CSE 551 Foundations of Algorithms*
Mid Term, Fall 2017
Closed Books, Closed Notes
Time: 1 hour
Answer any three questions
Each question carries 25 pts.

Problem 1: Let S be an ordered sequence of n distinct integers. Develop an algorithm to find a *longest increasing subsequence* of the entries of S. The subsequence is not required to be contiguous in the original sequence. For example, if $S = \{11, 17, 5, 8, 6, 4, 7, 12, 3\}$, a longest increasing subsequence of S is $\{5, 6, 7, 12\}$. Analyze your algorithm to establish its correctness and also its worst-case running time and space requirement.

Problem 2: Suppose that you are choosing between the following three algorithms:

(i) Algorithm A solves the problem by dividing it into six subproblems of half the size, recursively solving each subproblem, and then combining the solutions in linear time.

(ii) Algorithm B solves the problem of size n by recurisively solving three subproblems of size $n-1$ and then combining the solutions in constant time.

(iii) Algorithm C solves the problem of size n by dividing them into ten subproblems of size $n/3$, recurisively solving each subproblem, and then combining the solutions in $O(n^2)$ time.

What are the running times of each of the algorithms (in big-O notation), and which would you choose?

Problem 3: Suppose that the dimensions of the matrices A, B, C and D are $20\times12, 12\times5, 5\times60$ and 60×4, respectively. Find the fewest number of multiplications that will be necessary to compute the final matrix and the order in which the matrix chain be multiplied so that it will require the fewest number of multiplications. **Show all your work**.

Problem 4: Let $A_1, \ldots, A_n$ be matrices where the dimensions of the matrices $A_i, 1 \leq i \leq n$ are $d_{i-1} \times d_i$. The following algorithm is proposed to determine the best order (i.e., the order that requires the fewest number of multiplications) in which to perform the matrix multiplications to compute $A_1 \times A_2 \times \cdots \times A_n$:

At each step, choose the largest remaining dimension (from among $d_1, \ldots, d_{n-1}$), and multiply two adjacent matroces that share that dimension. What is the complexity of this algorithm? Is this algorithm always going to find the order in which to multiply the matrix chain that will require in the fewest number of multiplications? If your answer is "yes", then provide arguments to support the assertion that this algorithm will always find the best order. If your anser is "no", then provide an example where the algorithm fails to find the best order.