Albert-Ludwigs-Universität
Institut für Informatik
Prof. Dr. F. Kuhn

# Exam Algorithm Theory

Monday, August 25, 2014, 14:00 – 15:30

Name: .....

Matriculation Nr.: .....

Signature.: .....

**Do not open or turn until told so by the supervisor!**

## Instructions:

- Write your name and matriculation number on the cover page of the exam and sign the document! Write your name on all sheets!
- Your signature confirms that you have answered all exam questions without any help, and that you have notified exam supervision of any interference.
- Write legibly and only use a pen (ink or ball point). Do **not** use **red**! Do **not** use a pencil!
- You are **not** allowed to use any material except for a dictionary and a hand-written summary of at most 5 A4 pages (corresponds to 5 single-sided A4 sheets!).
- There are 5 problems (with several questions per problem) and there is a total of 90 points. At most 40% are needed to pass the exam, and 80% will net you the best grade, i.e., 18 points are bonus points.
- Use a separate sheet of paper for each of the 5 problems.
- Only one solution per question is graded! Make sure to strike out any solutions that you do not want to be considered!
- **Explain your solutions! Just writing down the end result is not sufficient.**

| Question | Achieved Points | Max Points |
|---|---|---|
| 1 | | 19 |
| 2 | | 18 |
| 3 | | 16 |
| 4 | | 22 |
| 5 | | 15 |
| Total | | 90 |

## Problem 1: Short Questions (19 points)

1. (5 points) We have seen that for Fibonacchi Heaps the operation *decreaseKey* has amortized running time $\mathcal{O}(1)$. What does this mean?

2. (7 points) Show that taking all nodes is a 2-approximation algorithm for the vertex cover problem on regular graphs (graphs where all nodes have the same degree).

   Recall: For a graph $G = (V, E)$, a set $S \subseteq V$ is called a vertex cover iff every edge $\{u, v\} \in E$ is covered by a node in $S$, i.e., if the intersection $S \cap \{u, v\}$ is non-empty. The vertex cover problem tries to find a minimum (in cardinality) vertex cover.

3. (7 points) You are given an unsorted array $A$ of integers. Your task is to find an element $e$ such that at least $40\%$ of all elements are smaller than or equal to $e$ and at least $40\%$ of all elements are larger than or equal to $e$. Devise an algorithm that outputs such an element $e$ with probability at least $1/2$, and otherwise just outputs "failure". The algorithm is supposed to have linear running time (in the worst case).

## Problem 2: Coins (18)

We are looking for an algorithm to determine the minimum number of coins $A$ which are necessary to pay a specific amount of money $B$ (the amount has to be paid exactly).

More explicitly: We are given $k$ different integer coin values $m_1, \ldots, m_k \in \mathbb{N}$, $k > 0$, $m_i \geq 1$ and an integer amount $B \in \mathbb{N}$. We want to minimize the number of coins $A := \sum_{i=1}^{k} a_i$ where $a_1, \ldots, a_k \in \mathbb{N}_0$ are integers with $\sum_{i=1}^{k} a_i m_i = B$.

A diligent student implemented the following algorithm:

```
int minimum(int[] M,  int B) {
    sort(M);        // sorts M in decreasing order
    int i = 0, c = 0;
    while (B > 0) {
        if (B < M[i]) {
            i++;
            if (i >= M.length) {
                return -1;      // no solution found
            }
        } else {
            B = B - M[i];
            c++;
        }
    }
    return c;
}
```

1. (4 points) Which algorithmic design principle does the above algorithm use?
2. (5 points) Give an example configuration of coin values $m_1, \ldots, m_k$, and a value $B$, where a solution exists, but the above algorithm does not find an optimal solution.
3. (9 points) Describe a dynamic programming algorithm that finds the minimum number of coins needed to pay $B$. What is the running time of your algorithm?

## Problem 3: TV Show (16 points)

A high school class is made an interesting offer by a reality TV show in which couples of students (one female and one male student) get a free vacation trip to an exciting location. This class consists of

- $n$ boys $B = \{b_1, \ldots, b_n\}$ and
- $n$ girls $G = \{g_1, \ldots, g_n\}$.

There are $n$ different locations $L = \{\ell_1, \ldots, \ell_n\}$ for the students to choose from. The girls are not picky about the destinations, but each girl $g$ is only willing to partner up with an individual subset $B_g \subseteq B$ of all available boys. The boys on the other hand do not care that much about with whom they go on vacation, but they care about the location; each boy $b$ has an individual subset $L_b \subseteq L$ of locations it is willing to visit.

Is it possible that everyone can go on a free vacation? Devise an algorithm that answers this question.

What is the time complexity of your algorithm if you assume that each girl is willing to partner up with at most $\sqrt{n}$ different boys and if you assume that each boy is willing to to visit at most $\sqrt{n}$ different locations?

## Problem 4: Sorted Matrix (22 points)

As input you are given an $n \times n$ square matrix $(F_{i,j})_{1 \le i,j \le n}$ of integers and an integer $x$. The matrix has a special property: Elements in every row and column are sorted in increasing order, i.e., $F_{i,j} \le F_{i',j}$ for $i \le i'$ and $F_{i,j} \le F_{i,j'}$ for $j \le j'$.

The task is to find out whether $x$ is one of the matrix elements.

The naïve approach needs time $\Theta(n^2)$ and therefore we devise a divide-and-conquer algorithm $\mathcal{A}$ for the problem. Using the intuition of the binary search algorithm for one-dimensional arrays, we compare $x$ with the center element $c(F)$ of the matrix, where $c(F) := F_{\lceil \frac{n}{2} \rceil, \lceil \frac{n}{2} \rceil}$ and we then solve the problem by recursively solving it in smaller square matrices. (You can ignore any rounding issues when dividing the problem into smaller ones. You can also assume that when solving the problem recursively, all involved smaller matrices are square matrices of exactly the same size.)

1. (7 points) Describe Algorithm $\mathcal{A}$ in more detail. (It should become clear how the problem is divided into subproblems and which of the subproblems have to be solved recursively.)
2. (5 points) Analyze the running time of Algorithm $\mathcal{A}$. You can use the master theorem.
3. (10 points) As it turns out, Divide-and-Conquer is not an efficient approach to solve the problem. Therefore forget about Algorithm $\mathcal{A}$ and try to find a better algorithm. State your algorithm and explain its complexity. Full points are awarded for an iterative algorithm that runs in time $\mathcal{O}(n)$, and partial scores for an algorithm that runs in time $\mathcal{O}(n \log n)$.

## Problem 5: Binomial Heap(15 points)

You are given a Binomial Heap where (as discussed in the lecture) each binomial tree satisfies the min-heap property.

1. (9 points) You need to extend the data structure by providing an operation

   $$\textit{increaseKey}(x, v)$$

   to *increase* the key of an element $x$ by value $v$.
   The natural way to solve this task is to increase the key of the element and to afterwards 'swap' the element down the tree until the heap property is satisfied. Explain this natural algorithm in more detail and analyze its running time.

2. (3 points) How can one delete an element $x$ from a binomial heap? What is the running time?

3. (3 points) Describe a different solution for the task of increasing a key in a binomial heap that runs in time $\mathcal{O}(\log n)$.