CMPSC 465 Data Structures & Algorithms
Fall 2021 Antonio Blanca and David Koslicki

HW 3

Due September 20, 10:00 pm

Instructions: You are encouraged to solve the problem sets on your own, or in groups of three to five people, but you must write your solutions strictly by yourself. You must explicitly acknowledge in your write-up all your collaborators, as well as any books, papers, web pages, etc. you got ideas from.

Formatting: Each part of each problem should begin on a new page. Each page should be clearly labeled with the problem number and the problem part. The pages of your homework submissions must be in order. When submitting in Gradescope, make sure that you assign pages to problems from the rubric. You risk receiving no credit for it if you do not adhere to these guidelines.

Late homework will not be accepted. Please, do not ask for extensions since we will provide solutions shortly after the due date. Remember that we will drop your lowest two scores.

This homework is due Monday, September 20, at 10:00 pm electronically. You need to submit it via Gradescope (Class code 6PERPR). Please ask on Canvas about any details concerning Gradescope.

Extra credit. Five extra credits will be added to your homework score this time if you use Latex to typeset your solutions.

1. (20 pts.) Heap and Heapsort

(a) Run Build-Heap on the array $[10,9,4,7,6,2,3,1,8,5]$ to construct a min heap. Write down the resulting array. Then, run the loop in Heapsort for three iterations. For each iteration write down the array after Heapify-Down is finished.

(b) Show that the leaves of a heap occupy the positions $\lfloor \frac{n}{2} \rfloor, \ldots, n-1$ of the array.

(c) In the procedure Build-Heap, why do we decrease i from $\lfloor \frac{n}{2} \rfloor - 1$ to 0 instead of increasing from 0 to $\lfloor \frac{n}{2} \rfloor - 1$?

2. (15 pts.) Sorting Lower Bound

Show that any array of integers $x[1 \ldots n]$ can be sorted in $O(n+M)$ time, where

$$M = \max_i x_i - \min_i x_i$$

For small M, this is linear time: why doesn't the $\Omega(n \log n)$ lower bound apply in this case?

3. (15 pts.) Binary search

(a) Suppose we want to check if a sorted sequence A contains an element v. For this, we can use Binary Search. Binary Search compares the value at the midpoint of the sequence A with v and eliminates half of the sequence from further consideration. The Binary Search algorithm repeats this procedure, halving the size of the remaining portion of the sequence each time. Write a recurrence for the running time of Binary search and solve this recurrence.

(b) Ternary Search is a generalization of Binary Search that can be used to find an element in an array. It divides the array with n elements into three parts and determines, with two comparisons, which part may contain the value we are searching for. For instance, initially, the array is divided into three thirds by taking $mid1 = \lfloor \frac{n-1}{3} \rfloor$ and $mid2 = \lfloor \frac{2(n-1)}{3} \rfloor$. Write a recurrence for the running time of Ternary search and solve this recurrence.

4. (20 pts.) Counting number of inversions.

We are given a sequence of n distinct numbers $a_1, \ldots, a_n$. We define an inversion to be a pair $i < j$ such that $a_i > a_j$. Give an $O(n \log n)$ algorithm to count the number of inversions. (*Hint: assume that your algorithm returns a sorted array of input sequences, as well as the number of inversions. Divide the input sequences into two halves, then count inversions and sort each half by doing recursive calls, and find a way to merge these two sorted halves and to count the number of inversion between them.*)

5. (30 pts.) Recurrence relations.

Solve the following recurrence relations and give a O bound for each of them.

(a) $T(n) = 9T(n/3) + 12n^2$

(b) $T(n) = 2T(n/5) + \sqrt{n}$

(c) $T(n) = T(n-1) + c^n$, where $c > 1$ is a constant

(d) $T(n) = T(n-1) + n^c$, where $c \geq 1$ is a constant

(e) $T(n) = 5T(n/4) + n$

(f) $T(n) = T(n/2) + 1.5^n$

(g) $T(n) = 49T(n/25) + n^{3/2} \log n$

(h) $T(n) = T(\frac{n}{2}) + T(\frac{n}{4}) + T(\frac{n}{8}) + n$

Hint: Use induction. To elaborate, try to find a function g such that for all $k < n$ there is a constant $c_1 > 0$ such that $T(k) \leq c_1 g(k)$; prove that the inequality holds for every k by induction.

(i) $T(n) = T(3n/5) + T(2n/5) + \Theta(n)$. (Hint: same as part h)

(j) $T(n) = \sqrt{n}T(\sqrt{n}) + 10n$. (Hint: unfold the recursive relation by $\log \log n$ steps.)