CSC165: Problem Set 4

Tiago Ferreira, Pan Chen, Hyun Hak Shin

March 29, 2019

1. (a) The Theta bound is $n \cdot \log n$.

   We will show $\sum_{i=1}^{n} \lceil \frac{n}{i} \rceil \in \Theta(n \cdot \log n)$ using two parts. That is,

$$\underbrace{\sum_{i=1}^{n} \lceil \frac{n}{i} \rceil \in \mathcal{O}(n \cdot \log n)}_{Part1} \wedge \underbrace{\sum_{i=1}^{n} \lceil \frac{n}{i} \rceil \in \Omega(n \cdot \log n)}_{Part2}$$

Part 1: Proving $\sum_{i=1}^{n} \lceil \frac{n}{i} \rceil \in \mathcal{O}(n \cdot \log n)$.

That is, we want to show $\exists c, n_0 \in \mathbb{R}^+, \forall n \in \mathbb{N}, n \geq n_0 \Rightarrow \sum_{i=1}^{n} \lceil \frac{n}{i} \rceil \leq c \cdot (n \cdot \log n)$

(1) From (Fact 1), we know $\sum_{i=1}^{n} \frac{1}{i} \in \Theta(\log n)$. And by the definition of big Theta, we know that $\sum_{i=1}^{n} \frac{1}{i} \in \mathcal{O}(\log n) \wedge \sum_{i=1}^{n} \frac{1}{i} \in \Omega(\log n)$. So, $\sum_{i=1}^{n} \frac{1}{i} \in \mathcal{O}(\log n)$. That is, we can assume there exists $n_1, c_1 \in \mathbb{R}^+$ such that $\forall n \in \mathbb{N}, n \geq n_1 \Rightarrow \sum_{i=1}^{n} \frac{1}{i} \leq c_1 \cdot \log n$

(2) From (Fact 2), we know $\lceil \frac{n}{i} \rceil \leq \frac{n}{i} + 1$ for $i = 1, 2, \cdots, n$.

Let $c = c_1 + 1, n_0 = n_1 + 2$

Let $n \in \mathbb{N}$ and assume $n \geq n_0$. We want to show $\sum_{i=1}^{n} \lceil \frac{n}{i} \rceil \leq c \cdot (n \cdot \log n)$.

We can calculate,

$$\begin{aligned}
\sum_{i=1}^{n} \lceil \frac{n}{i} \rceil &\le \sum_{i=1}^{n} \frac{n}{i} + 1 && \text{(from (2), we know } \lceil \frac{n}{i} \rceil \le \frac{n}{i} + 1 \text{ for } i = 1, 2, \cdots, n\text{)} \\
&= \sum_{i=1}^{n} \frac{n}{i} + \sum_{i=1}^{n} 1 \\
&= n \cdot \sum_{i=1}^{n} \frac{1}{i} + \sum_{i=1}^{n} 1 \\
&\le n(c_1 \cdot \log n) + n && \text{(because } n \ge (n_0 = n_1 + 2) \text{ so } n \ge n_1\text{. And from (1) we know } \sum_{i=1}^{n} \frac{1}{i} \le c_1 \cdot \log n\text{)} \\
&= n(c_1 \cdot \log n + 1) \\
&= n(c_1 \cdot \log n + \log 2) \\
&\le n(c_1 \cdot \log n + \log n) && \text{(since } n \ge 2 \text{ so } \log n \ge \log 2\text{)} \\
&= n(\log n \cdot (c_1 + 1)) \\
&= c \cdot (n \cdot \log n) && \text{(Since } c = c_1 + 1\text{)}
\end{aligned}$$

This shows $\sum_{i=1}^{n} \lceil \frac{n}{i} \rceil \in \mathcal{O}(n \cdot \log n)$, as required.

<u>Part 2:</u> Proving $\sum_{i=1}^{n} \lceil \frac{n}{i} \rceil \in \Omega(n \cdot \log n)$

We want to show $\exists n_0, c \in \mathbb{R}^+, \forall n \in \mathbb{N}, n \ge n_0 \Rightarrow \sum_{i=1}^{n} \lceil \frac{n}{i} \rceil \ge c \cdot (n \cdot \log n)$

(1) From <u>(Fact 1)</u>, we know $\sum_{i=1}^{n} \frac{1}{i} \in \Theta(\log n)$. And by the definition of big Theta, we know that $\sum_{i=1}^{n} \frac{1}{i} \in \mathcal{O}(\log n) \wedge \sum_{i=1}^{n} \frac{1}{i} \in \Omega(\log n)$. So, $\sum_{i=1}^{n} \frac{1}{i} \in \Omega(\log n)$. That is, we can assume $\exists n_1, c_1 \in \mathbb{R}^+$, such that $\forall n \in \mathbb{N}, n \ge n_1 \Rightarrow \sum_{i=1}^{n} \frac{1}{i} \ge c_1 \cdot \log n$

(2) From <u>(Fact 2)</u>, we know $\lceil \frac{n}{i} \rceil \ge \frac{n}{i}$ for $i = 1, 2, \cdots, n$.

