Exam in Algorithms and Advanced Data Structures (1DV516)

2023-10-31

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.

1. In a BST, we can find the next smallest element to a given element in $O(1)$ time.
2. AVL trees can be used to implement an optimal comparison-based sorting algorithm.
3. Every directed acyclic graph has a node with no incoming edges.
4. In a connected weighted graph, the edge with maximum weight is never in the minimum spanning tree.
5. Radix sort runs correctly when using any correct sorting algorithm to sort each digit.

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

1. How long will Depth-first search take in the worst case if the graph uses an adjacency matrix. Can we improve this by using an adjacency list?
2. Given a graph with tasks as vertices and their interdependencies as edges, we can use depth-first to compute a valid order to tackle the tasks. Can we use breadth-first instead? Motivate.
3. Can Dijkstra's shortest-path algorithm relax an edge more than once in a graph with a cycle? Motivate.
4. How can we use Djikstra's algorithm to find a minimum spanning tree in a weighted graph G where all edges have weight 1? Motivate.

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

1. You are running a library catalog. You know that the books in your collection are almost in sorted ascending order by title, with the exception of one book which is in the wrong place. You want the catalog to be completely sorted in ascending order. Which of the following algorithms would you pick: Insertion Sort, Merge Sort, Radix Sort, or Heap Sort? Why?
2. You are working on an embedded device (an ATM) that only has 4KB (4,096 bytes) of free memory, and you wish to sort the 2,000,000 transactions withdrawal history by the amount of money withdrawn (discarding the original order of transactions). Which of the following algorithms would you pick: Insertion Sort, Merge Sort, Radix Sort, or Heap Sort? Why?
3. To determine which of your Facebook friends were early adopters, you decide to sort them by their Facebook account ids, which are 64-bit integers. (Recall that you are super popular, so you have very many Facebook friends.) Which of the following algorithms would you pick: Insertion Sort, Merge Sort, Radix Sort, or Heap Sort? Why?
4. If you could use any sorting algorithm discussed in the course for questions 1-3, which would you swap and why?
5. Insertion Sort, Selection Sort, and Bubble Sort all have a worst-case complexity of $O(n^2)$ but their actual runtime performance differs for the same input. Why? Which one is generally faster in practice.

Problem 4 (4 + 6 = 10 p)

You are planning a dinner for a list of guests V.

1. Suppose you are also given a lookup table T where $T[u]$ for $u \in V$ is a list of guests that u knows. If u knows v, then v knows u. You are required to arrange the seating such that any guest at a table knows every other guest sitting at the same table either directly or through some other guests sitting at the same table. For example, if x knows y, and y knows z, then x, y, z can sit at the same table. Describe an efficient algorithm that, given V and T, returns

the minimum number of tables needed to achieve this requirement. Analyze the running time of your algorithm.

2. Now suppose that there are only two tables, and you are given a different lookup table S where $S[u]$ for $u \in V$ is a list of guests who are on bad terms with u. If v is on bad terms with u, then u is on bad terms with v. Your goal is to arrange the seating such that no pair of guests sitting at the same table are on bad terms with each other. Describe an efficient algorithm that, given V and S, returns *TRUE* if you can achieve the goal or *FALSE* otherwise. Analyze the running time of your algorithm.

Problem 5 (2 + 2 + 6 = 10 p)

1. Can you find the next largest element of any element in a min-heap in $O(\log n)$ time? Motivate.
2. Suppose a dynamic programming algorithm creates an $n \times m$ table and to compute each entry of the table it takes a minimum over at most m (previously computed) other entries. What would the running time of this algorithm be, assuming there is no other computations.
3. Assume a game with n bills of various denominations in a list $S[1 \dots n]$ where the i_{th} bill has value $S[i]$. On a player's turn, they may take either the first bill or the last bill in the list, and then it becomes the other player's turn. The goal is to collect as much money as possible. Create an algorithm that plays the game optimally. What is the complexity of the algorithm?

Problem 6 (2 + 2 + 6 = 10 p)

Consider the longest increasing subsequence problem defined as follows. Given a list of numbers $a_1, \dots, a_n$, an increasing subsequence is a list of indices $i_1, \dots, i_k \in 1, \dots, n$ such that $i_1 < i_2 < \dots < i_k$ and $a_{i_1} \leq a_{i_2} \leq \dots \leq a_{i_k}$. The longest increasing subsequence is the longest list of indices with this property.

1. What is the longest increasing subsequence of the list $5, 3, 4, 8, 7, 10$?
2. Consider a greedy algorithm that chooses the first element of the list, and then repeatedly chooses the next element that is larger. Is this a correct algorithm? Motivate.
3. Design a dynamic programming algorithm for longest increasing subsequence. Show its correctness and analyze its running time.