CSC 165 Problem Set 4

Eric Zhu, Emily Mazor, Kristin Huang

March 28, 2019

1. Printing Multiples

(a)

Proof of the upper bound (Big-oh):

We will prove that $\sum_{i=1}^{n} \lceil \frac{n}{i} \rceil \in \Theta(n \log n)$

Using Fact 2, substituting $x = \frac{n}{i}$, we know, $\lceil \frac{n}{i} \rceil < \frac{n}{i} + 1$.
Hence:

$$\begin{aligned}\sum_{i=1}^{n} \lceil \frac{n}{i} \rceil &< \sum_{i=1}^{n} (\frac{n}{i} + 1) \\ &= \sum_{i=1}^{n} \frac{n}{i} + \sum_{i=1}^{n} 1 \\ &= n \sum_{i=1}^{n} \frac{1}{i} + n\end{aligned}$$

By fact 1, we know: $\sum_{i=1}^{n} \frac{1}{i} \in \Theta(\log n)$, so $\sum_{i=1}^{n} \lceil \frac{n}{i} \rceil < n \log n + n$.

We can conclude: $\sum_{i=1}^{n} \lceil \frac{n}{i} \rceil \in \mathcal{O}(n \log n)$

Proof of the lower bound (Big-omega):

We will prove that $\sum_{i=1}^{n} \lceil \frac{n}{i} \rceil \in \Omega(n \log n)$.

By fact 2, substituting $x = \frac{n}{i}$, we know, $\frac{n}{i} \leq \lceil \frac{n}{i} \rceil$.

Hence,

$$\begin{aligned}\sum_{i=1}^{n}\lceil\frac{n}{i}\rceil &\geq \sum_{i=1}^{n}\frac{n}{i}\\ &= n\sum_{i=1}^{n}\frac{1}{i}\end{aligned}$$

Additionally, by fact 1, we know $\sum_{i=1}^{n}\frac{1}{i} \in \Theta(\log n)$.

Consequently, $\sum_{i=1}^{n}\lceil\frac{n}{i}\rceil \geq n\log n$ and $\sum_{i=1}^{n}\lceil\frac{n}{i}\rceil \in \Omega(n\log n)$

Since we have that $\sum_{i=1}^{n}\lceil\frac{n}{i}\rceil \in \Omega(n\log n)$ and $\sum_{i=1}^{n}\lceil\frac{n}{i}\rceil \in \mathcal{O}(n\log n)$, it must be that $\sum_{i=1}^{n}\lceil\frac{n}{i}\rceil \in \Theta(n\log n)$.

(b)
The theta bound of `print_multiples` is $\Theta(n\log n)$.
This is because we can write out the series of the runtime of each iteration of loop 1 as $\lceil n\rceil + \lceil\frac{n}{2}\rceil + \lceil\frac{n}{3}\rceil + ... + \lceil\frac{n}{n-1}\rceil + \lceil\frac{n}{n}\rceil$. We deduce this from examining the runtime of loop 2, which is dependent on d, specifically, `multiple` goes up by d each iteration. For example, when $d = 1$ we have $\lceil n\rceil$, when d = 2 we have $\lceil\frac{n}{2}\rceil$, when $d = 3$ we have $\lceil\frac{n}{3}\rceil$, and so on. We can now recognize that series is exactly represented by $\sum_{i=1}^{n}\lceil\frac{n}{i}\rceil$. As we've proven in part a, $\sum_{i=1}^{n}\lceil\frac{n}{i}\rceil \in \Theta(n\log n)$.

(c)
For a fixed iteration of Loop 1, Loop 3 runs for d iterations, from $i = 0$ to $i = d - 1$. Since each iteration of Loop 3 takes a single step, its total running time is d. Additionally, Loop 3 only runs if d is divisible by 5, and d goes from 0 to $n - 1$, and d increases by 1 for each iteration of Loop 1, so $d\%5 == 0$ evaluates to True a total of $\lfloor\frac{n}{5}\rfloor$ times.

For every iteration of Loop 3, it has a total running time of d. Since d changes from 1 to n, and Loop 3 only runs when $d\%5 == 0$, we know that d must be a multiple of 5. We also know d increments by 5 until the if condition is satisfied $\lfloor\frac{n}{5}\rfloor$ times. Hence we can write a summation for the total number of times Loop 3 iterates.