Let $c = c_1, n_0 = n_1$

Let $n \in \mathbb{N}$ and assume $n \ge n_0$. W.T.S $\sum_{i=1}^{n} \lceil \frac{n}{i} \rceil \ge c \cdot (n \cdot \log n)$.

We can calculate,

$$\begin{aligned}
\sum_{i=1}^{n} \lceil \frac{n}{i} \rceil &\ge \sum_{i=1}^{n} \frac{n}{i} && \text{(from (2), we know } \lceil \frac{n}{i} \rceil \ge \frac{n}{i} \text{ for } i = 1, 2, \cdots, n\text{)} \\
&= n \cdot \sum_{i=1}^{n} \frac{1}{i} \\
&\ge n(c_1 \cdot \log n) && \text{(because } n \ge (n_0 = n_1) \text{ so from (1) we know } \sum_{i=1}^{n} \frac{1}{i} \ge c_1 \cdot \log n\text{)} \\
&= c(n \cdot \log n) \text{(Since } c = c_1\text{)}
\end{aligned}$$

This shows that $\sum_{i=1}^{n} \lceil \frac{n}{i} \rceil \in \Omega(n \cdot \log n)$, as required.

In conclusion, we see that $\sum_{i=1}^{n} \lceil \frac{n}{i} \rceil \in \mathcal{O}(n \cdot \log n)$ and $\sum_{i=1}^{n} \lceil \frac{n}{i} \rceil \in \Omega(n \cdot \log n)$. So by the definition of big Theta, we know that $\sum_{i=1}^{n} \lceil \frac{n}{i} \rceil \in \Theta(n \cdot \log n)$. This is exactly what we wanted to prove.

(b) We will analyze the running time on print_multiples to determine a theta bound on the running time.

First let n be an arbitrary integer, note that in Loop 1, d iterates from 1 to n for a total of n iterations. Under a fixed loop 1 iteration, for loop 2, we know that range(0, n, d) consists of the multiples of d such that $n > d \geq 0$, where $d \in 0, d, \cdots, d(\lceil \frac{n}{d} \rceil - 1)$. We know this from the Python Note given to us. We also know the loop iterator only changes its value after one iteration in this algorithm. Therefore we conclude that there are $\lceil \frac{n}{d} \rceil$ iterations of loop 2 (which depends on the value of d, hence depends on how many loop 1 iterations have been executed), where each iteration takes constant time.

Therefore the running time takes a total of: $\sum_{d=1}^{n} \lceil \frac{n}{d} \rceil$ steps. Using what we know from part (a), we can determine from the number of steps that the running time is $\in \Theta(n \cdot log(n))$.

(c) First, let n be an arbitrary integer.

From b) we know the running time of loop 1 and 2 is

$$\sum_{d=1}^{n} \lceil \frac{n}{d} \rceil \in \Theta(n \cdot log(n))$$

For this question the only difference in the code itself in comparison to parts a) and b) is the addition of the if block. The if block executes when d is a multiple of 5. Also notice that d starts at 1. Therefore we can conclude that the if statement (line 7) will execute $\lfloor \frac{n}{5} \rfloor$ times throughout the running of the program. Since loop 3 iterates d times and d is always a multiple of 5 when the if block is entered, as well as coupling the analysis we did in previous parts, we conclude the total amount of steps will be:

$$\begin{aligned}
\sum_{d=1}^{n} \lceil \frac{n}{d} \rceil + \sum_{d=1}^{\lfloor n/5 \rfloor} 5 \cdot d &= \sum_{d=1}^{n} \lceil \frac{n}{d} \rceil + 5 \sum_{d=1}^{\lfloor n/5 \rfloor} d && \text{(By summation laws)} \\
&= \underbrace{\sum_{d=1}^{n} \lceil \frac{n}{d} \rceil}_{\substack{\in\Theta(n \cdot \log n) \\ \text{from b)}}} + \underbrace{5 \cdot \left[\frac{\lfloor \frac{n}{5} \rfloor (\lfloor \frac{n}{5} \rfloor + 1)}{2} \right]}_{\in\Theta(n^2)} \in \Theta(n^2) && (Since \sum_{i=1}^{n} i = \frac{n(n+1)}{2})
\end{aligned}$$

Notice that we conclude the above given the fact that loop 2 and loop 3 take one basic step on each iteration. Therefore we can conclude that the running time of print_multiples2 is $\in \Theta(n^2)$.

