הבינתחומי הרצליה
IDC HERZLIYA

Data Structures & Algorithms – Problem set 2

Due: 21.11.2021

Honor code:

a. Do not copy the answers from any source.
b. You may work in small groups but take no written notes, and write your solution on your own (mention collaborators in the submission).

Submission guidelines:

- Submit your solution via Moodle as a PDF file only. Other formats will not be graded.
- Typed submissions will get a bonus of 5 points.
- If you choose not to type your solution, make sure the scan is easy to read. We will deduct points for hard-to-read submissions.

Question 1 (15 points):

A d-ary heap is like a binary heap, but non-leaf nodes have d children instead of 2 children.

a. How would you represent a d-ary heap in an array A[0….(n-1)]?
b. What is the height of a d-ary heap of n elements in terms of n and d?
c. Give an efficient implementation of EXTRACT-MAX in a d-ary max-heap. Analyze its running time in terms of d and n.
d. Give an efficient implementation of INSERT in a d-ary max-heap. Analyze its running time in terms of d and n.

Question 2 (15 points):

Given an unsorted array, A, of n positive integer numbers.

a. Given an input number x. Suggest an efficient algorithm for checking if A contains x and x^2, and if so where (indexes). What is the worst-case time complexity of your algorithm?
b. Suggest an efficient algorithm for checking the amount of different elements in the array. What is the best and the worst-case time complexity of your algorithm?
c. Suggest an efficient algorithm for checking the maximal number of identical numbers in the array. What is the best and the worst case time complexity of your algorithm?
d. Suggest an algorithm for checking if the array contains two numbers, x, y such that $x^2 = y$. The time complexity should be O(n·logn) for the worst case.

Question 3 (30 points):

A $m \times n$ *Young tableau* is an $m \times n$ matrix such that the entries of each row are in sorted order from left to right and the entries of each column are in sorted order from top to bottom.
Some of the entries of a Young tableau may be ∞, which we treat as nonexistent elements.
Thus, a Young tableau can be used to hold $r \leq mn$ finite numbers.

a. Draw a 4×4 Young tableau containing the elements {9, 16, 3, 2, 4, 8, 5, 14, 12}.

b. Argue that a $m \times n$ young tableau Y is empty if $Y[1,1] = \infty$ and that Y is full if $Y[m,n] < \infty$.

c. Give an algorithm to implement EXTRACT-MIN on a nonempty $m \times n$ Young tableau that runs in O(m+n) time. In this algorithm we extract the minimum $Y[1,1]$ while keeping the table as a Young tableau.
Your algorithm should use a recursive subroutine that solves an $m \times n$ problem by recursively solving either an $(m-1) \times n$ or an $m \times (n-1)$ subproblem. (Hint: Think about percolation in binary heaps.)
Prove the correctness and running time of your algorithm

d. Show how to insert a new element into a nonfull $m \times n$ Young tableau in O(m+n) time. You do not have to formally prove correctness or formally analyze the running time.

e. Using no other sorting method as a subroutine, show how to use an $n \times n$ Young tableau to sort n^2 numbers in $O(n^3)$ time.

Question 4 (20 points):

a. As a computer store manager, you currently hold in your stock n different models of computers. To organize your shop, you wish to sort the different models according to their memory capacity. If two models have the same memory capacity, decide for yourself, which one comes first. A memory of a computer is composed of bits and is always a number of the form 2^k, for some natural number k. Assume that the maximal capacity is at most 2^n. Suggest an efficient algorithm for sorting all the models and analyze its runtime in the worst case.

b. Let A be an array of n numbers. Assume that at least $(n - 3\frac{n}{\log(n^2)})$ of the array's entries are integers between 1 and n. The remaining entries are non-integers, on which no prior knowledge exists. Is there an algorithm, which sorts A with time complexity $O(n)$ in the worst case? If your answer is yes, then describe the algorithm and give a short explanation of its runtime. If your answer is no, then explain why this cannot be done.

Question 5 (10 points):

Consider lucky a run of Quick Sort algorithm, where the pivot always split the n elements into n/5 and 4n/5. Write the recursive complexity function of this lucky run of the algorithm, and compute its worst case time complexity.

Question 6 (10 points):

Consider a modification of the radix-sort algorithm, in which we use Quick-Sort to sort each of the digits. Is this algorithm a valid sorting algorithm? If the answer is no, explain why. If the answer is yes, explain what is its **best-time complexity** of the modification? Your answer should depend on d, The number of digits needed to represent each number in the array.