Problem Set 2: Due Wednesday, March 7th 2018 by 10:10am
(within the first 5 minutes of class)

Homework Instructions
Much of what is learned in this course will be from trying to solve the homework problems, so make a conscientious effort to complete them well.

Your solutions should be concise and clear. Understandability of the solution is as necessary as correctness. Expect to lose points if you provide a “correct” solution with an unclear write-up. As with an English paper, do not expect to turn in a first draft: it takes refinement to describe something well. If you can’t solve a problem, briefly indicate what you’ve tried and where the difficulty lies. Don’t try to pull one over on us. :) You may (and are encouraged to) discuss the homework problems with your classmates. For each problem, you must acknowledge the people with whom you discussed your work, and you must independently write up your own solutions (you should not have the same writeup as another student). You must also acknowledge any other sources (online, text, etc.) that you use.

Your homework submission must:
• Be typed, stapled, and submitted as a hardcopy within the first 5 minutes of class on the due date.
• Include the names of any people with whom you discussed your work or any other sources (online, text, etc.) that you used.
• Include the honor code with your signature on the top of the first page (this acknowledges that you are not copying the solutions from online sources).

[20 Problem 1.] CLOSEST PAIR OF POINTS
For the Closest Pair of Points problem, recall the closestCrossing sub-routine that finds the minimum distance between two points such that one is in \( P_L \) and the other is in \( P_R \).

(a) [11 pts ] Describe, in words and with pseudocode, how to implement the closestCrossing sub-routine so that it finds this distance in \( O(n) \) time.

Hint: Think about what else can be done in the pre-processing step to make this sub-routine run faster.

(b) [4 pts ] Justify the running time of your algorithm.

(c) [5 pts ] Show how your closestCrossing sub-routine (and changes to the pre-processing step) would run on the following set of points with \( d = 10 \): 
(6, 18), (9, 20), (12, 5) (17, 3) (19, 16) (24, 24) (27, 6) (27, 15)

You do not have to find the distance between the closest pair of points - just show the steps your closestCrossing sub-routine would take to find the closest distance between pairs of crossing points.
Problem 2. Divide-and-Conquer

Given an array $A = \langle A[1], A[2], \ldots, A[n] \rangle$, we want to find the minimum and maximum values in $A$ and we do this by comparing elements of $A$. Assume that $n = 2^d$ for some $d \geq 1$.

(a) [5 pts] The “obvious” algorithm makes $2n - 2$ comparisons. Give pseudocode for the algorithm and explain the number of comparisons made.

(b) [7 pts] Can we do it better? Carefully specify (give pseudocode) a more efficient divide-and-conquer algorithm (for full credit, it should have the $T(n)$ stated in part (d)).

(c) [5 pts] Let $T(n)$ be the number of comparisons your algorithm makes. Write a recurrence relation for $T(n)$.

(d) [7 pts] Show that your recurrence relation has as its solution $T(n) = 3n/2 - 2$.

Problem 3. Recurrences

For our first example of the substitution method, we solved the recurrence:

$$T(n) = 2T(n/2) \text{ for } n > 1$$
$$T(n) = c \text{ for } n = 1$$

We correctly guessed $f(n) = n$ and showed $T(n) = O(n)$.

(a) [10 points] Show the steps of the substitution method with an incorrect guess of $T(n) = O(n^2)$. Does it work (i.e. does the proof follow through)? If so, why? If not, explain why not.

(b) [10 points] Now show the steps of the substitution method with an incorrect guess of $T(n) = O(1)$. Does it work? If so, why? If not, explain why not.

Problem 4. Recurrences

Here is a technique for computing $2^n$ for $n \geq 0$:

If $n = 0$ then $2^n = 1$, if $n = 1$ then $2^n = 2$, if $n > 1$ and $n$ is even then $2^n = 2^{n/2} \cdot 2^{n/2}$, and if $n > 1$ and $n$ is odd then $2^n = 2 \cdot 2^{(n-1)/2} \cdot 2^{(n-1)/2}$.

(a) [10 points] Use this technique to write a simple (i.e. don’t worry about runtime) recursive function that takes as input a value $n$ and returns $2^n$.

(b) [7 points] Write a recurrence for your recursive function from (a). For simplicity, your recurrence should have just two cases: the base case is when $n \leq 1$ and the recursive case is when $n > 1$ (so do not distinguish between even and odd $n$). Also for simplicity, indicate an addition of constant time with $+1$.

(c) [5 points] Solve your recurrence. Hint: notice that this part of the problem is worth only 5 points.

(d) [5 points] Draw a tree that illustrates the sequence of calls to the function for $n=8$. You may draw this by hand.

(e) [10 points] Modify your function from (a) so that it uses memoization.

(f) [7 points] What is the runtime of your function from (e)? Briefly explain.