COMP 550.001 - Fall 2017
Assignment 2

Part A due: Wednesday, September 13, 2017 (start of class)
Part B due: Friday, September 15, 2017 (4:00 p.m.)

For Part A, you should submit a physical copy of your written homework at the start of class.
For Part B, you should submit a .tar.gz or .zip file with your solutions on Sakai.

Optional Part C due: Friday, September 15, 2017 (start of class)

This assignment includes an optional Part C. Earning half of the points will be worth half of a late day (only integral late days may be used to turn in homework late, but a partial late day can count as partial extra credit at the end of the semester), and earning at least 80% of the points will be worth a full late day. You should submit your code in a .zip or .tar.gz file to Sakai, and the analysis in a physical copy.

Part A: Due Wednesday, September 13, 2017

[5 points] Problem 1: CLRS Exercise 4.3-3

We saw that the solution of $T(n) = 2T(\lfloor n/2 \rfloor) + n$ is $O(n \lg n)$. Show that the solution of this recurrence is also $\Omega(n \lg n)$. Conclude that the solution is $\Theta(n \lg n)$.

[5 points] Problem 2: CLRS Exercise 4.4-5

Use a recursion tree to determine a good asymptotic upper bound on the recurrence $T(n) = T(n-1) + T(n/2) + n$. Use the substitution method to verify your answer.

[20 points] Problem 3: Subset of CLRS Problem 4-1

Give asymptotic upper and lower bounds for $T(n)$ in each of the following recurrences. Assume that $T(n)$ is constant for $n \leq 2$. Make your bounds as tight as possible, and justify your answers.

a. $T(n) = 2T(n/2) + n^4$.

b. $T(n) = 7T(n/2) + n^2$.

c. $T(n) = 2T(n/4) + \sqrt{n}$.

d. $T(n) = T(n-2) + n^2$.

[15 points] Problem 4: Subset of CLRS Problem 4-3

Give asymptotic upper and lower bounds for $T(n)$ in each of the following recurrences. Assume that $T(n)$ is constant for sufficiently small n. Make your bounds as tight as possible, and justify your answers.

a. $T(n) = 3T(n/3 - 2) + n/2$.

b. $T(n) = T(n/2) + T(n/4) + T(n/8) + n$.

c. $T(n) = T(n - 1) + 1/n$.

[15 points] Problem 5: CLRS Exercise 9.3-8

Let $X[1..n]$ and $Y[1..n]$ be two arrays, each containing n numbers already in sorted order. Give an $O(\lg n)$-time algorithm to find the median of all $2n$ elements in arrays X and Y.

Part B: Due Friday, September 15, 2017

[15 points] Problem 1

In this problem, you'll create a memoized version of FIBONACCI, and compare its results with the straightforward recursive solution for large values of the input.

1(a)

On your machine, what is the lowest value of n such that FIB takes more than 10 seconds to complete?

1(b)

First, fill in the implementation of MEMOIZEDFIB. You should use memoization to avoid re-calculating values twice, but still call MEMOIZEDFIB recursively.

For the value of n you gave in 1(a), how long does MEMOIZEDFIB take to complete?

1(c)

Now, write a bottom-up iterative Fibonacci number generator. The idea is this: each non-base-case computation of FIB(N) requires computing all smaller values. Rather than starting with n and recursively calling the function, instead iteratively build up a table of the results for $i = 1 \text{to} n$. Your function should run in $\Theta(n)$ time. Put this implementation in BOTTOMUPFIB.

For the value of n you gave in 1(a), how long does BOTTOMUPFIB take to complete?

[25 points] Problem 2: Variant of CLRS Exercise 15.1-4

In this problem, you will implement different approaches to the rod cutting problem discussed in lecture. A simple `RodCuttingSolution` class is provided to you. It has members `value` and `lengths`, which you should use to store the optimal value and the cut-rod lengths to achieve that value.

2(a)

First, implement the following pseudocode by filling in the function `cutRod`.

```
Cut-Rod(v,n)
if n == 0:
    return 0
bestVal = −∞
for i = 1 to n:
    bestVal = max(bestVal, v[i] + Cut-Rod(v, n − i))
return bestVal
```

2(b)

Next, implement the dynamic programming memoized version in `cutRodMemoized`.

```
Cut-Rod-Memoized(v,n)
let res[0..n] be a new array
for i = 0 to n:
    res[i] = −∞
return Cut-Rod-Memoized-Aux(v, n, res)
```

```
Cut-Rod-Memoized-Aux(v,n,res)
if res[n] ≥ 0:
    return res[n] // use memoized result
if n == 0:
    q = 0
else:
    q = −∞
    for i = 1 to n:
        q = max(q, v[i] + Cut-Rod-Memoized-Aux(v, n − i, res))
res[n] = q // memoize the value for later
return q
```

2(c)

Finally, modify your implementations of cutRod and cutRodMemoized to return both the optimal value, and the resulting lengths of rod to get that value. A simple `RodCuttingSolution` class is provided to you. It has members `value` and `lengths`, which you should use to store the optimal value and the cut-rod lengths to achieve that value.

[Optional] Part C: Due Friday, September 15, 2017

Problem 4-6 An $m \times n$ array A of real numbers is a ***Monge array*** if for all i, j, k, and l such that $1 \le i < k \le m$ and $1 \le j < l \le n$, we have

$$A[i, j] + A[k, l] \le A[i, l] + A[k, j].$$

In other words, whenever we pick two rows and two columns of a Monge array and consider the four elements at the intersections of the rows and the columns, the sum of the upper-left and lower-right elements is less than or equal to the sum of the lower-left and upper-right elements. For example, the following array is Monge:

10	17	13	28	23
17	22	16	29	23
24	28	22	34	24
11	13	6	17	7
45	44	32	37	23
36	33	19	21	6
75	66	51	53	34

a.

Prove that an array is Monge if and only if for all $i = 1, 2, ..., m-1$ and $j = 1, 2, ..., n-1$, we have

$$A[i, j] + A[i+1, j+1] \le A[i, j+1] + A[i+1, j].$$

(*Hint:* For the "if" part, use induction separately on rows and columns.)

b.

The following array is not Monge. Change one element in order to make it Monge. (*Hint:* Use part (a).)

37	23	22	32
21	6	7	10
53	34	30	31
32	13	9	6
43	21	15	8

c.

Let $f(i)$ be the index of the column containing the leftmost minimum element of row i. Prove that $f(1) \le f(2) \le ... \le f(m)$ for any $m \times n$ Monge array.

d.

Here is a description of a divide-and-conquer algorithm that computes the left-most minimum element in each row of an $m \times n$ Monge array A:

> Construct a submatrix A' of A consiting of the even-numbered rows of A. Recursively determine the leftmost minimum for each row of A'. Then compute the leftmost minimum in the odd-numbered rows of A.

Explain how to compute the leftmost minimum in the odd-numbered rows of A (given that the leftmost minimum of the even-numbered rows is known) in $O(m + n)$ time.

e.

Write the recurrence describing the running time of the algorithm described in part (d). Show that its solution is $O(m + n \log m)$.