Fall 2018, CMPSC 465: Final Exam (practice) solutions.

1. A contiguous subsequence of a list S is a subsequence made up of consecutive elements of S. Give a linear-time dynamic programming algorithm for the task of determining a contiguous subsequence of maximum sum. Assume that a subsequence of length zero has sum zero.
(**10 points**)
Solution:
Let $C(i)$ denote the largest sum of a contiguous subsequence *ending* exactly at position i. i can be 0.

   We have the following recursive relationship: $C(0) = 0$, $C(i) = \max(0, C(i-1) + a_i)$

   The largest sum is given by the maximum element in the array C. Let's say the max element occurs at location k. This means the contiguous subsequence of maximum sum will terminate at k. Its beginning will be at the first index $j \leq k$ such that $C(j-1) = 0$, as this implies that extending the sequence before j will only decrease its sum.

   Correctness: The contiguous subsequence of largest sum ending at i will either be empty or contain a_i. In the first case, the value of the sum will be 0. In the second case, it will be the sum of a_i and the best sum we can get ending at $i-1$, i.e., $C(i-1)+a_i$. Because we are looking for the largest sum, $C(i)$ will be the maximum of these two possibilities.

   Running time: The running time for this algorithm is $O(n)$, as we have n subproblems and the solution of each can be computed in constant time.

2. Show the longest common subsequence (LCS) table for the following two strings:
X: EXAMINE and Y: ELIMINATE.
What is the length of the LCS?
(**7 points**)
Solution:
We have the following table:

			E	X	A	M	I	N	E
	L	-1	0	1	2	3	4	5	6
	-1	0	0	0	0	0	0	0	0
E	0	0	**1**	1	1	1	1	1	1
L	1	0	1	1	1	1	1	1	1
I	2	0	1	1	1	1	**2**	2	2
M	3	0	1	1	1	2	2	2	2
I	4	0	1	1	1	2	**3**	3	3
N	5	0	1	1	1	2	3	**4**	4
A	6	0	1	1	2	2	3	4	4
T	7	0	1	1	2	2	3	4	4
E	8	0	1	1	2	2	3	4	**5**

The length of the LCS is 5.

3. Suppose you are given an instance of the fractional knapsack problem in which all items have the same weight. Show how you can solve the problem in this case in *expected* $O(n)$ time.
(**7 points**)
Solution:
Let the weight of an item be w. The greedy solution to the fractional knapsack problem is to pick the largest items by value (which is defined as the ratio of benefit to weight), until the knapsack is full. Since all items have the same weight, the algorithm simplifies to picking items in decreasing order of their benefit. Specifically, we pick the top $\lceil \frac{W}{w} \rceil$ items ranked by benefit. Sorting the benefit array takes $\Theta(n \log n)$ time using a comparison-based sorting algorithm such as MergeSort, but we desire a faster algorithm.

Given the *unsorted* benefit array, we want to find the kth largest value ($k = \lceil \frac{W}{w} \rceil$) in this array. Let's call this value $b_{\min}$. We can do this using the QuickSelect algorithm (see Section 2.4 of the textbook; this is the only randomized algorithm we have studied so far), which takes *expected linear* time. After finding this element, we need to make another pass through the benefit array, and if an item's benefit is greater than $b_{\min}$, we pick this item to be in the solution. This takes linear time. So the overall running time is expected $O(n)$.

4. Prove that the following statement is correct or give a counter-example if it is incorrect: If e is part of some MST of G, then it must be a lightest edge across some cut of G. Assume that the graph G is undirected and connected.
(**7 points**)
Solution:
The statement is true. Let the edge be $e = \langle u, v$. Consider a cut with u in S and v in $V \setminus S$. Since e belongs to an MST, it must be the lightest edge across this cut according to the cut property.

5. Using appropriate notation, define the traveling salesman problem. Explain the distinction between the Minimum Spanning Tree (MST) problem and the traveling salesman problem.
(**7 points**)
Solution:
Consider a weighted undirected graph G with positive integer edge weights. The goal of the traveling salesman problem is to find the minimum weight simple cycle in this graph visiting each vertex exactly once. The output of the minimum spanning tree problem is a tree of minimum cost given a weighted connected graph. There are several efficient polynomial-time algorithms known for the MST problem, but the traveling salesman problem is NP-Complete.

6. Show that, if c is a positive real number, then $g(n) = 1 + c + c^2 + \cdots + c^n$ is $\Theta(1)$ if $c < 1$, $\Theta(n)$ if $c = 1$, and $\Theta(c^n)$ if $c > 1$.
(**6 points**)
Solution:
$g(n) = \frac{1-c^{n+1}}{1-c}$ when $c \neq 1$ (applying geometric series formula). When $c < 1$, $c^{n+1} \to 0$ as $c \to \infty$. Thus, $g(n) \to \frac{1}{1-c} = \Theta(1)$. When $c > 1$, $g(n) = \frac{c^{n+1}-1}{c-1}$ and $c^{n+1} \gg 1$. Thus, $g(n) = O(c^n)$. When $c = 1$, $g(n) = n + 1 = \Theta(n)$.

7. Given a sorted array of distinct integers $A[1, \ldots, n]$, you want to find out whether there is an index i for which $A[i] = i$. Give a divide-and-conquer algorithm that runs in time $O(\log n)$.
(**10 points**)
Solution:
Exercise 2.17 of textbook. The solution has a reasoning similar to binary search.

8. Suppose that we have numbers between 1 and 1000 in a binary search tree, and we want to search for the number 363. Which of the following sequences could not be the sequence of nodes examined?

   (a) 2, 252, 401, 398, 330, 344, 397, 363.
   (b) 924, 220, 911, 244, 898, 258, 362, 363.
   (c) 925, 202, 911, 240, 912, 245, 363.

   (**7 points**)
   Solution:
   Draw the trees to notice that the first two satisfy the BST property. However, the third sequence violates the order (911 and 912).

9. Given a directed acyclic graph G, give a linear time algorithm to determine if G contains a directed path that touches every vertex exactly once.
(**7 points**)
Solution:
We are asked to determine if a DAG has a Hamilton/Rudrata path. Perform a topological sort of the vertices, and let it be $v_1, v_2, \ldots, v_n$. For every consecutive pair of vertices, check if there is an edge $\langle v_1, v_2 \rangle$ in the graph. If yes, then there is a Rudrata path through these edges. Otherwise, this means that some vertices are unreachable in a path from a source to a sink in the DAG.

10. You are given a strongly connected directed graph $G = (V, E)$ with positive edge weights, along with a particular node $v_0 \in V$. Give an efficient algorithm for finding shortest paths between all pairs of nodes, with the one restriction that these paths must all pass through v_0.
(**7 points**)
Solution:
Exercise 4.14 from textbook, discussed in class. Consider a shortest path between any two vertices s and t, passing through v_0. Because this is a shortest path, we must have that $d(s, v_0) + d(v_0, t) = d(s, t)$. Since the graph is strongly connected, a shortest path between s and t is guaranteed to include the shortest path from s to v_0, and the shortest path from v_0 to t. We can thus execute two single-source shortest path computations, one from v_0 in the original graph, and the other from v_0 in the reversed graph, and use this information to compute shortest path lengths between all pairs of vertices.