Assignment 2. CS341, Winter 2011
Distributed Thursday, Jan. 20, due Feb. 3, 2011. Hand in to the assignment boxes on the 3rd floor of MC.

1. (10 marks) Solve the following recurrences by the Master Theorem. Please show your work and express your answer as a tight asymptotic bound on $T(n)$.

   (a) Suppose you have a divide and conquer algorithm for multiplying large matrices. Suppose your algorithm divides each of the original n by n matrices into $n/3$ by $n/3$ matrices. Suppose that you have a way of multiplying and recombining the $n/3$ matrices into the n by n product that requires 8 matrix multiplications and 23 matrix additions. What is the time complexity of your new algorithm? Does your algorithm perform better than Strassen's algorithm? Write the recurrence of $T(n)$ for your algorithm and solve it using the Master Theorem.

   (b) $T(n) = 3T(n/2) + n^2$ for $n > 1$ and $T(n) = 1$ for $n \leq 1$.

   (c) $T(n) = 4T(n/2) + n^2$ for $n > 1$ and $T(n) = 1$ for $n \leq 1$.

   (d) $T(n) = 3T(n/3) + n \log n$ for $n > 1$ and $T(n) = 1$ for $n \leq 1$.

2. (5 marks) Given a set A of n elements. I.e. $A = \{a_1, \ldots, a_n\}$. If we wish to give a unique binary index (or name) to each of the elements of A, show that more than half of the names must have more than $\log_2 n - 2$ bits each.

3. (10 marks) Given a number i with binary representation $b_{k-1} \ldots b_1 b_0$, define $\text{rev}_k(i)$ to be the number that has binary representation $b_0 b_1 \ldots b_{k-1}$. (Example: $\text{rev}_3(3) = 6$ since 3 and 6 have binary representations "011" and "110" respectively.)

   Given $n = 2^k$, consider the problem of computing the sequence $\text{rev}_k(0), \text{rev}_k(1), \ldots, \text{rev}_k(n-1)$. (Example: for $n = 8$, the output is (0,4,2,6,1,5,3,7), as the binary representatoins of these numbers are 000, 100, 010, 110, 001, 101,011, 111.)

   Give an $O(n)$-time algorithm by using divide-and-conquer. (This is faster than the trivial algorithm which reverses each number one by one, in total time $O(nk) = O(n \log n)$.) Remember, arithmetic operations involving two integers are performed in $O(1)$ time.

4. (10 marks) We are given an index k and two sorted arrays A and B of sizes n_A and n_B, respectively. Describe an $O(\log n_A + \log n_B)$ time algorithm that can find the k-th smallest element among all $n_A + n_B$ elements in the two arrays.

   Example: If $A = (3, 9, 15)$ and $B = (2, 4, 8, 10, 11)$, and $k = 5$, the ouput would be 9. You may assume that all elements are distinct.

   Hint: Aim for a recurrence of the form $T(N) = T(N/2) + O(1)$, where $N = n_A n_B$. Think in terms of the product of the two arrays.

   More hint: Let m_A and m_B be the middle elements of A and B, respectively. If $k < (n_A + n_B)/2$, and $m_A < m_B$, can we prune half of one of the arrays? What about the other cases?

5. (25 marks, programming) (Computer screen display problem.) You are to design and implement a program, using divide-and-conquer and runnning in time $O(n \log n)$, to display objects on the screen. To make your life simple, all objects are rectangles and standing on the same horizontal line (say $y = 0$). the i-th object is specified as (L_i, H_i, R_i) where L_i and R_i are its leftmost and rightmost coordinates, respectively, and H_i is its height. The input is a sequence of triples of integers. The output should show the outline of the objects such that hidden parts of the objects are not shown. For example, if the input is (you can assume 3 numbers in a line, separated by spaces):

   (1,11,5),(2,6,7),(3,13,9),(12,7,16),(14,3,25),(19,18,22),(23,13,29),(24,4,28)

   the output should be:

   (1,11), (3,13), (9,0), (12,7), (16,3), (19,18), (22,3), (23,13), (29,0)

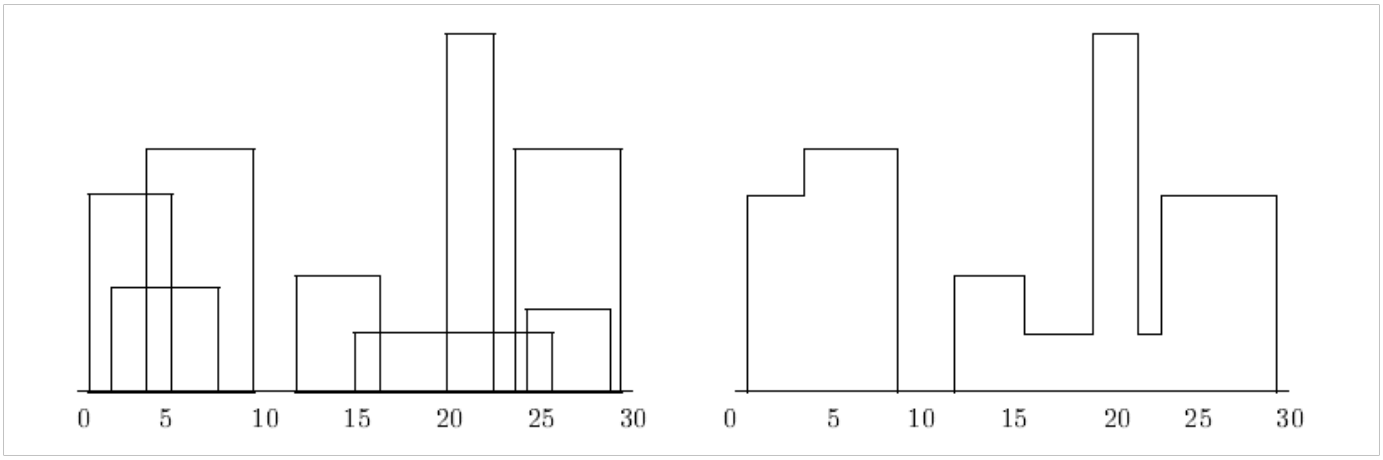

Figure 1: Ojbects and their display