CSCI 3104 Fall 2022
Instructors: Prof. Grochow and Nagesh

Problem Set 6

Due Date .... October 17, 2022
Name .... **Daniel Lee**
Student ID **109836390**
Collaborators .... **List Your Collaborators Here**

Contents

Instructions **1**

Honor Code (Make Sure to Virtually Sign the Honor Pledge) **2**

16 Standard 16 - Analyzing Code: Writing Down Recurrences **3**
16(a)Problem 16(a) 3
16(b)Problem 16(b) 5

17 Standard 17 - Solving Recurrences I: Unrolling **7**
17(a)Problem 17(a) 7
17(b)Problem 17(b) 9

18 Standard 18 - Divide and Conquer: Counterexamples **10**

Instructions

- The solutions **must be typed**, using proper mathematical notation. We cannot accept hand-written solutions. Useful links and references on LaTeXcan be found here on Canvas.
- You should submit your work through the **class Canvas page** only. Please submit one PDF file, compiled using this LaTeX template.
- You may not need a full page for your solutions; pagebreaks are there to help Gradescope automatically find where each problem is. Even if you do not attempt every problem, please submit this document with no fewer pages than the blank template (or Gradescope has issues with it).
- You are welcome and encouraged to collaborate with your classmates, as well as consult outside resources. You must **cite your sources in this document. Copying from any source is an Honor Code violation. Furthermore, all submissions must be in your own words and reflect your understanding of the material.** If there is any confusion about this policy, it is your responsibility to clarify before the due date.

- Posting to **any** service including, but not limited to Chegg, Reddit, StackExchange, etc., for help on an assignment is a violation of the Honor Code.
- You **must** virtually sign the Honor Code (see Section Honor Code). Failure to do so will result in your assignment not being graded.

Honor Code (Make Sure to Virtually Sign the Honor Pledge)

Problem HC. On my honor, my submission reflects the following:

- My submission is in my own words and reflects my understanding of the material.
- Any collaborations and external sources have been clearly cited in this document.
- I have not posted to external services including, but not limited to Chegg, Reddit, StackExchange, etc.
- I have neither copied nor provided others solutions they can copy.

In the specified region below, clearly indicate that you have upheld the Honor Code. Then type your name.

Honor Pledge. I agree to the above, Daniel Lee ☐

16 Standard 16 - Analyzing Code: Writing Down Recurrences

Problem 16. For each algorithm, write down the recurrence relation for the number of times they print "Welcome". (In this case, this is big-Θ of the runtime - do you see why?) **Don't forget to include the base cases.**

16(a). Problem 16(a)

Algorithm 1 Writing Recurrences 1

(a)
```
procedure Foo(Integer $n$)
    if $n \leq 5$ then return n

    Foo($n/10$)
    Foo($n/5$)
    Foo($n/5$)

    for $i \leftarrow 1; i \leq 3 * n; i \leftarrow i + 1$ do
        print "Welcome"
```

Answer.

Starting with the for loop: **for** $\{i \leftarrow 1; i \leq 3 * n; i \leftarrow i + 1\}$ **do**

print "Welcome" is $O(1)$.
i initialization is $O(1)$
$i \leq 3$ comparison is $O(1)$
$i \leftarrow i + 1$ is $O(2)$

Number of steps when n = 1

Steps	i
1	1
2	2
3	3

Number of steps taken is 3n.

Therefore the total run time of the for loop is equal to

$$T(n) = 1 + \sum_{i=1}^{3n}(1 + 1 + 2) = 1 + \sum_{i=1}^{3n}(4)$$
$$= 1 + 12n \in \Theta(n)$$

and **print** "Welcome" is ran $\sum_{i=1}^{3n}(1) = 3n$ times

The base case is $n \leq 5$, **return** n takes $O(1)$ or T(n) = 1, but the problem is asking for times printed so it is 0.

Foo($n/10$) is the recursive case T(n/10)
Foo($n/5$) is the recursive case T(n/5)

Foo($n/5$) is also the recursive case T(n/5)

The recurrence relation for the recursive cases is 2T(n/5) + T(n/10) + 3n

Therefore, the total recurrence relation is :

$$T(n) = \begin{cases} 0 & : n \leq 5, \\ 2T(n/5) + T(n/10) + 3n & : n > 5. \end{cases}$$

□

16(b). Problem 16(b)

Algorithm 2 Writing Recurrences 2

(b)
```
procedure Foo2(Integer $n$)
    if $n \leq 7$ then return

    Foo2($n/14$)
    Foo2($n/14$)

    for $i \leftarrow 1; i \leq n; i \leftarrow i * 2$ do
        for $j \leftarrow 1; j \leq n; j \leftarrow j * 3$ do
            print "Welcome"
```

Answer.

