COMP 3711 – Design and Analysis of Algorithms
2022 Fall Semester – Written Assignment 3
Distributed: October 26, 2022
Due: November 9, 2022, 23:59

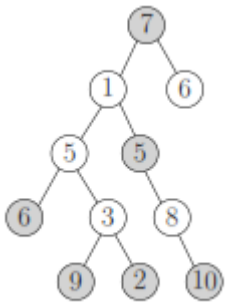
Your solution should contain
(i) your name, (ii) your student ID #, and (iii) your email address
at the top of its first page.

Some Notes:
• Please write clearly and briefly. In particular, your solutions should be written or printed on clean white paper with no watermarks, i.e., student society paper is not allowed.
• Please also follow the guidelines on doing your own work and avoiding plagiarism as described on the class home page. You must acknowledge individuals who assisted you, or sources where you found solutions. Failure to do so will be considered plagiarism.
• The term Documented Pseudocode means that your pseudocode must contain documentation, i.e., comments, inside the pseudocode, briefly explaining what each part does.
• Many questions ask you to explain things, e.g., what an algorithm is doing, why it is correct, etc. To receive full points, the explanation must also be understandable as well as correct.
• Please make a copy of your assignment before submitting it. If we can't find your submission, we will ask you to resubmit the copy.
• Submit a SOFTCOPY of your assignment to Canvas by the deadline. If your submission is a scan of a handwritten solution, make sure that it is of high enough resolution to be easily read. At least 300dpi and possiblydenser.

Question 1 (20%): Maximum sum in binary tree

Given a binary tree where each node has a positive number, describe a dynamic programming algorithm to select the subset of nodes with the maximum sum provided that adjacent nodes cannot be picked. In the following example, the shaded nodes form the optimal solution. For full credits, your algorithm should run in O(n) time, where n is the number of nodes. The algorithm just needs to output the sum, not the actual nodes picked.

Example:



Question 2 (20%): Longest increasing path in 2D Matrix
Consider a *n*x*n* matrix *M* of distinct integers. We wish to find the longest path in *M* such that: cell *M*[*i*,*j*] can be reached by moving right from *M*[*i*,*j*-1], or down from *M*[*i*-1,*j*], provided that *M*[*i*,*j*-1]< *M*[*i*,*j*] or *M*[*i*-1,*j*]] < *M*[*i*,*j*]. The path can start and end in any cell of *M*. Describe a dynamic programming algorithm that outputs the longest path, and discuss its time/space complexity.

Example: In the following matrix the longest paths are 1,3,5,6 and 2,4,7,8, both with length 4.

M	*j*=1	*j*=2	*j*=3
i=1	2	1	3
i=2	4	7	5
i=3	9	8	6

Question 3 (30%): Set partitioning
Let *S* be a set of *n* positive integers.

1] Describe a dynamic programming algorithm that decides whether it is possible to partition *S* into two subsets *S1* and *S2* that have equal sum. Assume that the sum of all elements in *S* is even. Describe the time and space complexity, considering that we only need the True/False answer, but not the sets *S1*, *S2*.

Example: if *S* = {3,5, 9, 1}, then the answer is True (*S1*={3,5,1} *S2*={9}). If *S* = {3,5, 11, 1}, the answer is False.

2] Describe a dynamic programming algorithm that partitions *S* into two sets *S1* and *S2* so that the absolute difference between their sums is minimum. Your algorithm should output *S1* and *S2*. Describe the time and space complexity.

Example: if *S* = {3,5, 11, 1}, the answer is *S1*={3,5,1}, *S2*={11}, or *S1*={11}, *S2*={3,5,1} and the absolute difference of sums is 2.

Question 4 (30%): Continuous subarray partitioning
Given an array *A* of *n* positive integers and an integer *m* (*m* is much smaller than *n*), we wish to split *A* into *m* **non-empty continuous** subarrays, so that the largest sum among these *m* subarrays is minimized. Provide the recurrence for the optimal solution, and describe a dynamic programming algorithm and its time/space complexity.

Example: Assume that the input array is *A*=[7,2,5,10,8] and *m* = 2. There are four ways to split *A* into two subarrays: **(1)** [7],[2,5,10,8], **(2)** [7, 2], [5,10,8], **(3)** [7, 2, 5],[10,8], **(4)** [7, 2, 5, 10],[8]

The best way is to split it into [7,2,5] and [10,8] because the largest sum between the two subarrays is only 18.