2. (a) Let $n \in \mathbb{N}$ and let lst be a list of length n where every element up until the last element is the integer 2 and the last element is the integer 1. We assume that $n \geq 2$ by the definition of a theta

bound.

First notice that lines 2 - 4 count as one basic step. Then the program will enter the while loop and because the first n-1 elements are all 2 which satisfies $lst[i]$ % 2 == 0 for $i = 1, 2, \cdots, n-1$, the if block will run for n - 2 iterations where each iterations takes one basic step (lines 7 - 8) for a total of (n - 2) steps (Notice that the first element does not matter because i starts at 1). After these iterations, we have:

$$i = n - 1$$

$$j = 2^{n-2}$$

This is because in the first n -2 iterations, only the if block will run and each time the if block is executed, i will increase by 1 and j will increase by a factor of 2.

Since $i = n - 1$ and we know $n - 1 < n$, we notice that the while loop will continue to iterate after this because $i < n$ is still true.

Now, because we know that the last item of list is 1, we notice that the program will enter the else block for one last iteration. After this iteration, The value of i will be incremented to 2n - 2 and Because $n \geq 2$, so $2n \geq 2 + n \Rightarrow 2n - 2 \geq n$. Thus, the while loop will break after this iteration. However, notice that the inner for loop in the else block will iterate $j = 2^{n-2}$ times with k going from 0 to j - 1 at constant time in each iteration for a total of 2^{n-2} steps. Therefore there will be a total of $1 + 1(n-2) + 2^{n-2}$ steps which is $\in \Theta(2^n)$ as required.

(b) The input family we find is: $\forall n \in N$, let lst be a list of length n which satisfies:

1. the first $\lceil\sqrt{n}\rceil$ items are 2.
2. the rest of the items, that is the last $n - \lceil\sqrt{n}\rceil$ items, are 1.

Note that we take $n \geq 4$ by the definition of a theta bound.

We first notice that lines(2-4) will count as 1 basic step. So the runtime before the algorithm enters Loop 1 is 1. The program will then enter the while loop (Loop 1). We notice that in lst, the first $\lceil\sqrt{n}\rceil$ items are 2 so $lst[i]$ % 2 == 0 will hold true for $i = 0, 1, \cdots, \lceil\sqrt{n}\rceil - 1$. Therefore the if block will be executed $\lceil\sqrt{n}\rceil$ times in the first $\lceil\sqrt{n}\rceil$ Loop 1 iterations. Since in the if branch there is on basic step (lines 7 - 8), we conclude the runtime of the first $\lceil\sqrt{n}\rceil$ Loop 1 iterations is $\lceil\sqrt{n}\rceil \times 1$, which is $\lceil\sqrt{n}\rceil$. Note that after $\lceil\sqrt{n}\rceil$ Loop 1 iterations, i will increase from 1 to 1 + $\lceil\sqrt{n}\rceil$ and j from 1 to $2^{\lceil\sqrt{n}\rceil}$ due to the fact that i increases by 1 and j increases by a factor of 2 in each if block execution. Therefore we have:

$$i = 1 + \lceil \sqrt{n} \rceil$$

$$j = 2^{\lceil \sqrt{n} \rceil}$$

And since we have that $n \geq 4$, we can calculate:

$$\begin{aligned} n &= (\sqrt{n})^2 \\ &= \sqrt{n} \cdot \sqrt{n} \\ &\geq \sqrt{n} \cdot \sqrt{4} \text{(since n is no less than 4)} \\ &= 2 \cdot \sqrt{n} \\ &= \sqrt{n} + \sqrt{n} \\ &\geq \sqrt{n} + \sqrt{4} \text{(since n is no less than 4)} \\ &= \sqrt{n} + 2 \\ &= \sqrt{n} + 1 + 1 \\ &> \lceil \sqrt{n} \rceil + 1 \textit{(since by definition of ceiling, we know} \lceil \sqrt{n} \rceil < \sqrt{n} + 1) \end{aligned}$$

So we get $i = (1 + \lceil \sqrt{n} \rceil) < n$ after the first $\lceil \sqrt{n} \rceil$ Loop 1 iterations. Therefore the while loop condition $(i < n)$ will hold true, so the program will enter the while loop again.

