Exam in Algorithms and Advanced Data Structures (1DV516)

2023-12-15

The exam consists of 6 problems, each worth 10 points. The questions are not arranged in order of difficulty. You may answer in Swedish or English. Illegible answers are not corrected!

- Grade A: 55 - 60 points
- Grade B: 49 - 54 points
- Grade C: 43 - 48 points
- Grade D: 37 - 42 points
- Grade E: 30 - 36 points

When an algorithm is asked for, it should be understood that it should be as efficient as possible (unless otherwise stated), and it should be expressed in such a way that it is understandable.

Problem 1 (2 + 2 + 2 + 2 + 2 = 10 p)

Indicate whether the following statements are *true* or *false*. Motivate your answers. Incorrect answers are worth -1 point, correct answers without motivation 0 points, and correct answers with a correct motivation 2 points. Note that you cannot get less than 0 points for the question.

- Finding a Hamilton circuit (cycle) in a DAG is an NP-Complete problem.
- There can be two strings in Java that are not equal but have the same `hashCode()`.
- Adding a constant to every edge weight in an undirected graph can change the set of edges that belong to the minimum spanning tree. Assume unique weights.
- Adding a constant to every edge weight in a directed graph can change the set of edges that belong to a shortest paths tree. Assume unique weights.
- We can design a compare-based algorithm to merge two sorted arrays, each of length $N/2$, into a sorted array of length N that makes $\sim 0.5N$ compares in the worst case.

Problem 2 $(3 + 2 + 3 + 2 = 10$ p)

1. What does it mean that a sorting algorithm is stable? Give examples of a stable and a not stable algorithm and explain.
2. What is the worst-case time complexity of Quicksort if median of three is used to determine pivot? Motivate.
3. Quicksort, Mergesort, and Heapsort have similar time complexities in the average case, but Quicksort is generally faster in practice. Why?
4. Why can shellsort with the right gap sequence have a lower time complexity than $O(N^2)$ in the average case?

Problem 3 $(3 + 2 + 3 + 2 = 10$ p)

1. Implement `add` and `delMax` for a (max) 3-heap. Show how they work by inserting the numbers 1–10 in order and removing the the max twice.
2. What are the time complexities for the two operations from #1? Motivate.
3. Explain top-down and bottom-up dynamic programming. Illustrate the differences with an example. You can, for example, compute the Fibonacci sequence.
4. What is a greedy algorithm and when would you use one?

Problem 4 $(3 + 3 + 4 = 10$ p)

1. Explain Prim's algorithm for computing an MST. What is the time complexity? Use an example graph to explain how the algorithm works.
2. What is the time complexity required to compute the shortest and longest paths in a graph. Motivate your answers and sketch the algorithms.
3. Given an edge-weighted graph G and an edge e, design a linear-time algorithm to determine whether e appears in an MST of G. For simplicity, assume that G is connected and that all edge weights are distinct.

Problem 5 (5 + 2 + 3 = 10 p)

1. Two strings s and t are cyclic rotations of one another if they have the same length and s consists of a suffix of t followed by a prefix of t. For example, "shellsort" and "sortshell" are cyclic rotations. Given N distinct strings, each of length L, design an efficient algorithm to determine whether there exists a pair of distinct strings that are cyclic rotations of one another.
2. What is the difference between an AVL tree and a Splay tree?
3. Explain the rotations used to balance an AVL tree.

Problem 6 (3 + 2 + 5 p)

1. Assume that you have a hashtable of length 10 that uses linear probing. After inserting six keys using the hash function $h(x) = x \% 10$, you get the hash table in Figure 1. List the possible orders that the keys could have been inserted in.
2. Explain the uniform hashing assumption and why it is important for hashing.
3. You are asked to implement a Least-Recently-Used (LRU) cache with two operations: `cache(x)` and `inCache(x)`. The cache has a capacity, N. If the cache is full, the least recently used value should be removed. `cache(x)` indicates that `x` was used if it is in the cache. If it is not in the cache, it should be added. If the cache is full, the least recently used value should be removed to make room for `x`. `inCache(x)` checks if the value `x` is in the cache but does not indicate that it was used. Give an implementation of the two operations that run in constant time.

Index	0	1	2	3	4	5	6	7	8	9
Values			42	23	34	52	46	33		

Figure 1: A hash table