University of Freiburg
Dept. of Computer Science
Prof. Dr. F. Kuhn

Algorithm Theory

20th of August 2024, 14:00-16:00

Name: .....

Matriculation No.: .....

Signature: .....

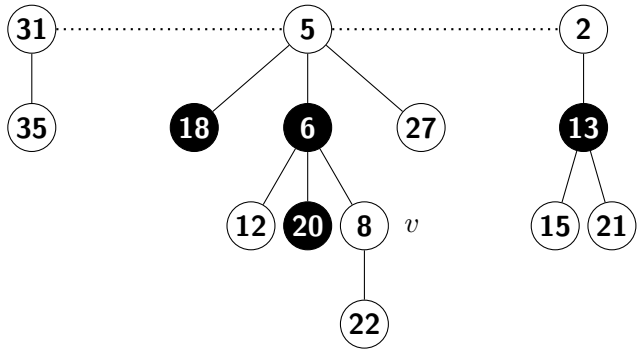
Do not open or turn until told so by the supervisor!

- Write your **name** and **matriculation number** on this page and **sign** the document.
- Your **signature** confirms that you have answered all exam questions yourself without any help, and that you have notified exam supervision of any interference.
- You are allowed to use a summary of **six handwritten A4 pages**.
- **No electronic devices** are allowed.
- Write legibly and only use a pen (ink or ball point). Do **not use red**! Do **not use a pencil**!
- You may write your answers in **English or German** language.
- Only **one solution per task** is considered! Make sure to strike out alternative solutions, otherwise the one yielding the minimal number of points is considered.
- **Detailed steps** might help you to get more points in case your final result is incorrect.
- The keywords **Show...**, **Prove...**, **Explain...** or **Argue...** indicate that you need to prove or explain your answer carefully and in sufficient detail.
- The keywords **Give...**, **State...** or **Describe...** indicate that you need to provide an answer solving the task at hand but without proof or deep explanation (except when stated otherwise).
- You may use information given in a **Hint** without further explanation.
- **Read each task thoroughly** and make sure you understand what is expected from you.
- **Raise your hand** if you have a question regarding the formulation of a task or if you need additional sheets of paper.
- A total of **45 points** is sufficient to pass and a total of **90 points** is sufficient for the best grade.
- Write your name on **all sheets**!

Task	1	2	3	4	5	6	**Total**
Maximum	36	13	18	18	13	22	120
Points							

Task 1: Short Questions *(36 Points)*

(a) Consider the following Fibonacci heap (black nodes are marked, white nodes are unmarked). How does the given Fibonacci heap look after `decrease-key(v, 3)` operation and how does it look after a subsequent `delete-min` operation? *(8 Points)*



(b) Let p be a polynomial of degree n and q a polynomial of degree $m < n$. For simplicity, you can assume that n is an integer multiple of m, i.e., $n = k \cdot m$ for some integer $k > 1$. Give an algorithm that finds the product of p and q in $O(n \cdot \log m)$ time. *(8 Points)*
Hint: You may use the multiplication algorithm for the case $n = m$ from the lecture as a subroutine.

(c) We have M children labeled from 1 to M and we have N sweets labeled from 1 to N. We have A_i pieces of sweet i. We now want to distribute all the pieces of sweets among all the children, however, the following conditions should be satisfied for every child j and every sweet i:

- Child j gets at most $B_{i,j}$ pieces of sweet i.
- Child j gets at most C_j pieces of sweets in total.

Transform the problem into a maximum flow problem such that a maximum integer flow directly gives a feasible solution for the problem (if such a solution exists). *(6 Points)*

(d) We want to divide the vertices of a graph into two groups such that the number of edges between the two groups is maximized. Provide a polynomial-time 1/2-approximation algorithm for this task and prove its correctness. You can either give a deterministic algorithm that always achieves an approximation ratio of at least 1/2 or you can give a randomized algorithm that achieves an approximation ratio of at least 1/2 in expectation. *(6 Points)*

(e) Assume that there are n chess players $1, \ldots, n$ that you need to pair up for playing against each other in a chess tournament.