Since $d\%5 == 0$ evaluates to True a total of $\lfloor\frac{n}{5}\rfloor$ times, Loop 3 will iterate $\sum_{i=1}^{\lfloor\frac{n}{5}\rfloor} 5$

times, so line 9 will run $\sum_{i=1}^{\lfloor\frac{n}{5}\rfloor} 5$ times. Rewriting the summation we have:

$$\begin{aligned}\sum_{i=1}^{\lfloor\frac{n}{5}\rfloor} 5 &= 5\sum_{i=1}^{\lfloor\frac{n}{5}\rfloor} 1\\ &= 5(\frac{\lfloor\frac{n}{5}\rfloor(\lfloor\frac{n}{5}\rfloor+1)}{2})\\ &= 5(\frac{(\lfloor\frac{n}{5}\rfloor)^2+\lfloor\frac{n}{5}\rfloor}{2})\\ &= \frac{5}{2}((\lfloor\frac{n}{5}\rfloor)^2+\lfloor\frac{n}{5}\rfloor)\end{aligned}$$

We also know from part b that line 5 will run in $\Theta(n\log n)$. Since $n\log n$ is a lower order term than $\frac{5}{2}((\lfloor\frac{n}{5}\rfloor)^2+\lfloor\frac{n}{5}\rfloor)$, we can conclude that a theta bound on `print_multiples2` is $\Theta(n^2)$

2. Varying running times, input families, and worst case analysis

(a)
To see that the running time for alg is $\Theta(2^n)$, let $n \in \mathbb{N}$ and consider the input family, lst, a List of length n where every element of lst from index 0 to index $n-2$ (inclusive) is the number 2, and the element at $n-1$ is the number 1.

Hence the input family would look like this, with lst having a length of n:

lst = [2, 2, 2, ... 1]

For the first $n-2$ iterations of Loop 1, the if branch executes, and each time the if branch executes, i increases by 1 and j doubles in value. After $n-2$ iterations, $i < n$ since $i = 1 + n - 2 = n - 1$.

As j doubles in value each time the if branch executes, by the end of $n-2$ iteration of Loop 1, we have that $j = 2^{n-2}$.

After $n-2$ iterations, $i = n-1 < n$, so the while loop still runs on iteration $n-1$. Now $i = n-1$ so $\text{lst}[i] = \text{lst}[n-1] = 1$. Since 1 is divisible by 1, the else branch of the while loop runs. In the else branch, i doubles to become $2(n-1)$, and Loop 2 runs. Loop 2 runs j times, from $k = 0$ to $k = j-1$, where each iteration takes constant time. Since $j = 2^{n-2}$ on the $n-1$'s iteration of Loop 1, Loop 2 runs 2^{n-2} times on the $n-1$'s iteration of Loop 1, with each iteration taking a single step, so the cost for this iteration is 2^{n-2}.

After the else branch is executed, i becomes $2(n-1)$. Given $i = 2n-2$, $i < n$ is only true if $n < 2$, but since i starts at 1 and the while loop only ran when $i < n$, where n is an integer (length of the lst), it is impossible for $n < 2$. Hence $i < n$ is False and the while loop stops.

So in total, the Loop 1 first iterated $n-2$ times executing the if branch with constant cost, so it cost $n-2$ for the first $n-2$ iterations. Then it iterated once, executing the else branch with a cost of 2^{n-2}. Adding the two together, we get $(n-2) + 2^{n-2}$, which is $\Theta(2^n)$.

(b)
To see that the running time for alg is $\Theta(log(n) \cdot 2^{\sqrt{n}})$, let $n \in \mathbb{N}$ and consider the input family, lst, a List of length n that consists of all 2's from index 0 to index $\lfloor\sqrt{n}\rfloor - 2$ (inclusive) and all 1's from index $\lfloor\sqrt{n}\rfloor - 1$ to $n-1$ (inclusive).

Hence, for the first iteration to the $\lfloor\sqrt{n}\rfloor - 1$'s iteration, i moves from 1 to $\lfloor\sqrt{n}\rfloor - 2$ so $lst[i] = 2$, and since 2 is divisible by 2, the if branch will always run for these iterations. After $\lfloor\sqrt{n}\rfloor - 1$ iterations, $i = 1 + \lfloor\sqrt{n}\rfloor - 1 = \lfloor\sqrt{n}\rfloor$, and j = $2^{\lfloor\sqrt{n}\rfloor - 1}$.

Now we move on to the index $\lfloor\sqrt{n}\rfloor$. Since starting from this index, $lst[i] = 1$, and since 1 is not divisible by 2, the else branch will always execute. Now we try to find how many times the else branch will execute, making k = maximum number of times the else branch executes.

We know the while loop stops when $i \geq n$, and after k-iterations, i multiplies by 2^k, so i will become $\lfloor\sqrt{n}\rfloor \cdot 2^k$. Hence we want to find k where $\lfloor\sqrt{n}\rfloor \cdot 2^k \geq n$.