Then we see that $lst[i]$ % 2 == 0 will be false for $i = \lceil \sqrt{n} \rceil, \lceil \sqrt{n} \rceil + 1, \cdots, n-1$. So, after the first $\lceil \sqrt{n} \rceil$ Loop 1 iterations, only the else block will be executed. Now we want to find how many iterations the else block will run for. We have that $i = 1 + \lceil \sqrt{n} \rceil$ from our previous induction, and we know that i will increase by a factor of two after every else block execution. We also notice that Loop 1 will exit once $i \geq n$. Therefore we want to find the smallest integer m such that $(1 + \sqrt{n}) \cdot 2^m \geq n$ holds true. We then calculate:

$$\begin{aligned} (1 + \sqrt{n}) \cdot 2^m &\geq n \\ log((1 + \sqrt{n}) \cdot 2^m) &\geq log(n) \\ log(1 + \sqrt{n}) + log(2^m) &\geq log(n) \\ log(1 + \sqrt{n}) + m \cdot log(2) &\geq log(n) \\ m \cdot log(2) &\geq log(n) - log(1 + \sqrt{n}) \\ m &\geq \frac{log(n) - log(1 + \sqrt{n})}{log(2)} \; (log(2) = 1, \; \textit{so dividing by } log(2) \textit{ won't change the inequality.}) \end{aligned}$$

Thus we take $m = \lceil \frac{log(n)-log(1+\sqrt{n})}{log(2)} \rceil$, which signals how many else branch executions will be needed to make Loop 1 terminate. So, the else branch of Loop 1 runs $\lceil \frac{log(n)-log(1+\sqrt{n})}{log2} \rceil$ times, which is $\in \Theta(log(n))$. This conclusion comes from ignoring the constant $\frac{1}{log(2)}$ and the lower order terms. **This is one of the statements we wanted to prove**.

Now back to the total time of the program; Since we know it runs $\lceil \sqrt{n} \rceil + \lceil \frac{log(n)-log(1+\sqrt{n})}{log2} \rceil$ iterations, with the first $\lceil \sqrt{n} \rceil$ iterations running the if branch which takes one single basic step and the last $\lceil \frac{log(n)-log(1+\sqrt{n})}{log2} \rceil$ iterations running the else branch, which takes $2^{\lceil \sqrt{n} \rceil}$ steps. We have shown that after the first $\lceil \sqrt{n} \rceil$ iterations, j becomes $2^{\lceil \sqrt{n} \rceil}$. Then by our previous analysis we know only the else branch will be executed and in the else branch j remains unchanged and it runs a for loop, which takes $j = 2^{\lceil \sqrt{n} \rceil}$ steps. Therefore, the runtime of last $\lceil \frac{log(n)-log(1+\sqrt{n})}{log2} \rceil$ Loop1 iterations is $\lceil \frac{log(n)-log(1+\sqrt{n})}{log2} \rceil \times 2^{\lceil \sqrt{n} \rceil}$.

We have shown that so the runtime before the algorithm enters Loop1 is 1 and the runtime of the first $\lceil \sqrt{n} \rceil$ Loop 1 iterations is $\lceil \sqrt{n} \rceil$. So, the total runtime is $1 + \lceil \sqrt{n} \rceil + \lceil \frac{log(n)-log(1+\sqrt{n})}{log2} \rceil \times 2^{\lceil \sqrt{n} \rceil}$, which is $\Theta(log(n) \cdot 2^{\sqrt{n}})$.

(c) We want to show that the worst case running time of alg is $\in \mathcal{O}(2^n)$.

First, let $n \in \mathbb{N}$, let lst be an arbitrary list of integers of length n.

Since Loop 1 ends when $i \geq n$ and in each Loop 1 iteration, the if branch will make i increase by 1, and the else branch will make i increase by a factor of 2, thus i increases by **at lease 1** after one Loop 1 iteration, so there are **at most n - 1** Loop 1 iterations (since i starts from 1). We claim for the fixed k^{th} iteration that it takes at most 2^{k-1} steps. This claim will be shown in this analysis.

