Eidgenössische
Technische Hochschule
Zürich

Ecole polytechnique fédérale de Zurich
Politecnico federale di Zurigo
Federal Institute of Technology at Zurich

Institut für Theoretische Informatik
Peter Widmayer
Tobias Pröger

Exam
Datenstrukturen und Algorithmen
D-INFK

August 8, 2013

Last name, first name: _____

Student number: _____

With my signature I confirm that I was able to participate in the exam under regular conditions, and that I read and understood the notes below.

Signature: _____

Please note:

- You may not use any accessories except for a dictionary and writing materials.
- Please write your student number on **every** sheet.
- **Immediately** report any circumstances that disturb you during the exam.
- Use a new sheet for every problem. You may only give one solution for each problem. Invalid attempts need to be clearly crossed out.
- Please write **legibly** with blue or black ink. We will only grade what we can read.
- You may use algorithms and data structures of the lecture without explaining them again. If you modify them, it suffices to explain your modifications.
- You have 180 minutes to solve the exam.

Good luck!

Student number: _____

problem	1	2	3	4	5	**Σ**
max. score	8	10	13	9	10	50
$\sum$ score						

Problem 1.

Please note:

1) In this problem, you have to provide **solutions only**. You can write them right on this sheet.
2) If you use algorithms and notation other than that of the lecture, you need to **briefly** explain them in such a way that the results can be understood and checked.
3) We assume letters to be ordered alphabetically and numbers to be ordered ascendingly, according to their values.

1 P (a) Execute one pivoting step of *quicksort* on the following array (in-situ, i.e., without auxiliary array). Use the rightmost element of the array as pivot.

9	5	19	11	7	15	1	0	2	17	4	8
1	2	3	4	5	6	7	8	9	10	11	12

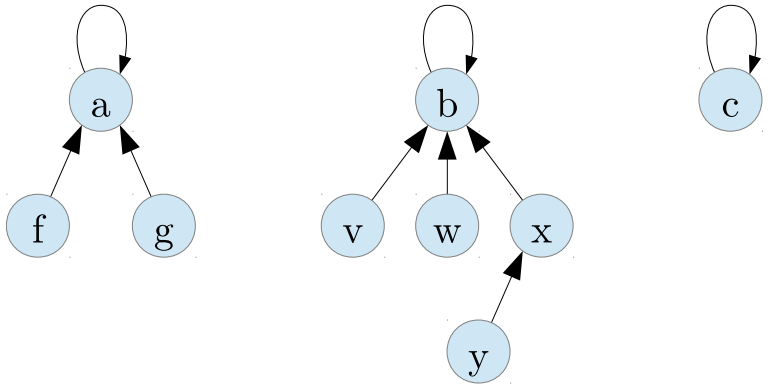
1	2	3	4	5	6	7	8	9	10	11	12

1 P (b) Draw an AVL tree with 7 nodes where *each* inner node has a balance factor different from 0.

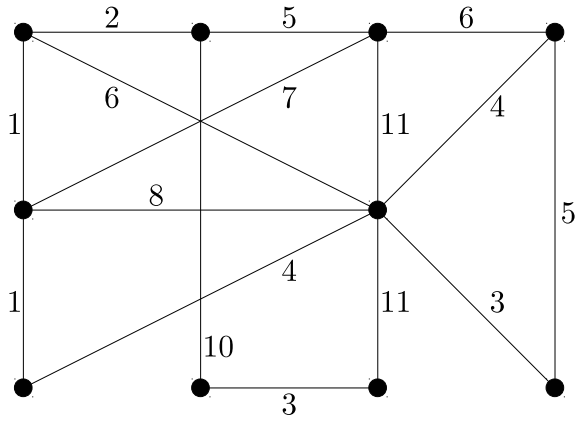
3 P (c) For each of the following statements, mark with a cross whether it is true or false. Every correct answer gives 0.5 points, for every wrong answer 0.5 points are removed. A missing answer gives 0 points. In overall the exercise gives at least 0 points. You don't have to justify your answer.

An inorder traversal of a binary search tree creates a sorted list of the stored keys.	☐ True	☐ False
If a sequence of m operations has a worst case running time of $\mathcal{O}(m)$, then each single operation has a worst case running time of $\mathcal{O}(1)$.	☐ True	☐ False
Let $G = (V, E)$ be a graph. Every subgraph of G with $\|V\| - 1$ edges is a spanning tree of G.	☐ True	☐ False
When a fibonacci heap contains n keys, the worst case time to extract the minimum is $\mathcal{O}(\log n)$.	☐ True	☐ False
Selection sort is a stable sorting algorithm.	☐ True	☐ False
On an array that is completely preordered, the running time of selection sort is linear.	☐ True	☐ False

1 P (d) On the following union-find data structure, execute first Union(a, c), and after that execute Union(Find$(f), b)$. Use "union by height", and draw the data structure that results after the two operations.



