Burton Rosenberg

Midterm

October 11, 2004. 1:25–2:15 PM

There are five problems each worth five points for a total of 25 points. Show all your work, partial credit will be awarded. Space is provided on the test for your work; if you use a blue book for additional workspace, sign it and return it with the test. No notes, no collaboration.

Name: _____

Problem	Credit
1	
2	
3	
4	
5	
Total	

1. Recall the *recursion tree:* when solving a recurrence you draw a tree in which each edge to a child is a recursive invocation of the function. You can solve the recurrence by summing the work associated with a node by each level, and calculating the number of levels.

   Draw a recursion tree and give a good Big-Oh estimate of work for an algorithm with the recurrence,

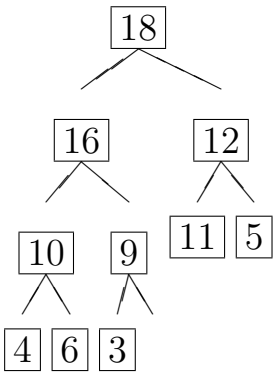
   $$T(n) = 3T(n/3) + C$$

   where C is a positive constant. *Hint:* The answer is *not* $n \log n$!

2. Indicate for each pair of expression A, B whether A is O, o, Ω, ω or Θ of B. Assume that $k \geq 1, \epsilon > 0$ and $c > 1$ are constants. Write "yes" or "no" in each box.

A	B	O	o	Ω	ω	Θ
$n \log n$	$n\sqrt{n}$					
$3n^3 + 40n^2$	$0.01n^3 - n^2 \log n$					
$\log n$	1					
2^n	n^{1000}					

3. Consider the following max-heap:



Show (in pictures) the process of delete max on this heap. Show two iterations of delete max, that is, remove the 18 and then the 16 from the heap.

4. You are given an array $A[1 \dots n]$ of n integers and an integer value v. We are looking for an algorithm which finds indices i, j such that $A[i] + A[j] = v$, if possible, or indicates that this is not possible, that is, no two integers in the array sum to v.

   (a) Consider the algorithm which simply tries all possible indices. What is the run time for this algorithm?

   (b) Can you find an $O(n \log n)$ algorithm solving this problem? Hint: sort the array A. Now set i to 1 and j to n, and either increments i or decrements j depending on the comparison of values $A[i] + A[j]$ to v. Argue the correctness and justify the runtime claim of your algorithm.

5. Suppose that you are given n red and n blue water jugs, all of different shapes and sizes. All red jugs hold different amounts of water, as do the blue ones. Moreover, for every red jug, there is a blue jug that holds the same aount of water, and vice versa.

   It is your task to find a grouping of the jugs into pairs of red and blue jugs that hold the same amount of water. To do so, you may perform the following operation: pick a pair of jugs in which one is red and one is blue, fill the red jug with water, and then pour the water into the blue jug. The operation will tell you whether the red or the blue jug can hold more water, or if they are the same volume. Assume that such a comparison takes one time unit.

   Give an $\Theta(n^2)$ algorithm which determines the grouping.

   For *extra credit*, your algorithm will be randomized and have expected run time $O(n \log n)$.