From the definition of the floor, we know $\sqrt{n} \geq \lfloor\sqrt{n}\rfloor$, so we want to find k where $\sqrt{n} \cdot 2^k \geq n$.

$$\begin{aligned} \text{so, we have: } \sqrt{n} \cdot 2^k &\geq n \\ &\iff \\ 2^k &\geq \sqrt{n} \\ &\iff \\ log_2(\sqrt{n}) &\leq k \end{aligned}$$

So Loop 1 terminates when $k = \lceil log_2(\sqrt{n}) \rceil$.
Hence, we know Loop 1 run $k = \lceil log_2(\sqrt{n}) \rceil$ times, with each iteration taking constant time. Loop 2 has j iterations and constant time steps, so it has a cost of j. From the executions of the if branch, we know $j = 2^{\lfloor \sqrt{n} \rfloor - 1}$, which leads us to the runtime of the else branch: $2^{\lfloor \sqrt{n} \rfloor - 1} \cdot \log(n)$. The if branch also executes $\lfloor \sqrt{n} \rfloor - 1$ times. This concludes a total runtime of $2^{\lfloor \sqrt{n} \rfloor - 1} \cdot \log(n) + \lfloor \sqrt{n} \rfloor - 1$. Since $\sqrt{n}$ is a lower order term than $2^{\sqrt{n}} \cdot \log(n)$ we have $\Theta(2^{\sqrt{n}} \cdot \log(n))$ overall.

(c) Proof.
The initialization lines before the while loop take one step, which is constant time. At the end of the while loop two groups of events would've occurred. Either i and j increase by 1 and by a factor of 2 respectively, or i increases by a factor of 2 and loop 2 is run, costing a constant amount of operations per iteration for j iterations. The consequence of this relationship is that j can be considered to be dependent on i, i.e, we can express j as $j = 2^i$ after $i = k$ iterations where $k \in \mathbb{N}$. We also know that the smallest change in i is 1, so that at worst, loop 1 will run n times. Also, in each iteration, at worst 2^k iterations will run. Because we assume n iterations of loop 1, we can write this as:

$$\sum_{i=0}^{n-1} 2^i = 2^{n+1} - 1$$

Alternatively, we have:

$$2^{n+1} - 1 = 2^n \cdot 2 - 1$$

Since $2^n \cdot 2 - 1 \in \mathcal{O}(2^n)$, we have shown what was required. ■

3. Rearrangements, best-case analysis

(a)

(i)

$\forall n \in \mathbb{N}, BC_{func}(n) \leq f(n)$

$\iff \forall n \in \mathbb{N}, min\{$running time of executing $f(x)|x \in I_n\}) \leq f(n)$

$\iff \forall n \in \mathbb{N}, \exists x \in I_n,$ running time of executing $f(x) \leq f(n)$

(ii)

$\forall n \in \mathbb{N}, BC_{func}(n) \geq f(n)$

$\iff \forall n \in \mathbb{N}, min\{$running time of executing $f(x)|x \in I_n\}) \geq f(n)$

$\iff \forall n \in \mathbb{N}, \forall x \in I_n,$ running time of executing $f(x) \geq f(n)$

(b)

Lower Bound

Let n be an arbitrary natural number and let $len(lst) = n$. In the lower bound of the best case, the stopping condition of Loop 2 and 3 is always True so Loops 2 and 3 iterate 0 times. Since Loop 2 is in the if-branch and Loop 3 is in the else-branch, either the if branch or the else branch will execute for each iteration of the for loop. For a fixed iteration of Loop 1, either line 5 or line 10 will run, and since in the best case the stopping condition of Loops 2 and 3 is True for every iteration, Loops 2 and 3 iterate 0 times. Since Loop 1 iterates $n-2$ times, from $i = 2$ to $i = n-1$, and since Loops 2 and 3 iterate 0 times for a fixed iteration of Loop 1, the algorithm runs for $n-2$ iterations. Since each iteration takes a single step, we have a lower bound running time of $n-2$, which is $\Omega(n)$.

Upper Bound

Let lst be an input family where lst is a List of length n and every element of lst is the number 1. Then, it is always true that lst$[j+2]$ = lst$[j]$ since lst$[j+2] = 1$ and lst$[j]$ = 1 for all j. Since the stopping condition of Loop 2 and Loop 3 is $j < 0$ or lst$[j+2] \geq$ lst$[j]$, the stopping condition is always True, so Loop 2 and Loop 3 never execute. Loop 1 runs $n-3$ times since i goes from $i = 2$ to $i = n-1$ regardless of the input list, and so Loop 1 runs for $n-2$ iterations. Since Loops 2 and 3 iterate 0 times for a fixed iteration of Loop 1, and since each iteration of Loop 1 takes a single step, we have an upper bound running time of $n-2$, which is $\mathcal{O}(n)$.

Since the runtime is $\Omega(n)$ and $\mathcal{O}(n)$, we can conclude that the runtime is $\Theta(n)$.