CS 161 Winter 2020 Section 2

January 16-17, 2020

Recurrence Relations

Master Theorem

Recall the Master theorem from lecture:

Theorem 0.1 *Given a recurrence* $T(n) = aT(\frac{n}{b}) + O(n^d)$ *with* $a \geq 1$, *and* $b > 1$ *and* $T(1) = \Theta(1)$, *then*

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

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:

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

$$T(1) = 1$$

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 series 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.