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

Exam Algorithm Theory

Tuesday, August 25, 2015, 9:00–10:30, 101-01-009/13

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% (36 points) are needed to pass the exam and 80% (72 points) 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 unless otherwise indicated.**

Question	Achieved Points	Max Points
1		21
2		18
3		15
4		15
5		21
Total		90

Problem 1: Short Questions (21 points)

1. (4 points) A data structure supports an operation *foo* such that a sequence of n operations *foo* takes $\Theta(n \log n)$ time to perform in the worst case.

   What is the amortized time of an *foo* operation?

   How large can the actual time of a single *foo* operation be?

2. (4 points) Assume that you are given a parallel algorithm with $\Theta(n^{3/2})$ work (total number of operations) and $\Theta(\sqrt{n})$ span (critical path length). What is the optimal asymptotic running time of this algorithm when you have p processors available (as a function of p and n)?

3. (6 points) In class, we have seen Prim's algorithm to compute a minimum spanning tree (MST). Let us assume that instead of a minimum spanning tree, we want to compute a maximum spanning tree, i.e., a spanning tree of maximum total weight. Can the idea of Prim's algorithm be used to compute such a maximum spanning tree? Explain your answer!

4. (7 points) Assume that we are given an undirected, unweighted graph $G = (V, E)$ and some cost function $c : E \to \mathbb{R}_{>0}$ assigning a positive cost $c(e)$ to each edge $e \in E$. Your goal is to compute a minimum cut of G of minimum cost (i.e., a minimum cost cut among the cuts consisting of the smallest number of edges). Give an algorithm which with high probability finds such a minimum cost minimum cut in polynomial time. What is the running time of your algorithm?

   Hint: Recall the properties of the randomized minimum cut algorithm of the lecture.

Problem 2: Longest Alternating Subsequence (18 points)

Consider a sequence $A = a_1, a_2, a_3, \ldots, a_n$ of integers. A subsequence B of A is a sequence $B = b_1, b_2, \ldots, b_m$ which is created from A by removing some elements but by keeping the order. For example, the integer sequence $B = 5, 1, 8, 19, 7, 4, 5, 5$ is a subsequence of the sequence $A = 1, 5, 3, 1, 7, 8, 14, 15, 19, 8, 7, 4, 5, 1, 5$. Given an integer sequence A, the goal is to compute an *alternating subsequence* B, i.e., a sequence $B = b_1, \ldots, b_m$ such that for all $i \in \{2, \ldots, m-1\}$, if $b_{i-1} < b_i$ then $b_i > b_{i+1}$ and if $b_{i-1} > b_i$ then $b_i < b_{i+1}$.

a) (11 points) Give a dynamic programming algorithm which finds a longest alternating subsequence. What is the asymptotic running time of your algorithm?

b) (7 points) Now, consider an online version of the problem, where the sequence A is given element-by-element and each time, one needs to directly decide whether to include the next element in the subsequence B. Is it possible to achieve a constant competitive ratio (by using a determinstic online algorithm)? Either give an online algorithm which achieves a constant competitive ratio or show that it is not possible to find such an online algorithm.

Problem 3: Dominating Sets in Unit Interval Graphs (15 points)

Given a graph $G = (V, E)$, a dominating set $S \subseteq V$ of G is a subset of the nodes of G such that for every node $u \notin S$, there is a neighbor $v \in S$. That is, a dominating set S is a set of nodes such that all nodes of G are within distance at most 1 of S. In the *minimum dominating set* problem, the goal is to find a dominating set of minimum cardinality.

In this problem, you need to devise an algorithm to compute a minimum dominating set in so-called *unit interval graphs*. In a unit interval graph, each node $v \in V$ is associated with a real number $x_v \in \mathbb{R}$. Two nodes $u, v \in V$ are connected by an edge if and only if $|x_v - x_u| \leq 1$. Give a greedy algorithm for the minimum dominating set problem for unit interval graphs and prove that your algorithm computes an optimal (i.e., minimum cardinality) dominating set.

Problem 4: Union-Find Data Structure (15 points)

We are given a union-find data structure that uses the union-by-size heuristic and path compression. Starting from an empty data structure, perform the following operations. Draw the forest representing the state of the data structure after each block of operations (in total, you need to draw 6 forests).

Whenever in a union operation, the union-by-size heuristic does not uniquely determine how to combine two sets, perform the union operation so that the smaller key becomes the new root node.

a)
```
for i = 1 to 16 do
    MakeSet(x[i])
for i = 1 to 15 by 2 do
    Union(x[i],x[i+1])
```

b)
```
for i = 1 to 13 by 4 do
    Union(x[i],x[i+2])
```

c)
```
Union(x[1],x[5])
Union(x[5],x[13])
```

d)
```
Union(x[10],x[3])
```

e)
```
Find(x[4])
```

f)
```
Find(x[12])
```

Problem 5: Almost Disjoint Paths (21 points)

We are given an **undirected** graph $G = (V, E)$, two non-empty disjoint nodes sets $S \subset V$ and $T \subset V$ ($S, T \neq \emptyset$, $S \cap T = \emptyset$), and an integer parameter $a \geq 1$. The goal is to find a largest possible set of paths $P_1, \ldots, P_k$ such that each path P_i connects a node in S with a node in T and each edge of G is used at most by a paths.

a) (7 points) Describe in detail how to formulate the given problem as a maximum flow problem (i.e., how to construct a flow network that allows to solve the given problem).

b) (5 points) Describe how to convert a solution of the constructed maximum flow problem into a set of paths.

c) (5 points) Now, you are given an additional integer parameter $b \geq 1$. Explain how to modify your solution such that every node in G is used by at most b of the paths.

d) (4 points) Which algorithm would you use to solve the maximum flow problem?
Assume your algorithm finds k such paths. Try to give a best possible asymptotic estimate running time of your algorithm as a function of k, the number of nodes n and the number of edges m.