Ruolin Li
31764160

CPSC 320 2020S2: Tutorial 7

1 Practice at Solving Recurrences

A student in the class has proposed a sophisticated algorithm to predict whether or not there will be a heat wave during the final, based on data from n previous years. The recurrence of the student's algorithm is as follows:

$$T(n) = \begin{cases} 2T(n/4) + T(3n/4) + cn^2, & \text{if } n \geq 2 \\ c, & \text{if } n = 1. \end{cases} \tag{1}$$

1. You want to get an upper bound on $T(n)$. Since the Master Theorem does not handle recurrences like this, with two terms involving $T()$ on the right hand side, you decide to work with the following recurrence:

$$T(n) \leq \begin{cases} 3T(3n/4) + cn^2, & \text{if } n \geq 2 \\ c, & \text{if } n = 1. \end{cases} \tag{2}$$

   Apply the Master Theorem to solve recurrence (2).

2. Draw level 0 (the root), level 1 and level 2 of the recursion tree for the original recurrence (1). Within each node, write (i) the size of the subproblem at this node and (ii) the time needed for the subproblem at this node (not counting times at deeper levels of recursion).

3. As a function of n, what is the total time needed at levels 1, 2 and 3 of the tree?

   Level 1: ☐ Level 2: ☐ Level 3: ☐

4. Generalizing the pattern from the first three levels of the tree, write down the total time needed at level i of the tree.

   Level i: ☐

5. Write down a sum that upper bounds the total time over all levels of the tree.

6. Use part 5 to provide a big-O bound on the running time of the algorithm as a function of n.

7. Which method leads to a better bound, the Master Theorem or the recursion tree?

Q1

1. Solving Recurrences

$$T(n) \leq \begin{cases} 3T(3n/4) + cn^2, & n \geq 2 \\ c & n = 1 \end{cases}$$

$a=3 \quad b=4/3 \quad k=2$

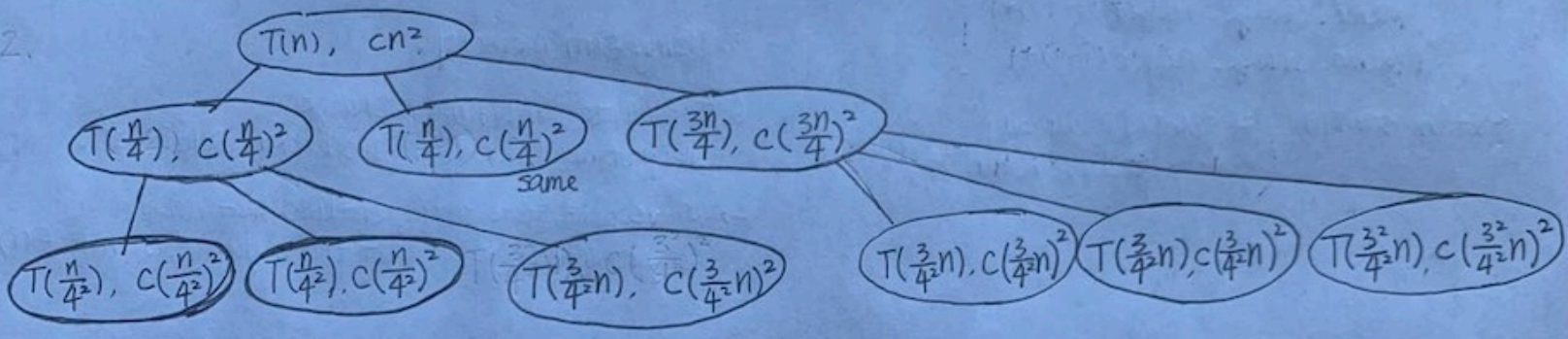
$b^k = (\frac{4}{3})^2 = 1.77$

So $a > b^k$.

Case 1 of Master Theorem applies.

$\boxed{O(n \log_{\frac{4}{3}} 3)}$

2.



