CSCI 3104 Fall 2021
Profs Josh Grochow and Bo Waggoner

Problem Set 6

Due Date ..... October / 11*th* / 2021
Name ..... **Michael Ghattas**
Student ID **109200649**
Collaborators ..... **Me, Myself & I**

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 agree to the above, Michael Ghattas. □

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:

We note the base case condition is $(n \leq 4)$**, and the recursion of** Foo **happens twice on** $(\frac{n}{3})$**.**

Thus the recurrence relation for the runtime complexity of these algorithms is:

$$T(n) = \begin{cases} \Theta(1) & : n \leq 4, \\ 2T(\frac{n}{3}) + \Theta(n) & : n > 4. \end{cases}$$

□

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:

We note the base case condition is $(n \leq 4)$**, and the recursion of** Foo **happens once on** $(\frac{n}{4})$ **and twice on** $(\frac{n}{3})$**.**

Thus the recurrence relation for the runtime complexity of these algorithms is:

$$T(n) = \begin{cases} \Theta(1) & : n \leq 4, \\ T(\frac{n}{4}) + 2T(\frac{n}{3}) + \Theta(n) & : n > 4. \end{cases}$$

□

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:

We first determine the number of times that we need to unroll. We hit a base case when $n - 2k < 2$. Solving for k, we obtain that:

$$\begin{aligned} (n-2k) &< 2 \\ n-2 &< 2k \\ \frac{n-2}{2} &< k \end{aligned}$$

So we have to unroll until $k = \left\lceil \frac{(n-2)}{2} \right\rceil$

When we unroll $T(n-2i)$, we obtain $2T(n-2(i+1))+1$. Note that when we unroll $2T(n-2(i+1))$, we obtain $2[2T(n-2(i+2)))+1]$. So:

$$\begin{aligned} T(n-2i) &= 2[2T(n-2(i+2)))+1]+1 \\ &= 2^2T(n-2(i+2))) + 2^1 * 1 + 2^0 * 1 \end{aligned}$$

We intentionally left the coefficients as 2^2, 2^1, and 2^0 respectively, to emphasize a pattern. Thus:

$$\begin{aligned} T(n) &= 2^k * T(n-2k) + 2^k - 1 \\ &= 2^{\left\lceil \frac{(n-2)}{2} \right\rceil} * T(n - 2\left\lceil \frac{(n-2)}{2} \right\rceil) + 2^{\left\lceil \frac{(n-2)}{2} \right\rceil} - 1 \\ &= 2^{\left\lceil \frac{(n-2)}{2} \right\rceil} * 3 + 2^{\left\lceil \frac{(n-2)}{2} \right\rceil} - 1 \\ &= 4 * 2^{\left\lceil \frac{(n-2)}{2} \right\rceil} \\ &= 4 * 2^{\left\lceil \frac{(n-2)}{2} \right\rceil} - 1 \end{aligned}$$

Taking the high-order term, we obtain that $T(n) \in \Theta(2^{\frac{n}{2}})$. □

(b)

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

Answer:

Based on the methodology we followed in part (a), we proceed.

We first start by unrolling $T(n)$:

$$\begin{aligned} T(n) &= T(n-3) + 5n \\ &= T(n-6) + 5n + 5(n-3) \\ &= T(n-9) + 5n + 5(n-3) + 5(n-6) \end{aligned}$$

item **Next we find k:**

$$\begin{aligned} n - 3k &< 3 \\ n - 3 &< 3k \\ \frac{n-3}{3} &< k \end{aligned}$$

Thus:

$$\begin{aligned} T(n) &= T(n-3k) + 5nk - 5 * (\frac{3k(k-1)}{2}) \\ &= T(n - 3 * \left\lceil \frac{n-3}{3} \right\rceil) + [5n * \left\lceil \frac{n-3}{3} \right\rceil] - [5 * \frac{3 * \lceil \frac{n-3}{3} \rceil * (\lceil \frac{n-3}{3} \rceil - 1)}{2}] \end{aligned}$$

Taking the high-order term, we obtain that $T(n) \in \Theta(n^2)$.

□

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:

We will be proceeding based on the provided algorithm for Closest pair.

An example of an input that fails : [(1,1) , (1,2) , (1,3)

The input fails because of the following:

- $n = P.length = 3$
- $mid = \lceil \frac{n}{2} \rceil = \lceil \frac{3}{2} \rceil = \lceil 1.5 \rceil = 2$
- closest left = CLOSEST(P[$1 \to mid$]) = CLOSEST(P[$1 \to 2$]) = CLOSEST(P[$(1,1),(1,2)$])
  CLOSEST(P[$(1,1),(1,2)$]) $\to n = 2$ and $mid = 1$
  closest left = CLOSEST(P[1]) = (1, 1)
- closest right = CLOSEST(P[$mid + 1 \to n$]) = CLOSEST(P[3]) = CLOSEST(P[$(1,3)$])
  CLOSEST(P[$(1,3)$]) $\to n = 1 \to$ returns ∞

Thus the function return $\to$ min{closest left,closest right} = min{(1, 1) , (∞)}. Clearly the wrong answer! □