Homework 3

CS 4104 (Summer 2024)

Assigned on July 12, 2024.
Submit a PDF file containing your solutions on Canvas by 11:59pm on Friday, July 19, 2024.

Instructions:

- The Honor Code applies to this homework with the following exception:
  - You are allowed to discuss possible algorithms and bounce ideas with your classmates. Do not discuss proofs of correctness, final answer or running time in detail with your classmate. You must write down your solution individually and independently. Do not send a written solution to your classmates for any reason whatsoever.
  - You are not allowed to consult sources other than your textbook, the slides on the course web page, your own class notes, the TA, and the instructor. **In particular, do not use a search engine or large language models such as ChatGPT**.
- Do not forget to typeset your solutions. Every mathematical expression must be typeset as a mathematical expression, e.g., the square of n must appear as n^2 and not as `n2`. You can use the LaTeX version of the homework problems to start entering your solutions. At the end of each problem are three commented lines that look like the example below. You can uncomment these lines and type in your solution within the curly braces.

```
% \solution{
%
% }
```

- Do not make any assumptions not stated in the problem. If you do make any assumptions, state them clearly, and explain why the assumption does not decrease the generality of your solution
- You must also provide a clear proof that your solution is correct (or a counter-example, where applicable). Type out all the statements you need to complete your proof. You must convince us that you can write out the complete proof. You will lose points if you work out some details of the proof in your head but do not type them out in your solution.
- If you are proposing an algorithm as the solution to a problem, keep the following in mind:
  - Describe your algorithms as clearly as possible. The style used in the text book is fine, as long as your description is not ambiguous. Explain your algorithm in words. A step-wise description is fine. *However, if you submit detailed pseudo-code without an explanation, we will not grade your solutions.*Do not describe your algorithms only for a specific example you may have worked out.
  - Make sure to state and prove the running time of your algorithm. You will only get partial credit if your analysis is not tight, i.e., if the bound you prove for your algorithm is not the best upper bound possible.
  - You will get partial credit if your algorithm is not the most efficient one that is possible to develop for the problem.
- In general for a graph problem, you may assume that the graph is stored in an adjacency list and that the input size is m + n, where n is the number of nodes and m is the number of edges in the graph. Therefore, a linear time graph algorithm will run in $O(m + n)$time.

Acknowledgement: Some of the questions in this Homework are inspired by/adopted from previous CS 4104 offerings at VT including by T. M. Murali and others.

Problem 1 (7 Points) Use substitution method to guess and prove that the running time of $T(n) = T(\frac{n}{4}) + \Theta(\log n)$.

Problem 2 (8 Points) Use the Master Method to establish the asymptotic bound for the following recurrences, if the Master Method applies:

1. $T(n) = T\left(\frac{8n}{11}\right) + n$
2. $T(n) = 7T\left(\frac{n}{2}\right) + n^2 \log n$
3. $T(n) = 2T\left(\frac{n}{4}\right) + \sqrt{n}$
4. $T(n) = T(n-2) + n^2$

Problem 3 (30 Points) Given a sorted array and a value x, the floor of x is the largest element in the array smaller than or equal to x. Write efficient functions to find the floor of x using recursion. **Example 1**

- **Input:**
  - Array: [1, 2, 4, 6, 10, 12, 14]
  - $x = 5$
- **Output**:
  - Floor of 5 : 4

Example 2

- **Input:**
  - Array: [2, 5, 8, 10, 11, 15, 18]
  - $x = 13$
- **Output**:
  - Floor of 13 : 11

Problem 4 (30 Points) Given a sorted array in which all elements appear twice (one after the other) and one element appears only once. Write a recursive algorithm to find that element in $O(\log n)$ complexity. **Example 1**

- **Input:** [1, 1, 2, 2, 3, 4, 4, 5, 5]
- **Output**: 3

Example 2

- **Input:** [0, 0, 1, 1, 2, 2, 3, 4, 4]
- **Output**: 3

Problem 5 [1] (25 Points) You are interested in analyzing some hard-to-obtain data from two separate databases. Each database contains n numerical values—so there are $2n$ values total—and you may assume that no two values are the same. You'd like to determine the median of this set of $2n$ values, which we will define here to be the n^{th} smallest value. However, the only way you can access these values is through queries to the databases. In a single query, you can specify a value k to one of the two databases, and the chosen database will return the k^{th} smallest value that it contains. Since queries are expensive, you would like to compute the median using as few queries as possible. Give an algorithm that finds the median value using at most $O(\log n)$ queries.

Example 1

- **Input:**

[1] From "Algorithm Design" by Jon Kleinberg and Eva Tardos

  - Database 1: [1, 4, 7, 10, 13]
  - Database 2: [2, 3, 5, 6, 8]
- **Output**:
  - Median: 5
- **Explanation**:
  - Combined Sorted Array: [1, 2, 3, 4, 5, 6, 7, 8, 10, 13]
  - Median Position:
    - Since there are 10 elements in total, the median is the 5

Example 2

- **Input:**
  - Database 1: [11, 13, 15, 17, 19]
  - Database 2: [2, 4, 6, 8, 10]
- **Output**:
  - Median: 11
- **Explanation**:
  - Combined Sorted Array: [2,4,6,8,10,11,13,15,17,19]
  - Median Position:
    - Since there are 10 elements in total, the median is 11

Problem 6 (20 Points) In a distant kingdom, a dragon guards a hoard of treasure. To reach the treasure, a knight must solve a puzzle where they need to find the k^{th} smallest element in a list of n unsorted integers. The knight can use a divide and conquer approach to solve the problem efficiently. Write a recursive algorithm to find the k^{th} smallest element in an unsorted array. **Example 1**

- **Input:**
  - List: [7, 10, 4, 3, 20, 15]
  - $K = 3$
- **Output**:
  - 3^{rd} smallest element: 7
- **Explanation**:
  - The $3rd$ smallest element in the sorted array $[3, 4, 7, 10, 15, 20]$ is 7.

Example 2

- **Input:**
  - List: [12, 3, 5, 7, 19]
  - $K = 2$
- **Output**:
  - 2^{nd} smallest element: 5
- **Explanation**:
  - The $2nd$ smallest element in the sorted array $[3, 5, 7, 12, 19]$ is 5.