CS 201: Algorithms, Mid Semester Examination, 19 September 2023

Marks: 50

- All the questions are compulsory.
- You can use propositions that were proved in class, *i.e.*, without re-proving them.
- Errors in questions are unavoidable; students are expected to distribute time across all the questions. If something is missing/incorrect in a problem description, clearly mention the assumption with the solution.
- Be precise. There is no credit for being unnecessarily verbose, *i.e.*, filling pages with no substance.
- Do not panic.

1. Suppose to solve a problem P, you are choosing among the following three algorithms proposed by –

   - **Friend who is coming to the class regular:** Algorithm solves the problem P of size N by dividing it into five sub-problems of half the size, recursively solving each sub-problem, and then combining the solutions in linear time.
   - **Friend who is not coming to the class regular:** Algorithm solves the problem P of size N by recursively solving two sub problems of size $N-1$ and then combining the solutions in constant time.
   - **ChatGpt:** Algorithm solves the problem of size N by dividing it into nine sub problems of size $N/3$, recursively solving each sub problem, and then combining the solutions in $\mathcal{O}(N^2)$ time.

   Who is giving you the best algorithm in term of time complexity? Justify the reason. [7]

2. In merge Sort, we need to merge two sorted arrays. Let the time complexity to merge two sorted arrays of size $N/2$ is $\Theta(N \log N)$. Obtain the time complexity of this kind of merge sort. [3]

3. Consider there are N distinct positive integers which are stored in an array A. We want to sort this array in ascending order using Quick sort. However, we cannot spend more than $\mathcal{O}(N \log N)$ time. What changes you will make into the Quick Sort so that the worst case time complexity remains $\mathcal{O}(N \log N)$. Also analyze the time complexity of your algorithm, *i.e.*, how you have obtained $\mathcal{O}(N \log N)$ time. [3+2]

4. To find k^{th} smallest element in an array A of N elements, suppose we modify the SELECT algorithm, *i.e.*, MEDIAN OF MEDIAN algorithm, by breaking the elements into group of size M, where M is an odd number. Analyze the time complexity of this algorithm where the group size is M. [7]

5. Suppose we are given an array A of N distinct elements, and we want to find $N/2$ elements in the array whose median is also the median of A. State **TRUE** or **FALSE** for the following statement and justify your answer "Any algorithm that does this must take at-least $\Omega(N \log N)$ time." [5]

6. You are given a sorted array of n elements which has been circularly shifted. For example, $35, 42, 5, 12, 23, 26$ is a sorted array that has been circularly shifted by 2 positions. Give an $\mathcal{O}(\log n)$ time algorithm to find the largest element in a circularly shifted array. (The number of positions through which it has been shifted is unknown to you.) [5]

7. Suppose that in a $0-1$ knapsack problem, the order of the items when sorted by increasing weight is the same as their order when sorted by decreasing value. Give an $\mathcal{O}(N)$ time algorithm to find an optimal solution to this variant of the knapsack problem where there are N items. Justify the correctness of your algorithm. [4+4]

8. Let A be a matrix of integers with N rows and M columns. We use $A(i, j)$ to denote integer in row i and column j, where $1 \leq i \leq N$ and $1 \leq j \leq M$. A path in the matrix is a sequence of integers from the matrix $A(i_1, j_1), A(i_2, j_2), \ldots, A(i_k, j_k)$, such that each next integer is to the **right** of the previous integer or **below** the previous integer in the matrix. For example, in the matrix A shown below, $A(1, 1), A(2, 1), A(2, 2)$ (which is $7, 1, 2$) is a path, and $A(2, 4), A(2, 5), A(2, 6), A(3, 6)$ (which is $12, 5, 3, 2$) is a path.

7	4	25	16	9	4	11
1	2	1	12	5	3	-4
10	7	5	-3	8	2	8
4	8	7	10	4	6	17

A path is called *decreasing* if each next integer on the path is strictly less than the previous integer. For example, path $A(2, 4), A(2, 5), A(2, 6), A(3, 6)$ (which is $12, 5, 3, 2$) is decreasing, while path $A(1, 1), A(2, 1), A(2, 2)$ (which is $7, 1, 2$) is not. The length of a path is the number of integers in the path. For instance, path $12, 5, 3, 2$ has length 4. A path consisting of just one number has length 1. The problem is to find the length of the longest decreasing path in a given matrix of integers and a sequence of integers in the longest path. In the above example, the length of the longest path is 6. There are several paths with length 6; one of them is $25, 16, 12, 5, 3, 2$.

1. Design a dynamic programming solution to this problem.
2. Explain how to calculate using dynamic programming (Basically, explain how to fill a table).
3. Describe the time complexity of your dynamic programming algorithm.

[4+3+3]