CSCI 3104 Fall 2022
Instructors: Prof. Grochow and Nagesh

Problem Set 6

Due Date .... October 17, 2022
Name .... **Tyler Huynh**
Student ID **109603994**
Collaborators .... **N/A**

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 **11**

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, **Tyler Huynh** on my honor pledge that my submission is a reflection of my own understanding of the material, any and all collaborations/sources have been properly cited, I have not posted any material to external sources, and I have not copied other solutions as my own. □

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)

```
1: procedure Foo(Integer n)
2:     if n ≤ 5 then return n
3:
4:     Foo(n/10)
5:     Foo(n/5)
6:     Foo(n/5)
7:
8:     for i ← 1; i ≤ 3 * n; i ← i + 1 do
9:         print "Welcome"
```

Answer. **Referenced Levet Notes**
Base Case:
For the base case it will just return n, such that it will be:

$$
\begin{aligned}
T(0) &= \Theta(1) \\
T(1) &= \Theta(1) \\
T(2) &= \Theta(1) \\
T(3) &= \Theta(1) \\
T(4) &= \Theta(1) \\
T(5) &= \Theta(1)
\end{aligned}
$$

From the above that we can see that for the base case when it returns n it will take $\Theta(1)$ to run.

Recursive Case:
-On line 8 for the recursive case the function will take 1 step to initialize i at 1
-It will also take 2 steps, 1 step to evaluate $i + 1$ and 1 more step for the assignment of $i \leftarrow$ on line 8
-It will take 1 step to print "Welcome" on line 9

The function will run a total 3n times.
From this we can find the total runtime complexity of the for loop:

$$
\begin{aligned}
1 + \sum_{i=1}^{3n}(1 + 2 + 1) &= 1 + \sum_{i=1}^{3n} 4 \\
&= 1 + 12n \\
&= \Theta(n)
\end{aligned}
$$

I will now find the runtime complexity of the function in general:

$$\begin{aligned} T(n) &= T(\frac{n}{10}) + T(\frac{n}{5}) + T(\frac{n}{5}) + \Theta(n) \\ &= 2T(\frac{n}{5}) + T(\frac{n}{10}) + \Theta(n) \end{aligned}$$

$$T(n) = \begin{cases} \Theta(1) & : n \leq 5, \\ 2T(\frac{n}{5}) + T(\frac{n}{10}) + \Theta(n) & : n > 5. \end{cases}$$

From the above this is final answer for the runtime complexity of the function. $\square$

16(b). Problem 16(b)

Algorithm 2 Writing Recurrences 2

(b)

```
1: procedure Foo2(Integer n)
2:     if n ≤ 7 then return
3:
4:     Foo2(n/14)
5:     Foo2(n/14)
6:
7:     for i ← 1; i ≤ n; i ← i * 2 do
8:         for j ← 1; j ≤ n; j ← j * 3 do
9:             print "Welcome"
```

Answer. **Base Case:**
For the base case it will just, such that:

$$T(0) = \Theta(1)$$
$$T(1) = \Theta(1)$$
$$T(2) = \Theta(1)$$
$$T(3) = \Theta(1)$$
$$T(4) = \Theta(1)$$
$$T(5) = \Theta(1)$$
$$T(6) = \Theta(1)$$
$$T(7) = \Theta(1)$$

From the above we can see that for the base case when it returns n it will take $\Theta(1)$ to run.

Recursive Case:
Inner Loop:
-We see that initializing j at 1 will take 1 step on line 8
-We see that the evaluation of j * 3 will take 1 step and the 1 more step for the assignment of $j \leftarrow$
-We see that the print statement "Welcome" will take 1 step
-The inner loop will run $\log_3 n$ times

Outer Loop:
-We see that initializing i at 1 will take 1 step on line 7
-We see that the evaluation of i * 2 will take 1 step and the 1 more step for the assignment of $i \leftarrow$
-For the rest of the steps it will exist within the inner loop of j
-The outer loop will run $\log_2 n$ times

I will now find the runtime complexity of the inner loop:

$$\begin{aligned} 1 + \sum_{j=1}^{\log_3 n} (1 + 2 + 1) &= 1 + \sum_{j=1}^{\log_3 n} 4 \\ &= 1 + 4(\log_3 n) \\ &= 1 + 4 \log_3 n \\ &= \Theta(\log n) \end{aligned}$$

