CSCI4110, SP2017, Final Test

All questions have same number of points. Solve any 6 problems. Please turn in 6 problems only. If you turn in more than 6 problems, then the first 6 will be graded. You will be graded for accuracy, and clarity.

(1)- Solve $T(n) = 2T(n) + n, n = 2^k, T(1) = 1$.

(2)-Solve $T(n) = 4T(n/3) + n^2, n = 2^k, T(1) = 1$.

(3) Consider the matrix chain problem. We are given matrices, $A_1, A_2, \ldots, A_n$, where $A_i, i = 1, 2, \ldots, n$ has $p_{i-1}$ rows and $p_i$ columns. We want to compute $A_1 \times A_2 \times \ldots \times A_n$ by utilizing minimum number of scalar multiplications. Design a dynamic programming algorithm for matrix chain problem, and derive its time complexity. You should define all proper terms first, and then derive a recurrence relation, and then solve it.

(4) Let $G = (V, E)$ be a directed graph with weight $w(i, j)$ assigned to any edge $ij \in E$. Write the pseudo-code for Floyd-Marshall algorithm. What is the time complexity of this algorithm?

(5) Let $X$ and $Y$ be sequences of lengths $n$ and $m$, respectively. Design a dynamic programming algorithm for computing the longest common subsequence of $X$ and $Y$, and derive its time complexity. You should define all proper terms first, and then derive a recurrence relation, and then derive the algorithm.

(6) Assume that we can multiply two 2 by 2 matrices using 6 scalar multiplications. Write a recurrence relation for a divide and conquer algorithm that uses this fact, to multiply two $n$ by $n$ matrices, and solve this recurrence relation, to derive the time complexity of the algorithm.

(7) Consider the travelling salesman problem, with triangular inequalities. Describe a 2 times the optimal approximation algorithm for solving this problem and justify its correctness. What is the time complexity of this algorithm?

(8) Consider the problem for finding the minimum $n$ numbers. Prove that any algorithm that solves this problem by only comparing, has to perform at least $n - 1$ comparisons in the worst case.

(9) Answer True or False.

1. Time complexity of the standard greedy algorithm for solving the activity selection is $O(n \log \log(n))$.

2. Time complexity of Strassen’s algorithm is $O(n^3)$ ($n$ is the number of rows in the matrix.)

3. Time complexity of Dijkstra’s algorithm (using a priority queue) is $O(m \log n)$, where $n$ and $m$ are the number of vertices and edges.

4. Time complexity of Heap Sort is $O(n \log \log(n))$, where $n$ is the input size.

5. One can solve $0 - 1$ Knapsack problem in $O(2^n)$ time where $n$ is the number of objects.