CS41 Lab 6: Recurrence Relations

Most of the lab problems this week center around recurrence relations. Recurrence relations are important tools for modeling and analyzing the efficiency of algorithms. The purpose of this lab is to gain practice using the different techniques for solving recurrence relations.

The following equalities might be helpful:

- For any $a, b > 0$, $\log(a^b) = b \log a$
- For any $a, b \geq 1$, $a^{\log b} = b^{\log a}$.
- For any $a, b \geq 1$, $\log(a \cdot b) = \log(a) + \log(b)$.
- For any $x \in \mathbb{R}$, $2^{\log x} = x$

Try to use both the Recursion Tree Method and the Substitution Method to solve these recurrence relations. You do not need to solve these exactly, only asymptotically. e.g. if a recurrence relation has an exact solution of $T(n) = 3n^2 - 19\sqrt{n}$, then an answer of $T(n) = O(n^2)$ suffices.

1. The following summation appears often in recurrence relations. Fix any constant $c > 0$. Using induction, show that for all $m > 0$, we have
$$\sum_{k=0}^{m-1} c^k = \frac{c^m - 1}{c - 1} \ .$$
2. $S(n) = S(n-1) + 3n$,
   $S(1) = 3$
3. $M(n) = 3M(n/2) + 2n$,
   $M(1) = 1$
4. $W(n) = 3W(n/3) + n^2$,
   $W(1) = 1$
5. $H(n) = 4H(n/2) + 2n^2$,
   $H(4) = 5$
6. $T(n) = 3T(n/3) + 10\sqrt{n}$,
   $T(1) = 5$.
7. An investment company has designed several new trading strategies and wants to compare them. Unfortunately, the strategies were tested on different commodities over different time periods, so the total profit earned is not a fair comparison. Instead, the company wants to compare each strategy's profit to the maximum possible profit that could have been made (with perfect hindsight) during the same time span. Each strategy was allowed to buy a fixed amount of the commodity once and then sell what it bought at some later date.

   During each testing period, the investment company recorded the minimum and maximum prices that were reached each day. Your algorithm receives a list of n pairs, where the i^{th} pair has the minimum price of the commodity and the maximum price of the commodity on day i. Your algorithm should output the largest price difference that could have been achieved by buying at the minimum price on day i and selling at the maximum price on day $j > i$.

For example, suppose $n = 3$ and the (min, max) prices were $[(8, 14), (1, 2), (4, 6)]$. Then you should return a per-unit profit of 5, corresponding to buying for 1 on day 2 and selling for 6 on day three (recall that the trading strategies must buy first, and can't sell on the same day).

Clearly, there is a simple algorithm that takes time $O(n^2)$: try all possible pairs of buy/sell days and see which makes the most money. Design a divide and conquer algorithm to determine the best profit in hindsight more efficiently. Set up a recurrence that describes the running time of your algorithm, and solve it to show that your algorithm is faster than $O(n^2)$.