1 P (e) In the following weighted graph, mark the first edge that Kruskal's algorithm does *not* add to a minimum spanning tree.



1 P (f) Draw the binary search tree with the key set $\{1, \ldots, 8\}$ having a preorder traversal starting with $3, 2, 1, 5, 4$, and having a postorder traversal ending with $7, 8, 6, 5, 3$.

Problem 2.

1 P (a) Specify an **order** for the functions below, such that the following holds: If function f is left of function g, then $f \in \mathcal{O}(g)$.

Example: The three functions n^3, n^7, n^9 are already in a correct order, since $n^3 \in \mathcal{O}(n^7)$ and $n^7 \in \mathcal{O}(n^9)$.

- $\frac{n^2}{\log n}$
- $\binom{n}{2}$
- $n \log n$
- $n\sqrt{n}$
- $2^{\sqrt{n}}$
- $\log(n^2)$

3 P (b) Consider the following recursive formula:

$$T(n) := \begin{cases} 6 + 7T(n/3) & n > 1 \\ 6 & n = 1 \end{cases}$$

Specify a closed (i.e., non-recursive) form for $T(n)$ that is *as simple as possible*, and prove its correctness using mathematical induction.

Hints:

(1) You may assume that n is a power of 3.

(2) For $q \neq 1$, we have $\sum_{i=0}^{k} q^i = \frac{q^{k+1}-1}{q-1}$.

1 P (c) Specify (as concisely as possible) the asymptotic running time of the following code fragment in Θ notation depending on $n \in \mathbb{N}$. You do not need to justify your answer.

```
1 for(int i = 1; i <= n*n; i += 10) {
2         for(int j = 1; j*j <= n; j ++)
3                 ;
4 }
```

1 P (d) Specify (as concisely as possible) the asymptotic running time of the following code fragment in Θ notation depending on $n \in \mathbb{N}$. You do not need to justify your answer.

```
1 for(int i = n; i > 1; i = i/2) {
2         for(int j = 1; j <= n; j ++)
3                 ;
4 }
```

4 P (e) We consider a decimal counter that is initialized with 0, and that only supports one operation INCREASE that increases the value of the counter by 1. The cost of this operation is exactly the number of digits that are changed. For example, if the counter has value 18 and INCREASE is called, then the value of the counter is 19 after the operation, and the cost was 1 (only the 8 was changed). If the value of the counter is 19999 and INCREASE is called, then the value of the counter is 20000, and the cost of the operation was 5.

Show using amortized analysis that the amortized cost of the operation INCREASE is $\mathcal{O}(1)$. Define an appropriate potential function Φ_i, and analyse the real and the amortized costs if the last $k \geq 0$ digits of the counter have the value 9.

Problem 3.

A *palindrome* is a word whose meaning may be interpreted the same way in either forward or reverse direction, e.g. the word `RACECAR`. Formally, a palindrome is a sequence $\langle a_1, \ldots, a_n \rangle$ where either $n = 1$, or $a_1 = a_n$ and $\langle a_2, \ldots, a_{n-1} \rangle$ is a palindrome. Let $A[1 \ldots n]$ be an array storing a string of length n. A subarray $A[i \ldots j]$, $1 \leq i \leq j \leq n$, is called *palindrome in* A, if $\langle A[i], \ldots, A[j] \rangle$ is a palindrome.

Example: The array [`L`, `A`, `R`, `A`] contains the palindromes `A`, `R`, `L` and `ARA` (the palindrome `A` occurs twice). The array [`A`, `N`, `N`, `A`] contains the palindromes `A`, `N`, `NN` and `ANNA` (the palindromes `A` and `N` occur twice).

9 P (a) Let A be an array containing a string of length n. Describe a *dynamic programming* algorithm that outputs all pairs (i, j) where $\langle A[i], \ldots, A[j] \rangle$ is a palindrome. Address the following aspects in your solution.

1) What is the meaning of a table entry, and which size does the DP table have?
2) How can an entry be computed from the values of other entries? Distinguish between palindromes of length 1, 2, and longer palindromes.
3) In which order do the entries have to be computed?
4) How can the final solution be extracted once the table has been filled?

Example: For the input [`L`, `A`, `R`, `A`], the output consists of the pairs $(1, 1)$, $(2, 2)$, $(3, 3)$, $(4, 4)$, $(2, 4)$. For the input [`A`, `N`, `N`, `A`], the output consists of the pairs $(1, 1)$, $(2, 2)$, $(3, 3)$, $(4, 4)$, $(2, 3)$, $(1, 4)$. In the output, no special order of the pairs is required.

