Recurrence Relations

Master Theorem

Recall the Master Theorem from lecture:
Theorem (Master Theorem). *Given a recurrence* $T(n) = aT(\frac{n}{b}) + O(n^d)$ *with* $a \geq 1$, $b > 1$ *and* $T(1) = \Theta(1)$, *then*

$$T(n) = \begin{cases} O(n^d \log n) & \textit{if } a = b^d \\ O(n^d) & \textit{if } a < b^d \\ O(n^{\log_b a}) & \textit{if } a > b^d \end{cases}.$$

What is the Big-Oh runtime for algorithms with the following recurrence relations?

1. $T(n) = 3T(\frac{n}{2}) + \Theta(n^2)$
2. $T(n) = 4T(\frac{n}{2}) + \Theta(n)$
3. $T(n) = 2T(\sqrt{n}) + O(\log n)$

Substitution Method

Use the Substitution Method to find the Big-Oh runtime for algorithms with the following recurrence relation:

$$T(n) = T\left(\frac{n}{3}\right) + n; \quad T(1) = 1$$

You may assume n is a multiple of 3, and use the fact that $\sum_{i=0}^{\log_3(n)} 3^i = \frac{3n-1}{2}$ from the finite geometric sum. Please prove your result via induction.

Divide and Conquer

Penguins in a Line

You arrive on an island of n penguins. All n penguins are standing in a line, and each penguin has a distinct height (i.e. no 2 penguins have the same height). A *local minimum* is a penguin that is shorter than both its neighbors (or one neighbor for the first and last penguin).

Design an efficient algorithm that takes as input an array of penguin heights, and finds a local minimum. Please give a clear English description, pseudocode, explanation of runtime, and a formal proof of correctness.

Maximum Sum Subarray

Given an array of integers $A[1..n]$, find a contiguous subarray $A[i, ..j]$ with the maximum possible sum. The entries of the array might be positive or negative.

1. What is the complexity of a brute force solution?
2. The maximum sum subarray may lie entirely in the first half of the array or entirely in the second half. What is the third and only other possible case?

3. Using the above, apply divide and conquer to arrive at a more efficient algorithm.
   (a) Prove that your algorithm works.
   (b) What is the complexity of your solution?
4. Advanced (Take Home) - Can you do even better using other non-recursive methods? ($O(n)$ is possible)

Light Bulbs and Sockets

You are given a collection of n differently sized light bulbs that have to be fit into n sockets in a dark room. You are guaranteed that there is exactly one appropriately-sized socket for each light bulb and vice versa; however, there is no way to compare two bulbs together or two sockets together as you are in the dark and can barely see! (You are, however, able to see where the sockets and light bulbs are.) You can try and fit a light bulb into a chosen socket, from which you can determine whether the light bulb's base is too large, too small, or is an exact fit for the socket. If the bulb fits exactly, it will flash once, in which case you have a correct match. (Note that the flashing light does not allow you to visually compare bulbs/sockets to other bulbs/sockets.) Suggest a (possibly randomized) algorithm to match each light bulb to its matching socket. Your algorithm should run strictly faster than quadratic time in expectation. Give an upper bound on the worst-case runtime, then prove your algorithm's correctness and expected runtime.