There are some players who must play their next game with the white pieces and there are some players who must play their next game with the black pieces. There are also players for whom it does not matter if they play with the white or the black pieces.

In addition, each player i has a rating value r_i, which is a positive integer.

Each chess game in the tournament must be played between exactly two players: one playing with the white pieces and the other with the black pieces. Further, each player should play in at most one game. Additionally, to ensure balanced games, the absolute difference in rating between the two players in a game must be smaller than 100.

Describe a polynomial-time algorithm to determines a largest possible set of chess games that can be arranged with the available n players. You can use algorithms from the lecture as a black box (but you must explain which algorithms you use). *(8 Points)*

Bonus Question: You can gain up to 6 bonus points if you can answer the following question. Assume that we make the (strange) requirement that the absolute difference in rating between the two players of a game must be an odd number < 100? Argue why the problem of determining a maximum set of possible chess games now becomes easier! *(6 Points)*

Solution Task 1

Task 2: Cuts *(13 Points)*

Let $G = (V, E)$ be a flow network with source $s \in V$, sink $t \in V$, and *integer capacities* $c_e \geq 0$ for all $e \in E$. Give a new flow network $G' = (V, E)$ with the same nodes and edges as G and the same source and sink nodes s and t as G, but with different *capacities* $c'_e \geq 0$. The capacities c'_e should be chosen such that such that any minimum s-t-cut in G' is a minimum s-t-cut in G with the smallest possible number of edges (among all minimum s-t-cuts in G). Prove that your capacities c'_e satisfy this property!

Hint: The capacities c'_e do not need to be integers. You may consider the new capacities c'_e as a function of c_e and the number of edges/nodes of the graph.

Solution Task 2

Task 3: Greedy Job Scheduling *(18 Points)*

Given n jobs $1, \ldots, n$, each with a processing time $p_i > 0$ and a weight $w_i > 0$, determine an ordering π of the jobs such that when processing the jobs in this order, to weighted sum of completion times (or equivalently, the weighted average completion time) is minimized.

More formally, for any $i, j \in \{1, \ldots, n\}$, we set $\pi(i) = j$ iff job j is at the i^{th} position according to the ordering π and we have to minimize the objective function:

$$\min_{\pi} \sum_{i=1}^{n} w_{\pi(i)} \cdot C_{\pi(i)},$$

where the completion time $C_{\pi(i)}$ is defined as

$$C_{\pi(i)} = \sum_{j=1}^{i} p_{\pi(j)}$$

For example, given 3 jobs with processing times and weights as pairs (p_i, w_i):

$$(2, 3), \quad (1, 5), \quad (3, 2)$$

The optimal schedule is $2, 1, 3$ (i.e., $\pi(1) = 2, \pi(2) = 1, \pi(3) = 3$) with an objective value of 26.

(a) To start, we first look at the problem, where the weight $w_i = 1$ for all jobs i. In this case, we therefore need to minimize the sum of all completion times. *(6 Points)*
Hint: Try to write the sum of completion time as a sum in terms of the n processing times $p_1, \ldots, p_n$.

(b) Now, the weights w_i can take any positive real-valued number. *(12 Points)*
Hint: For every job, look at the ratio of its two values.

In both cases, solve the problem in time $O(n \log n)$ and prove that your solution is correct.

Solution Task 3

Task 4: Online Bin Packing *(18 Points)*

The Online Bin-Packing problem is a variant of the Knapsack problem. We are given an *unlimited number of bins* labeled $1, 2, \ldots$, each *of capacity* 1. Further, we get a *sequence of items* $i = 1, 2, \ldots$ of size $s_i \in [0, 1]$ in an *online fashion.*

Whenever a new item i arrives, we are required to place the item in a bin that still has enough space (the sizes of the items assigned to each bin must sum up to at most 1).

Our goal is to *minimize* the number of bins we use.