We have shown that there will be at most n - 1 iterations of the while loop (Loop 1). We then see that the if block will take 1 basic step in each iteration, which takes constant time. We also note that the if branch will make j increase by a factor of 2 and that the runtime of the else branch only depends on the value of j. This helps us to notice that in the k^{th} iteration, j can be at most 2^{k-1}, which happens only if the if branch is executed $k-1$ times in the first $k-1$ iterations of the while loop. Therefore the else block will take at most 2^{k-1} steps in the k^{th} Loop 1 iteration. This proved our claim.

Therefore there is at most 2^{k-1} steps for the k^{th} iteration of the while loop for at most n - 1 iterations. Therefore there will be at most $\sum_{k=1}^{n} 2^{k-1} = 2^0 + \cdots 2^{n-1}$ steps, where this expression represents the sum of the most amount of steps at each iteration, for the most iterations. By ignoring the low order items, we conclude it is $\in \mathcal{O}(2^n)$

3. (a) (i)

$$\begin{aligned} &\forall n \in \mathbb{N}, BC_{func}(n) \leq f(n) \\ \Leftrightarrow &\forall n \in \mathbb{N}, min\{\text{running time of executing func(x)}|x \in \mathcal{I}_{func,n}\} \leq f(n) \\ \Leftrightarrow &\forall n \in \mathbb{N}, \exists x \in \mathcal{I}_{func,n}, \text{runtime of executing func(x)} \leq f(n) \end{aligned}$$

(ii)

$$\begin{aligned} &\forall n \in \mathbb{N}, BC_{func}(n) \geq f(n) \\ \Leftrightarrow &\forall n \in \mathbb{N}, min\{\text{running time of executing func(x)}|x \in \mathcal{I}_{func,n}\} \geq f(n) \\ \Leftrightarrow &\forall n \in \mathbb{N}, \forall x \in \mathcal{I}_{func,n}, \text{runtime of executing func(x)} \geq f(n) \end{aligned}$$

(b) We will prove $BC(n) \in \Theta(n)$, that is, $BC(n) \in \mathcal{O}(n) \wedge BC(n) \in \Omega(n)$.

Part 1, Prove $BC(n) \in \mathcal{O}(n)$: We need to find an input family whose runtime is $\mathcal{O}(n)$.

Let $n \in \mathbb{N}$. And let lst be a list of length n where every element of lst is the integer 0. Then we notice that since there is no early return, Loop 1 iterates for exactly $n - 2$ iterations, with i going from 2 to n - 1(increase by 1 after each Loop 1 iteration).

Since i goes from 2 to n - 1, i will be even on the first iteration and odd on the second and will keep alternating between even and odd until it reaches n - 1. Therefore it will enter the if block when it is even and the else block when it is odd. We notice that since all elements in our input list are 0, so $lst[j + 2] = lst[j]$ is always true, which will make both loop 2 and loop 3's condition false every time the if or else block is entered. So neither loop 2 or loop 3 will ever be executed. We thus take lines four and five as one basic step when the if block is entered and lines nine and ten as one basic step when the else block is entered. Therefore a fixed Loop 1 iteration, whether the if branch executes or the else branch executes, will take one basic step.

Then since we know there are n - 2 iterations of the for Loop 1, where each iteration is one basic step(as we have shown above) for a total step count of n - 2. Therefore, the runtime is n - 2, which is O(n). And by the definition of upper bound on best case, we know that : $BC(n) \in \mathcal{O}(n)$.

Part 2, Prove $BC(n) \in \Omega(n)$: First, let $n \in \mathbb{N}$, let lst be an arbitrary list of integers of length n.

Note that loop 1 will iterate for exactly n - 2 iterations (as we have shown in Part 1). Then notice that:

If it enters the if block then the least amount of runtime the inner while loop (loop 2) body will run is 0 (with line 5 executes only once). We take lines 4 and 5 to count as 1 basic step. So there will be at least one basic step in the if block.

If it enters the else block then the least amount of runtime the inner while loop (loop 3) body will run is 0 (with line 10 executes only once). We take lines 9 and 10 to count as 1 basic step. So there will be at least one basic step in the else block.

Therefore a fixed Loop 1 iteration, whether the if branch executes or the else branch executes, will take one basic step.

And because there is a total of $n - 2$ iterations with at least 1 step for each iteration, the total runtime is at least: $n - 2$, which is $\Omega(n)$. And by the definition of upper bound on best case, we know that : $BC(n) \in \Omega(n)$.

Therefore since $BC(n) \in \Omega(n) \wedge BC(n) \in \mathcal{O}(n)$, by definition of Theta we can conclude that $BC(n) \in \Theta(n)$ as required.