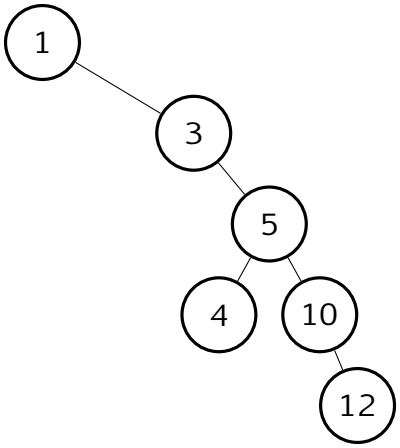
Assuming that $T(n)$ is constant for sufficiently small n, show that $T(n) \in \Theta(n \ln n)$.
Hint: You may use without proof that $\ln(n+1) < \frac{1}{n} + \ln n$.

(b) Solve the following recurrence relation using the characteristic polynomial:

$$a_n = -4a_{n-1} - 4a_{n-2} \text{ for } n \geq 2, \quad a_0 = 6, a_1 = -20 \ .$$

Question 3 (1+1+1 Points)

Perform the following operations sequentially on the Splay Tree shown below so that it remains a Splay Tree. Show what the tree looks like after each operation (always carry out each operation on the result of the previous operation).



(a) Find(12)

(b) Insert(2)

(c) Delete(4)

Question 4 (2+4 Points)

(a) Suppose you are given a magical data structure $\mathbb{P}$ that supports the operations INSERT, MINIMUM, DELETE-MIN and INCREASE-KEY defined as usual. INSERT, MINIMUM and INCREASE-KEY run in amortized constant time. DELETE-MIN runs in amortized $\mathcal{O}(\log n)$ time.

Design an algorithm for sorting a list of n elements using $\mathbb{P}$ with running time $o(n \log n)$.

(b) A 0-1-matrix of size $n \times n$ is a *permutation matrix* if each row and column contains a single 1 entry. Let H be a 0-1-matrix of size $n \times n$ and suppose that each row and each column contains exactly k 1's.

Show that it is possible to represent H as the sum of k permutation matrices of size $n \times n$.

Hint: Think about matchings in bipartite graphs.

Question 5 (4 Points)

Suggest how to use a randomized skip list so that given a pointer to a node with key x, we can return a pointer to a node with key $y < x$ in $O(\log k)$ expected time where k is the distance between the nodes with values y and x in list L_0. Prove that your method works!

Question 6 (3+4 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.

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

(b) Describe how to perform an INSERT operation in Binomial Arrays. Analyze the worst-case running time of your algorithm. 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.

Question 7 (5+2 Points)

Programmer Paul wants to transfer each of his n applications from Python 7 to Python 8. As the new language has a much better performance, transferring application i saves him $b_i \geq 0$ Euros over a year. Compatibility problems, however, make it expensive to move only some of the applications: If only one of applications i and j moves to the new language, the extra cost over a year is $c_{ij} = c_{ji} > 0$ Euros.

(a) Unfortunately, Paul finds out that Application 1 cannot be transferred to Python 8 at all. Using a suitable maximum flow network, decide which applications (if any) should be transferred to Python 8 in order to maximize Paul's savings. Show that your solution is correct.

(b) In a more general scenario, the benefit b_i of moving to the new language might be negative, i.e., moving an application may incur cost. Again, use a suitable maximum flow network to decide which applications (if any) should be transferred. You do not need to show that your algorithm works correctly.

For both parts of the question, describe precisely how your network is constructed.

```
A word of a Gentleman is better than a proof,
but since you are not a Gentleman - please
provide a proof.
                                - Leonid A. Levin
```