Tutorial (week 12)

All the answers can be given with pseudocode or with Python. Remember that most of the time, many solutions are possible.

1. Characterize each of the following recurrence equations using the master theorem (assuming that $T(n) = c$ for $n < d$, for constants $c > 0$ and $d \geq 1$).

   a) $T(n) = 2T(n/2) + \log n$
   b) $T(n) = 8T(n/2) + n^2$
   c) $T(n) = 16T(n/2) + (n \log n)^4$
   d) $T(n) = 7T(n/3) + n$
   e) $T(n) = 9T(n/3) + n^3 \log n$

2. Use the divide-and-conquer algorithm, to compute $J \cdot I$, where $J =$ `10110011` and $I =$ `10111010` in binary. Show your work.

3. Use Strassen's matrix multiplication algorithm to multiply the matrices

$$X = \begin{pmatrix} 3 & 2 \\ 4 & 8 \end{pmatrix} \qquad\qquad Y = \begin{pmatrix} 1 & 5 \\ 9 & 6 \end{pmatrix}$$

4. A complex number $a + bi$, where $i = \sqrt{-1}$, can be represented by the pair (a, b). Describe a method performing only three real-number multiplications to compute the pair (e, f) representing the product of $a + bi$ and $c + di$.

5. What is the maximal set from the following set of points:

$$\{(7,2), (3,1), (9,3), (4,5), (1,4), (6,9), (2,6), (5,7), (8,6)\}$$

6. Give an example of a set of n points in the plane such that every point is a maximum point, that is, no point in this set is dominated by any other point in this set.

7. Consider the recurrence equation, $T(n) = 2T(n-1) + 1$, for $n > 1$, where $T(n) = 1$ for $n = 1$. Prove that $T(n)$ is $O(2^n)$.

8. A very common problem in computer graphics is to approximate a complex shape with a bounding box. For a set, S, of n points in $2-$dimensional space, the idea is to find the smallest rectangle, R, with sides parallel to the coordinate axes that contains all the points in S. Once S is approximated by such a bounding box, we can often speed up lots of computations that involve S. For example, if R is completely obscured some object in the foreground, then we don't need to render any of S. Likewise, if we shoot a virtual ray and it completely misses R, then it is guarantee to completely miss S. So doing comparisons with R instead of S can often save time. But this savings is wasted if we spend a lot of time constructing R; hence, it would be ideal to have a fast way of computing a bounding box, R, for a set, S, of n points in the plane. Note that the construction of R can be reduced to two instances of the problem of simultaneously finding the minimum and the maximum in a set of n numbers; namely, we need only do this for the x-coordinates in S and then for the y-coordinates in S. Therefore, design a divide-and-conquer algorithm for finding both the minimum and the maximum element of n numbers using no more than $3n/2$ comparisons.

Hints

- **Question 5: Recall when a point is dominated by another.**
- **Question 6: Review the concept of when one point dominates another.**
- **Question 7: Use a good induction hypothesis, such as $T(n) = 2^n - 1$.**
- **Question 8: Think about what happens in the base case.**