COMS 4231: Analysis of Algorithms I, Fall 2021

Problem Set 3, due Friday October 29, 11:59pm on Gradescope.

Please follow the homework submission guidelines posted on Courseworks.

In all problems that ask you to give an algorithm, include a justification of the correctness of the algorithm and of its running time.

All time bounds refer to deterministic worst-case complexity, unless specified otherwise.

Problem 1. [15 points] Given a min-heap A, in the form of an array, and another key x (which may or may not be in A), we want to find all the elements of A that are smaller than x, and print them; the elements can be printed in any order (not necessarily in sorted order). For example, if A = [4 7 5 9 8 5] and x=8, then the output should consist of 4, 7, 5, 5 (printed in any order).
Give an algorithm for this problem that runs in O(k) time, where k is the number of elements of A that are smaller than x.
(*Hint*: Consider the tree representation of the heap, and show that if an element is in the output, then its parent must also be in the output.)

Problem 2. 15 points] Give an algorithm which takes as input a binary search tree T with distinct keys and two keys x, y (which may or may not be in T), and outputs in sorted order all the elements of T whose key is in the (closed) interval [x,y], i.e., such that their key satisfies $x \leq$ key $\leq y$. The algorithm should run in time O(h+s) where h is the height of T and s is the number of elements output.

Problem 3. [15 points] Give an O($n$log$k$)-time algorithm to merge k nonempty sorted lists into one sorted list, where n is the total number of elements in all the input lists.
(*Hint:* One solution uses a min-heap.)

Problem 4. [15 points] We are given n lists $L_1, L_2, ..., L_n$ of integers in the range 1 to n. The sum of the sizes of the lists is m. Give an O(n+m)-time algorithm to sort all the lists; i.e., the algorithm should compute lists $L_1', L_2', ..., L_n'$, where L_i' contains in sorted order the same elements as L_i for each i=1,…,n.
Note: There are no restrictions on n, m or on the numbers in the lists. In particular, the same integer may appear in different lists, or may appear multiple times on the same list. Some lists may be empty or short, some may be long.

Problem 5. [20 points] In the *Bin Packing* problem, we want to pack a set of n items with sizes $s_1, s_2, \ldots, s_n$ into bins of capacity B; all sizes satisfy $0< s_i <B$. In other words we want to assign each item to a bin, so that the sum of the sizes of the items assigned to each bin is at most B. The objective is to use as few bins as possible. There are several heuristic algorithms for the Bin Packing problem, which compute reasonable solutions, although not always the optimal solution. One of them is the *Best Fit* heuristic, which works as follows:
Process the items in order 1,…,n. For each item i, put i in a bin that still has enough room for the item and is as full as possible (in case of a tie, pick any such bin); if none of the bins used so far has enough room for item i, then start a new bin and put i in it.
For example, if we have n=6 items with sizes s_1=4, s_2=6.3, s_3=2.5, s_4=3, s_5=1, s_6=4, and the bin capacity is B=10, then Best Fit will put item 1 in bin 1, item 2 in bin 2, item 3 in bin 2, item 4 in bin 1, item 5 in bin 2, and item 6 in bin 3.
Give an algorithm that implements the Best Fit heuristic and runs in time O($n$log$k$), where k is the number of bins used. The algorithm should compute the mapping from items to bins according to the Best Fit heuristic.

Problem 6. [20 points] We wish to maintain a collection S of pairs (birthdate, salary) for the employees of a company. Note that different employees may have the same birthdate and/or salary, so S may contain duplicate pairs. Propose a data structure that supports efficient Insertion and Deletion of pairs and in addition allows us to answer efficiently the following queries:
1. *MinBday*: What is the minimum birthdate?
2. *Count*(d): How many employees (tuples) have birthdate $\leq d$?
3. *AvgSal*(d): What is the average salary of employees with birthdate $\leq d$?
Describe each operation and analyze its time complexity. Make your data structure as efficient as you can. You can assume that comparison of dates and salaries as well as arithmetic operations take constant time.