1. (10 pts) Suppose a server has $n$ customers waiting to be served. The service time required by each customer is known in advance: $t_i$ minutes for customer $i$. So, for instance, if the customers are serviced in the order of increasing index $i$, then the $i$th customer has to wait $\sum_{j=1}^{i} t_j$ minutes. (Note that a customer’s waiting time includes the time he spends getting served.) Our goal is to find an ordering that minimizes the total waiting time:

$$T = \sum_{i=1}^{n} (\text{time spent waiting by customer } i)$$

Give an efficient algorithm for computing the optimal order in which to service the customers. You must prove that the algorithm always finds optimal order, and also analyze its worst-case running time.

2. (10 pts) The linear-time Selection algorithm described in class divides the input elements into groups of 5, and then recursively solves the problem.

   - Re-analyze the algorithm when we divide the elements into groups of 3. Specifically, determine the sizes of the subproblems on which recursive calls will be made. Write down the final recurrence, and solve it. Does the algorithm still run in $O(n)$ time?

   - Re-analyze the algorithm if the elements are divided into groups of 7. Again, determine the sizes of the recursive subproblems, derive the final recurrence, and solve it. Does the algorithm still run in $O(n)$ time?

3. (10 pts) Suppose $A$ and $B$ are two sorted arrays, each with $n$ numbers. Design an $O(\log n)$ time algorithm to find the median of the set $A \cup B$. (You can assume that there are no duplicates in the set $A \cup B$.)

4. (10 pts) Let $A_1, \ldots, A_6$ be six matrices, where $A_1$ is $10 \times 20$, $A_2$ is $20 \times 5$, $A_3$ is $5 \times 40$, $A_4$ is $40 \times 2$, $A_5$ is $2 \times 10$, and $A_6$ is $10 \times 10$. Determine the optimal way to compute their product $A_1 A_2 A_3 A_4 A_5 A_6$. Show the dynamic programming table, with all the subproblem solutions.