Runtime complexity of the outer loop:

$$
\begin{aligned}
1 + \sum_{i=1}^{\log_2 n} (1+2) + (1 + (4\log_3 n)) &= 1 + \sum_{i=1}^{\log_2 n} (1+2) + (1 + (4\log_3 n)) \\
&= 1 + \sum_{i=1}^{\log_2 n} 4 + (4\log_3 n) \\
&= 1 + \sum_{i=1}^{\log_2 n} 4 + 4\sum_{i=1}^{\log_2 n} \log_3 n \\
&= 1 + 4(\log_2 n) + 4(\log_2 n)(\log_3 n) \qquad \log_3 n = \frac{\log_2 n}{\log_2 3} \\
&= 1 + 4\log_2 n + 4(\log_2 n)(\frac{\log_2 n}{\log_2 3}) \\
&= 1 + 4\log_2 n + (\frac{4}{\log_2 3})((\log_2 n)^2) \\
&= \Theta((\log n)^2)
\end{aligned}
$$

From the above we have found the run time of the two loops, I will now write out our recurrence relation:

$$
\begin{aligned}
T(n) &= T(\frac{n}{14}) + T(\frac{n}{14}) + \Theta((\log_n)^2) \\
&= 2T(\frac{n}{14}) + \Theta((\log_n)^2)
\end{aligned}
$$

Thus, we have our final runtime complexity of the function:

$$
T(n) = \begin{cases} \Theta(1) & : n \leq 7, \\ 2T(\frac{n}{14}) + \Theta((\log_n)^2) & : n > 7. \end{cases}
$$

$\square$

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. I will first begin by finding how many times we must unroll up to k:

We will hit a base case when:
$\frac{n}{2^k} < 3$ I will now solve for k such that:
$k > \log_2(\frac{n}{3})$

I will now begin the unrolling process:

$$\begin{aligned} T(n) &= 2T(\frac{n}{2}) + 4n \\ T(\frac{n}{2}) &= 2T(\frac{\frac{n}{2}}{2}) + 4(\frac{n}{2}) \\ &= 2T(\frac{n}{4}) + 4(\frac{n}{2}) \\ T(\frac{n}{4}) &= 2T(\frac{\frac{n}{4}}{2}) + 4(\frac{n}{4}) \\ &= 2T(\frac{n}{8}) + 4(\frac{n}{4}) \end{aligned}$$

From the above I will now find the function such that $T(n) \in \Theta(f(n))$:

$$\begin{aligned} T(n) &= 2T(\frac{n}{2}) + 4n \\ &= 2[2T(\frac{n}{4}) + 4(\frac{n}{2})] + 4n \\ &= 2[2[2T(\frac{n}{8}) + 4(\frac{n}{4})] + 4(\frac{n}{2})] + 4n \\ &= 2^3(T(\frac{n}{8})) + 2^2(4n) + 2^1(4n) + 2^0(4n) \end{aligned}$$

I will now sum this up into a summation:

$$T(n) = \sum_{i=0}^{\log_2(\frac{n}{3})} 2^i * 4n$$

$$
\begin{aligned}
T(n) &= (\text{cost of base case}) \cdot (\text{number of times the base case is reached}) + \sum_{i=0}^{\log_2(\frac{n}{3})} 2^i * 4n \\
&= 3n * 2^{\log_2(\frac{n}{3})} + \sum_{i=0}^{\log_2(\frac{n}{3})} 2^i * 4n \\
&= 3n * 2^{\log_2(\frac{n}{3})} + 4n(\frac{1 - 2^{\log_2(\frac{n}{3})-1}}{1-2}) \\
&= 3n * 2^{\log_2(\frac{n}{3})} + 4n(\frac{1 - 2^{\log_2(\frac{n}{6})}}{-1}) \\
&= 3n * 2^{\log_2(\frac{n}{3})} + \frac{4n - 4n(2^{\log_2(\frac{n}{6})})}{-1} \\
&= 3n * \frac{n}{3} + \frac{4n - 4n(\frac{n}{6})}{-1} \\
&= n^2 + -4n + \frac{2n^2}{3}
\end{aligned}
$$

From the above we can see that the highest component of the function would be:

$$T(n) = \Theta(n^2)$$

Thus, my final answer is $T(n) = \Theta(n^2)$ $\square$

17(b). Problem 17(b)

b.