Work done at level 0 : cn^2

Work done at level 1 : $c(\frac{11}{16})n^2$

Work done at level 2 : $c(\frac{11}{16})^2 n^2$

work done at level i : $c(\frac{11}{16})^i n^2$

Total work : $\sum_{i=0}^{\infty} cn^2(\frac{11}{16})^i = cn^2 \frac{1-(\frac{11}{16})^{i+1}}{1-\frac{11}{16}} = \frac{16}{5}cn^2$ $\boxed{O(n^2)}$

Master Theorem : $O(n^{\log_{\frac{4}{3}} 3})$ So recursion tree is the tighter bound.

Recursion tree : $O(n^2)$

2 Moving Mattresses

You have been hired by *thekingboomattress.com*, a mattress selling company, to preform a data analysis operation. Over the next several months, the company has decided to host weekend sales where some number of items will be on sale. Each weekend, the company can either have a *small*, *medium*, or *large* sale, where a small, medium, and large number of items will be on sale.

For each weekend you are given an estimate of the profit that will be made for each type of sale. That is, for weekend i, you are given values s_i, m_i, l_i that represent the respective profits for each type of sale. Note that these values may be negative if a loss would be incurred. If no sale occurs on weekend i, the the profit for that weekend is zero.

The Boo-King himself however is concerned about having too many consecutive medium and large sales may result in the website running out of inventory for the following weekend! As such the following constraints are present:

- If a large sale occurs on weekend i, then on both weekends $i-1$ and $i-2$, no sale can occur.
- If a medium sale occurs on weekend i, then on weekend $i-1$ no sale can occur.

There is sufficient inventory for the first two weekends so **the above constraints only apply for weekends** $i \geq 3$.

Given the estimated profit for each sale type over an n weekend period, your goal is to determine what sale should occur for each weekend, if any, such that the estimated profit is maximized and the above constraints are realized.

1. Let $n = 2$. (Note that the inventory constraints do not apply in this case.) Using the math operators max and $+$, write down an expression for the maximum profit over the two weekends.
2. Now let $n = 3$. First, use math operators to describe the max possible profit (as a function of the profit estimates s_i, m_i and l_i, $1 \leq i \leq 3$). Then describe how you would use this expression to determine which type of sale to have on each weekend.

Q2.

① $\text{maxprofit} = \max\{0, s_1, m_1, l_1\} + \max\{0, s_2, m_2, l_2\}$

② $$\text{maxprofit} = \begin{cases} l_3 \\ m_3 + \max\{0, s_1, m_1, l_1\} \\ s_3 + \max\{0, s_1, m_1, l_1\} + \max\{0, s_2, m_2, l_2\} \\ 0 + \max\{0, s_1, m_1, l_1\} + \max\{0, s_2, m_2, l_2\} \end{cases}$$

3. Consider the following `GreedySales` algorithm to solve the Moving Mattresses problem when $n \geq 3$. What is the output of the algorithm on the following problem instance, where $n = 3$? Do you think that the algorithm is correct?

Days:	1	2	3
S	1	1	99
M	-1	-1	1
L	1	1	100

---

procedure GREEDYSALES(n weekends, estimated profit for each weekend) # $n \geq 3$
    $i \leftarrow n$
    $netProfit \leftarrow 0$
    while $i > 0$ **do**
        $var \leftarrow \max\{0, s_i, m_i, l_i\}$
        if $var == l_i$ **then**
            $netProfit \leftarrow netProfit + l_i$
            $i \leftarrow i - 3$
        else if $var == m_i$ **then**
            $netProfit \leftarrow netProfit + m_i$
            $i \leftarrow i - 2$
        else if $var == s_i$ **then**
            $netProfit \leftarrow netProfit + s_i$
            $i \leftarrow i - 1$
        else
            $i \leftarrow i - 1$
        end if
    end while
    return $netProfit$
end procedure

var = 100

netProfit = 100

i = 0

since i = 0, the while loop breaks so we return netProfit

But based on the dynamic programming, the maximum sale value we can get is 1+1+99 = 101, so greedy algorithm only focuses on the maximum value at current step, and dp will consider how to reach maximum result during the whole process