Starting with the inner for loop: **for** $\{j \leftarrow 1; j \leq n; j \leftarrow j * 1\}$ **do**

print "Welcome" is $O(1)$.
j initialization is $O(1)$
$j \leq n$ comparison is $O(1)$
$j \leftarrow j * 3$ is $O(2)$

Number of steps when n = 9

Steps	j
1	1
2	3
3	9

Number of steps taken is $\lfloor \log_3 n \rfloor + 1$.

Only looking at the run time of **print** "Welcome"

$$T(n) = \sum_{i=1}^{\lfloor \log_3 n \rfloor + 1} (1) = \lfloor \log_3 n \rfloor + 1 \in \Theta(\log_3 n)$$

Looking at the outer loop: **for** $\{i \leftarrow 1; i \leq n; i \leftarrow i * 2\}$ **do**

The inner loop is $\lfloor \log_3 n \rfloor + 1 \in \Theta(\log_3 n)$
i initialization is $O(1)$
$i \leq n$ comparison is $O(1)$
$i \leftarrow i * 2$ is $O(2)$

Number of steps when n = 4

Steps	i
1	1
2	2
3	4

Number of steps taken is $\lfloor \log_2 n \rfloor + 1$.

The total run time of the nested loops is

$$\begin{aligned}T(n) = 1 + \sum_{i=1}^{\lfloor \log_2 n \rfloor + 1} (1 + 2 + \lfloor \log_3 n \rfloor + 1) &= 1 + \sum_{i=1}^{\lfloor \log_2 n \rfloor + 1} (4 + \lfloor \log_3 n \rfloor) \\ &= 1 + 4 + 4\lfloor \log_2 n \rfloor + \lfloor \log_3 n \rfloor + \lfloor \log_2 n \rfloor \lfloor \log_3 n \rfloor \\ &= 5 + 4\lfloor \log_2 n \rfloor + \lfloor \log_2 n \rfloor \lfloor \log_3 n \rfloor \in \Theta((\log_2 n)(\log_3 n))\end{aligned}$$

The base case $n \leq 7$ **return** is $O(1)$ or T(n) = 1, but the problem is asking for times printed so it is 0.

Foo($n/14$) is the recursive case T(n/14)
Foo($n/14$) is also the recursive case T(n/14)

The recurrence relation for the recursive cases is 2T(n/14) + $(\log_2 n)(\log_3 n)$

Therefore the recurrence relation for this function is :

$$T(n) = \begin{cases} 0 & : n \leq 7, \\ 2T(n/14) + (\log_2 n)(\log_3 n) & : n > 7. \end{cases}$$

□

17 Standard 17 - Solving Recurrences I: Unrolling

Problem 17. For each of the following recurrences, solve them using the unrolling method (i.e. find a suitable function $f(n)$ such that $T(n) \in \Theta(f(n))$). **Note:** Show all the work.

17(a). Problem 17(a)

a.

$$T(n) = \begin{cases} 3n & : n < 3, \\ 2T(n/2) + 4n & : n \geq 3. \end{cases}$$

Answer.

for n $\geq$ 3:

T(n) = 2T(n/2) + 4n
T(n/2) = 2[2T(n/2^2) + 4(n/2)] + 4n
T(n/2^2) = 2^2[2T(n/2^3)+4(n/2^2)] + 4n + 4n

T(n) = 2^kT(n/2^k)+4k(n)

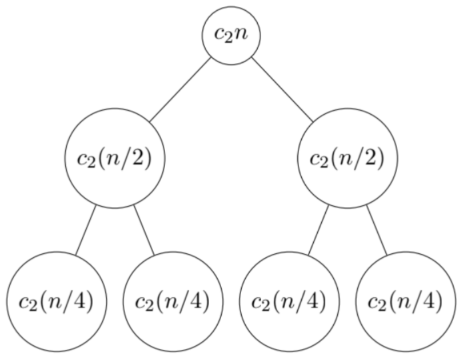
Suppose n/2^k < 3,
2^k > n/3
k > $\log_2(n/3)$ = $\lceil\log_2(n/3)\rceil$

It will take $\lceil\log_2(n/3)\rceil$ to reach the Base Case

Now looking at the relation as

$$T(n) = \begin{cases} c_1 n & : n < 3, \\ 2T(n/2) + c_2 n & : n \geq 3. \end{cases}$$

We can make a tree to see the number of times the non-recursive time takes.



For $i \in k$, we can see that at the ith layer, it is called $c_2 \cdot n/2^i$ times. Also, there are 2^i nodes, so the total time of the non-recursive calls is $c_2 \cdot n/2^i \cdot 2^i = c_2 n$. For the base case, it will take $c_1 \cdot n/2^i \cdot 2^i = c_1 n$.