In this question we consider a simple online algorithm for this problem called **First-Fit**. Recall that the bins are ordered (they are labeled $1, 2, \ldots$). First-Fit places each item into the *first* bin that has enough space to hold the item.

a) Show that First-Fit has a competitive ratio of at least $3/2$.
Hint: There is a sequence consisting of only 4 items for which First-Fit uses $3/2$ *as many bins as an optimal offline algorithm.* *(4 Points)*

b) Show that no deterministic online algorithm can have a competitive ratio better than $3/2$. *(6 Points)*

c) Prove that First-Fit has a competitive ratio of at most 2.
Hint: We do not require a strict competitive ratio here. *(8 Points)*

Solution Task 4

Task 5: Dynamic Programming *(13 Points)*

Suppose you are given a graph $G = (V, E)$, where each node $v \in V$ is labeled with an elevation value $h(v) \in \mathbb{N}$. You can safely traverse an edge $(u, v) \in E$ from node u to node v if and only if the absolute difference between the elevation values of u and v is at most δ, that is, $|h(u) - h(v)| \leq \delta$, where $\delta > 0$ is an integer parameter.

Our goal now is to find a longest possible walk in the graph such that the walk starts at some node $s \in V$ and ends at another node $t \in V$. The walk should consist of two segments: an uphill segment, where the elevation must strictly increase in each step, followed by a downhill segment, where the elevation must strictly decrease in each step.

Note that a walk in a graph is path on which nodes are allowed to repeat. Note however that in the uphill segment, the elevation values are strictly increasing and in the downhill segment, the elevation values are strictly decreasing. Each node can therefore appear at most once in the uphill segment and at most once in the downhill segment. A node can however appear in the uphill and in the downhill segment.

Your input consists of the graph $G = (V, E)$, the elevation function $h : V \to \mathbb{N}$, the starting node s, the target node t, and the parameter $\delta > 0$. Your task is to find a longest possible walk as described above or determine that no such walk exists.

Give an algorithm that solves the problem in time $O(nm)$ and argue why your algorithm is correct (where as usual $n = |V|$ and $m = |E|$).

Hint: It makes sense to first solve the following problem: For every $v \in V$, find the longest path from s to v on which the elevation values strictly increase or determine that no such path from s to v exists.

Solution Task 5

Task 6: Independent Sets *(22 Points)*

Let $G = (V, E)$ be a graph on n nodes with maximum degree Δ. Recall that an *independent set* is a set $I \subseteq V$ such that no two nodes in I are connected by an edge in E. The goal in this part is to get a relatively large independent set of G. Consider the following parallel randomized algorithm.

Algorithm 1 `IndependentSet(G)` ▷ `Given` $G = (V, E)$

1: $I := \phi$
2: Each node $v \in V$ in parallel picks an integer value R_v independently and uniformly at random from the set $\{1, 2, ..., n^{c+2}\}$, for some parameter $c > 0$.
3: **for** all $v \in V$ **do**
4: **if** $R_v < R_u$ for all neighbors u of v **then**
5: $I := I \cup \{v\}$
6: Output I

(a) Show that all nodes v succeed in choosing a unique value R_v in step 2 of the algorithm (i.e., no two nodes pick the same value) with probability at least

$$1 - \frac{1}{n^c}.$$

(8 Points)

(b) In the following, we assume that the values R_v are always chosen to be unique (e.g., step 2 is repeated until the values are unique). Show that in this case, the expected size of I is

$$\mathbb{E}[|I|] \geq \frac{n}{\Delta + 1}.$$

(6 Points)

(c) The goal now is to show that the lower bound on the size of I only holds in expectation and not with high probability. More specifically, show that there exists a constant $p > 0$ (p is independent of n) such that for any $n \geq 6$, there exists a graph G_n with n nodes for which

$$\Pr\left(|I| < \frac{n}{\Delta + 1}\right) \geq p$$

(8 Points)

Hint: For simplicity assume that n is an integer multiple of 6. You can choose G_n as a graph consisting of six cliques $C_1, \ldots, C_6$ of size $n/6$ such that for any $i \in \{1, \ldots, 5\}$, cliques C_i and C_{i+1} are connected by a complete bipartite graph. Try to understand which nodes are in the independent set I and in which case, I only consists of a single node.

Solution Task 6