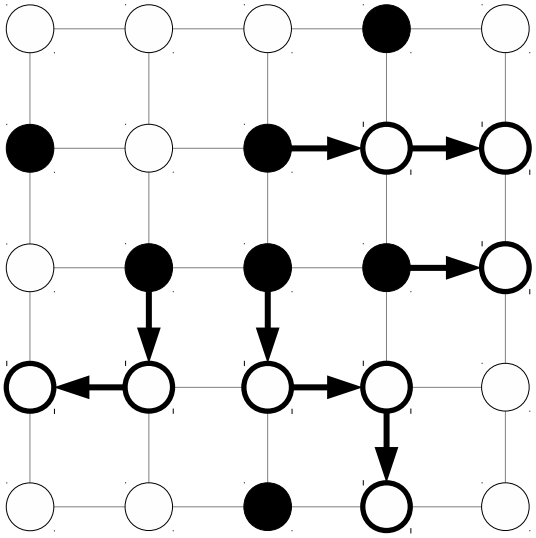
Notice that the exercise can easily be solved without dynamic programming by trivial enumeration of all palindromes in time $\mathcal{O}(n^3)$. We search for a more efficient algorithm.

1 P (b) Provide the running time of your solution.

3 P (c) Suppose that the algorithm of (a) computed the DP table already. Describe in detail, how a longest palindrome in A can be extracted from the DP table. Provide also the required running time.

Problem 4.

Consider an $n \times n$ grid, i.e. an undirected graph with n rows and n columns containing an edge between each pair of vertices that are neighbored in horizontal or vertical direction. The *escape route problem* asks to decide, given $m \leq n^2$ starting vertices $s_1, \ldots, s_m$, whether there exist m edge-disjoint paths from the starting vertices s_i to some boundary vertices. Notice that two paths may share one or more vertices, but no edges. In the following example, the starting vertices are pictured black, and the escape routes are pictured bold.



4 P (a) Model the above problem as a flow problem. Notice that we consider *directed* graphs in flow problems, but the grid itself is undirected. Describe in detail, how an appropriate network $N = (V, E, c)$ can be constructed, i.e., which vertices V and which edges E have to be defined, and which capacities have to be assigned to the edges. Describe how you can conclude from the value of a maximum flow whether or not there exist m edge-disjoint escape routes.

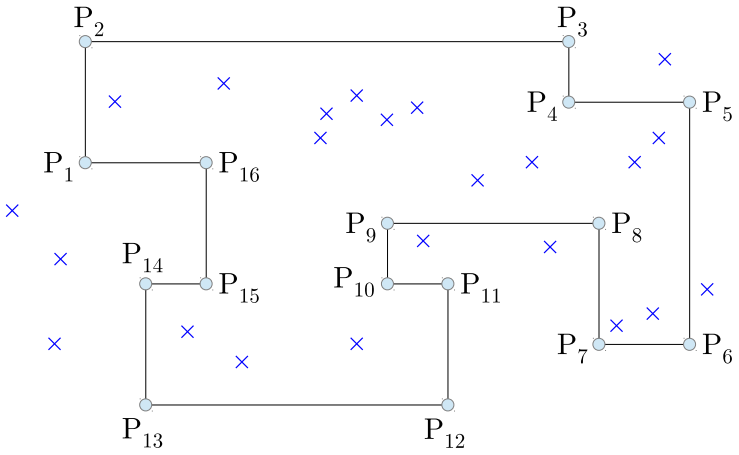
2 P (b) Specify a flow algorithm that solves (a) as efficient as possible. Provide the running time of the above-mentioned algorithm in dependency of n and m.

3 P (c) Now we want to decide whether there exists a set of m *vertex-disjoint* escape routes, where no pair of two paths shares a common vertex. Describe in detail how your solution from (a) has to be modified such that every vertex is used by at most one escape route. Does the running time of your solution change, and if yes, how?

Problem 5.

We consider a set of n cows and an enclosure consisting of m straight orthogonal fence segments. The position of each cow i, $1 \leq i \leq n$, can be determined using a GPS receiver, and is given by the point $K_i \in \mathbb{Q}^2$. The enclosure is a closed simple polygon with the vertices (i.e., the fence pickets) $P_i \in \mathbb{Q}^2$, $1 \leq j \leq m$. We assume that for $i \in \{1, \ldots, m-1\}$, P_i and P_{i+1} and additionally P_m and P_1 are connected by a fence segment. Furthermore, let P_1 be the leftmost point (if there exist more such points, then let P_1 additionally be the bottommost among these). You can assume that no cow is located directly on a fence segment, i.e. no point K_i is lying on an edge of the polygon. We want to determine how many cows are staying inside the enclosure.

9 P (a) We assume that all fence segments are parallel to the x- or the y-axis. Provide a scanline algorithm that obtains as input the positions of the cows K_i as well as the positions of the fence pickets P_j, and that computes the number of cows inside the enclosure.



1 P (b) Provide the running time of your algorithm.