Using $\lceil \log_2(n/3) \rceil$, we get

$$\begin{aligned} T(n) &= \sum_{i=0}^{\lceil \log_2(n/3) \rceil} (c_1 n + c_2 n) \\ &= c_1 n \log_2(n/3) + c_2 \log_2(n/3) \\ &= (c_1 + c_2) n \log_2 n/3 \\ &= (c_1 + c_2) n \log_2 n - (c_1 + c_2) n \log_2 3 \end{aligned}$$

Therefore, T(n) $\in \Theta(n \log_2(n))$ $\square$

17(b). Problem 17(b)

b.

$$T(n) = \begin{cases} 5 & : n < 2, \\ 7T(n-2) + 9 & : n \geq 2. \end{cases}$$

Answer.

Base Case: T(n) = 5 when n < 2.

for n $\geq$ 2:

T(n) = 7T(n-2) + 9
T(n-2) = 7[7(n-4) + 9] + 9
T(n-4) = 7^2[7T(n-6) + 9] + 18

T(n) = 7^kT(n-2k) + 9 $\sum_{i=0}^{k-1} 7^i$

Suppose n-2k < 2,
k > $\frac{n-2}{2} = \lfloor \frac{n-2}{2} \rfloor + 1$

Plug in $\lfloor \frac{n-2}{2} \rfloor + 1$ for k

$$\begin{aligned} T(n) &= 7^{\frac{n-2}{2}+1} T(n - 2(\frac{n-2}{2} + 1)) + 9 \sum_{i=0}^{\frac{n-2}{2}} (7^i) \\ &= 7^{\frac{n-1}{2}} \cdot 5 + 9 \cdot (7^{\frac{n-2}{2}+1} - 1)/6 \\ &= 5 * 7^{\frac{n-2}{2}} + 7^{\frac{n-2}{2}} \frac{9}{6} n - \frac{9}{6} \\ &= \frac{31}{2} 7^{\frac{n-2}{2}} - \frac{9}{6} \end{aligned}$$

Therefore, T(n) $\in \Theta(7^{n/2})$ □

18 Standard 18 - Divide and Conquer: Counterexamples

Problem 18. Consider the following problem:

> MAX PAIR SUM
> *Input:* A list L of integers
> *Output:* An index $i \in \{1, \dots, len(L) - 1\}$ such that $L[i] + L[i+1]$ is maximized (that is, such that $L[i] + L[i+1] \geq L[j] + L[j+1]$ for all j), and the value of $L[i] + L[i+1]$

(Note the list here is 1-indexed, so the problem simply does not consider the last element, as it has nothing to pair it with.)

Consider the algorithm below that attempts to solve this problem. **Give an instance** of input (preferably a list of length at most 6) for which it fails to output the correct value for the above problem, and **explain why it fails**.

Algorithm 3 Proposed divide-and-conquer algorithm for the Max Pair Sum problem

```
procedure MaxPairSum(List $L$) $n \leftarrow len(L)$
    if $n \leq 1$ then return ;
    if $n = 2$ then return $(1, L[1] + L[2])$;
    $(i1, sum1) \leftarrow$ MaxPairSum($L[1..\lfloor n/2 \rfloor]$);
    $(i2, sum2) \leftarrow$ MaxPairSum($L[\lfloor n/2 \rfloor + 1..n]$);
    if $i1 \geq i2$ then
        return $(i1, sum1)$;
    else
        return $(i2, sum2)$;
```

Proof.

Suppose the list L = [1, 2, 4, 1]
n = len(L) = 4

The algorithm splits the list in half using the recursive call MaxPairSum(1..$\lfloor n/2 \rfloor$) and MaxPairSum($\lfloor n/2 \rfloor$+1..n)

The new lists L_1 = [1,2] and L_2 = [4,1] lengths are compared. len(L_1) and len(L_2) both are n = 2.

According to the algorithm, when n = 2, return 1 and (index 1 + index 2), so L_1 is (1,3) and L_2 = (1,5). Then the algorithm compares the indexes of the 2 lists.

If 1 $\geq$ 1, return (1,3), else return (1,5). The algorithm returns (1,3). This is incorrect because the pair [2,4] has the max sum of 6 which is greater than 5 or 3.

Ultimately, this algorithm fails because when the list is even, it skips comparing the pair ($\lfloor n/2 \rfloor$, $\lfloor n/2 \rfloor$ + 1), and if this pair contains the max pair, the algorithm never returns it.

Also is

```
if $i1 \geq i2$ then
    return $(i1, sum1)$;
else
    return $(i2, sum2)$;
```

a typo? If $i1 \geq i2$, then for lists of size 2, it will return the first list every time. Did the algorithm mean to say

```
if $sum1 \geq sum2$ then
    return $(i1, sum1)$;
else
    return $(i2 + \lfloor (n/2) \rfloor, sum2)$;
```

$\square$