CSCI 3104 Fall 2021
Profs Josh Grochow and Bo Waggoner

Problem Set 6

Due Date ..... TODO
Name ..... **John Blackburn**
Student ID **Jobl2177**
Collaborators ..... **none**

Contents

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 have upheld the honor code, John Blackburn □

1 Standard 16 - Analyzing Code III: (Writing down recurrences)

Problem 1. Write down the recurrence relation for the runtime complexity of these algorithms. **Don't forget to include the base cases.**

Algorithm 1 Writing Recurrences 1

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

Answer. The recurrence relation is as follows:

The base case is: $\theta(1) : n \leq 4$

The are 2 recursive calls within the function, therefore the rest of the recurrence relation is: $2T(n/3) + \theta(n) : n > 4$

The $\theta(n)$ comes from the rest of the function. The work to find that is below:

It takes one step to compare whether n is less than or equal to four

one step to initialize i in the for loop

the loop will run $2n$ times

within the loop:

one step to compare if i is above $2n$

one step for the print

two steps for the update and assignment of i

So, overall the time it takes for everything other than the recursive calls is:

$1 + 1 + \sum_{k=0}^{2n} 4 = 2 + 8n$

and from here we can gather that the function will be within $\theta(n)$ but just to check I will bound the limit of $2 + 8n$ over n.

$\lim_{n\to\infty} \frac{2+8n}{n}$

I have to do L'hopitals rule to find the limit, so I take the derivative of the top and bottom to get:

$\lim_{n\to\infty} \frac{8}{1} = 8$

and since the limit is equal to a constant we can know for sure that the rest of the function is bounded by $\theta(n)$

and thus sums up the work for my recurrence relation

□

Algorithm 2 Writing Recurrences 2

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

Answer. The recurrence relation is as follows:

The base case is: $\theta(1) : n \leq 4$

The are 3 recursive calls within the function, therefore the rest of the recurrence relation is: $T(n/4)+2T(n/3)+\theta(\log(n)) : n > 4$

The $\theta(\log(n))$ comes from the rest of the function.

□

2 Standard 17 - Unrolling

Problem 2. Consider the following recurrences and solve them using the unrolling method (i.e. find a suitable function $f(n)$ such that $T(n) \in \Theta(f(n))$).

(a)

$$T(n) = \begin{cases} 3 & : n < 2, \\ 2T(n-2)+1 & : n \geq 2. \end{cases}$$

Answer. First I determine the number of times the relation needs to be unrolled. Let k denote the amount of times the function needs to be unrolled. I hit a base case when $n - 2k < 2$. Now, solving for k I get that $k > (n/2) - 1$. So I will unroll until I get $k = (n/2) - 1$.

Next, I will unroll starting at $T(n - 2i)$ to help determine a pattern.

$T(n - 2i) = 2T(n - 2(i + 1)) + 1$

Unroll again:

$= 2(2T(n - 2(i + 2) + 1) + 1$

I can now see the pattern that is forming, and can move to finalize the $T(n)$ function

$T(n)$ =(cost of the base case) * (number of times the base case is reached) + $\sum_{i=0}^{(n/2)-1} 2^i$

$T(n) = 3 * 2^{(n/2)-1} + 2^{n/2} - 1$

Now to find $f(n)$ such that $T(n) \in \Theta(f(n))$

To do so I will just take the highest order term within $T(n)$

That gives me $T(n) \in \Theta(2^{n/2})$ □

(b)

$$T(n) = \begin{cases} 4 & : n < 3, \\ T(n-3) + 5n & : n \geq 3. \end{cases}$$

Answer. First I determine the number of times the relation needs to be unrolled. Let k denote the amount of times the function needs to be unrolled. I hit a base case when $n - 3k < 3$. Now, solving for k I get that $k > (n-1)/3$. So I will unroll until I get $k = (n-1)/3$.

$T(n) = T(n-3) + 5n$

$T(n) = T(n-6) + 5(n-3) + 5n$

$T(n) = T(n-9)+5(n-6)+5(n-3)+5n \ldots T(n) = T(n-2((n-1)/3)+5*3+5*6+...+5(n-6)+5(n-3)+5n$

I can now see the pattern that is forming, and can move to finalize the $T(n)$ function

$T(n) = 4 + 5 * \sum_{i=0}^{(n-1)/3} n - 3i$

$T(n) = 4 + 5 * \sum_{i=0}^{(n-1)/3} n - 3 * 5 \sum_{i=0}^{(n-1)/3} i$

$T(n) = 4 + 5((n-1)/3) - 3 * 5(\frac{((n-1)/3)*((n-1)/3)+1)}{2})$

Now to find $f(n)$ such that $T(n) \in \Theta(f(n))$

To do so I will just take the highest order term within $T(n)$

That gives me $T(n) \in \Theta(n^2)$

□

3 Standard 18 - D&C counter examples

Problem 3. Consider the algorithm below which tries to find the smallest Euclidean distance between a pair of points among n points in x-y plane. Assume that input array P is sorted according to x-coordinate. You may break ties amongst two points with the same x-coordinate as you wish. Give an instance of input (preferably containing 8 or less points) for which it fails to output the closest pair and explain why it fails.

Algorithm 3 Closest pair

```
1: procedure CLOSEST(Array of points P[1, ..., n])
2:     n = P.length
3:     if n = 2 then return norm(P[1],P[2]) (where norm refers to 2-norm which is Euclidean distance)
4:     if n = 1 then return ∞
5:     if n = 0 then return ∞
6:     mid = ⌈n/2⌉
7:     closest_left = CLOSEST( P[1, ..., mid] )
8:     closest_right = CLOSEST( P[mid+1, ..., n] )
9:     return min{closest_left,closest_right}
```

Answer. An answer for which the algorithm gives the wrong answer is $P[(0,0),(4,4),(5,5),(9,9),(15,15)]$. The incorrect answer given by the algorithm is 2.8, when the correct answer is 1.4. The algorithm is incorrect because it doesn't compare all possible combinations of points. The way it splits the array in half stops it from comparing certain pairs, that could be the smallest distance. Whenever the closest pair is in the middle of the array, the answer will never be correct, because dividing the array stops them from being compared. For instance, in the example I've given the closest pair of points is (4,4) and (5,5) but they were never even compared because they were split into two different arrays before they could be compared. Therefore, the divide and conquer implementation in this algorithm fails to yield correct results for all input.

□