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

Answer. We will now find how many times we must unroll first up to k:

We will hit a base case when:
$n - 2k < 2$ I will now solve for k such that:
$k > \frac{n-2}{2}$

From the piecewise function we see that 2 will be subtracted multiple times in a row up to k, such that:

$$T(n-2) = n - 2 - 2 - 2 - 2 - 2 - 2 - 2$$
$$n - 2 - 2 - 2 - 2 - 2 - 2 - 2 = k$$

From this we can find the steps of $T(n)$, such that:

$$\begin{aligned} T(n) &= 7T(n-2) + 9 \\ T(n-2) &= 7T(n-4) + 9 \\ T(n-4) &= 7T(n-6) + 9 \\ T(n-6) &= 7T(n-8) + 9 \end{aligned}$$

From this we can then find the function such that $T(n) \in \Theta(f(n))$:

$$\begin{aligned} T(n) &= 7T(n-2) + 9 \\ &= 7[7T(n-4) + 9] + 9 \\ &= 7[7[7T(n-6) + 9] + 9] + 9 \\ &= 7^3(T(n-6)) + 7^2(9) + 7^1(9) + 7^0(9) \end{aligned}$$

To save us time we can sum this data up into a summation such that:

$$T(n) = \sum_{i=0}^{\frac{n-2}{2}} 7^i * 9$$

$$
\begin{aligned}
T(n) &= (\text{cost of base case}) \cdot (\text{number of times the base case is reached}) + \sum_{i=0}^{\frac{n-2}{2}} 7^i * 9 \\
&= 5 * 7^{\frac{n-2}{2}} + 9 * \sum_{i=0}^{\frac{n-2}{2}} 7^i \\
&= 5 * 7^{\frac{n-2}{2}} + 9 * \frac{1 - 7^{\frac{n-2}{2}+1}}{1-7} \\
&= 5 * 7^{\frac{n-2}{2}} + \frac{9 - 9 * 7^{\frac{n-2}{2}+\frac{2}{2}}}{-6} \\
&= 5 * 7^{\frac{n-2}{2}} + \frac{9 - 9 * 7^{\frac{n}{2}}}{-6}
\end{aligned}
$$

From the above we can observe that the highest component of the function is:

$$T(n) = \Theta(7^{\frac{n}{2}})$$

Thus, my final answer is $T(n) = \Theta(7^{\frac{n}{2}})$ $\square$

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 $i \in \{1, \ldots, len(L) - 1\}$ 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. Referenced Levet Notes
Here we will create our input list at length 6 such that:

[7, 9, 6, 4, 3, 2]

- (i1, sum1) = MaxPairSum([7, 9, 6])
  - (i1, sum1) = MaxPairSum([7]) - This would just return because the length of the list is 1.
  - (i2, sum2) = MaxPairSum([9, 6])
    - (1, 15) - This would be returned because this is the sum of index 1 and index 2 and the length of the list is 2.
- (i2, sum2) = MaxPairSum([4, 3, 2])
  - (i1, sum1) = MaxPairSum([4]) - This would just return because the length of the list is 1.
  - (i2, sum2) = MaxPairSum([3, 2])
    - (1, 5) - This would be returned because this is the sum of index 1 and index 2 and the length of the list is 2.

i1 $\geq$ i2 Where i1 and i2 are both equal to 1

- (1, 15) - This comparison returns (1, 15), but the MaxPairSum for our list should be (1, 16), thus this function fails.

From the above we can see that the function fails because it is not finding the MaxPairSum of the list that it is being given. This is because on line 6 by comparing the index 1 and index 2 it does not allow for us to find the MaxPairSum, because we should be comparing the sums at these indexes instead.

Also, the indexes that are returned are not relative to the entire list itself, but instead it is relative to the sublist that arises when it is divided that is passed into the function. Further, depending on how the odd length list is divided it will influence what is returned, since some comparisons will not computed such that:

If we had partitioned it into ([7, 9]) and ([6]) we would have arrived at the correct answer.

□