CSC165H1 Problem Set 4

Xiaoyu Zhou, Yichen Xu

March 29, 2019

1. Printing multiples

   (a) *Proof.*

   From Fact 2, we can know that $\frac{n}{i} \leq \lceil \frac{n}{i} \rceil \leq \frac{n}{i} + 1$. So, $\sum\limits_{i=1}^{n} \frac{n}{i} \leq \sum\limits_{i=1}^{n} \lceil \frac{n}{i} \rceil \leq \sum\limits_{i=1}^{n} (\frac{n}{i} + 1)$.
   From Fact 1. we can know that $\exists c_1, c_2, n_0 \in \mathbb{R}^+$, $n \in \mathbb{N}$, $n \geq n_0 \Rightarrow c_1 \log \mathrm{n} \leq \sum\limits_{i=1}^{n} \frac{1}{i} \leq c_2 \log\ \mathrm{n}$.
   We want to prove that $\sum\limits_{i=1}^{n} \lceil \frac{n}{i} \rceil \in \Theta(n * \log \mathrm{n})$, which is $\exists a, b, n_1 \in \mathbb{R}^+$, $n \in \mathbb{N}$, $n \geq n_1 \Rightarrow a(n \log n) \leq \sum\limits_{i=1}^{n} \lceil \frac{n}{i} \rceil \leq b(n \log n)$.
   Let a = c_1, b = $c_2 + 1$, n_1 = max(n_0, 10). Let $n \in \mathbb{N}$. Assume $n \geq n_1$. We want to prove that $a(n \log n) \leq \sum\limits_{i=1}^{n} \lceil \frac{n}{i} \rceil \leq b(n \log n)$. Let's divide the proof in two parts.
   Part 1: Proof $a(n * \log \mathrm{n}) \leq \sum\limits_{i=1}^{n} \lceil \frac{n}{i} \rceil$.

$$\begin{aligned} \sum_{i=1}^{n} \left\lceil \frac{n}{i} \right\rceil &\geq \sum_{i=1}^{n} \frac{n}{i} \\ &= n * \sum_{i=1}^{n} \frac{1}{i} \\ &\geq (c_1 \log n) * n \\ &= c_1(n * \log n) \\ &= a(n * \log n) \end{aligned} \tag{1}$$

   Part 2: Proof $\sum\limits_{i=1}^{n} \lceil \frac{n}{i} \rceil \leq b(n * \log n)$

$$
\begin{aligned}
\sum_{i=1}^{n}\left\lceil\frac{n}{i}\right\rceil &\le \sum_{i=1}^{n}(\frac{n}{i}+1)\\
&= \sum_{i=1}^{n}\frac{n}{i}+\sum_{i=1}^{n}1\\
&= n\sum_{i=1}^{n}\frac{1}{i}+n\\
&\le n(c_2\log n)+n\\
&\le n(c_2\log n)+n\log n\,(since\,n=max(n_0,10), n\ge 10, so\ \log n\ge 1)\\
&= (c_2+1)(n\log n)\\
&= b(n\log n)
\end{aligned}
\tag{2}
$$

So, $a(n\log n)\le \sum\limits_{i=1}^{n}\left\lceil\frac{n}{i}\right\rceil \le b(n\log n)$.

So, $\sum\limits_{i=1}^{n}\left\lceil\frac{n}{i}\right\rceil \in\Theta(n\log n)$

(b) For loop 2, there are $\left\lceil\frac{n}{d}\right\rceil$ iterations, and every iterations takes constant time (since the runtime doesn't depend on the input size).
So, the runtime of loop 2 is $\left\lceil\frac{n}{d}\right\rceil$.
For loop 1, there are n iterations, and every ieration takes $\left\lceil\frac{n}{d}\right\rceil$ time.
So the runtime of loop 1 is $\sum\limits_{d=1}^{n}\left\lceil\frac{n}{d}\right\rceil$.
From part (a), we can know that $\sum\limits_{i=1}^{n}\left\lceil\frac{n}{i}\right\rceil \in\Theta(n\log n)$. So the runtime of loop 1 is $\Theta(n\log n)$.
Since there are no extra steps besides loop 1 in **print-multiples**, the total runtime of **print-multiples** is $\Theta(n\log n)$

(c) For loop 2, from the part(b), we can know its runtime is $\left\lceil\frac{n}{d}\right\rceil$.
For loop 3, there are d iterations and every iterations takes constant time (since the runtime doesn't depend on the input size). So the runtime of loop 3 is d.
For loop 1, we know the if statement will execute only when d is a multiple of 5. So, when $d\in\{1*5, 2*5, ..., \left\lfloor\frac{n}{5}\right\rfloor *5\}$, which is $d=5i$, where $i\in\{1,2,...,\left\lfloor\frac{n}{5}\right\rfloor\}$, the if statement will execute. So the total runtime of loop 1 is the sum of loop 2 and loop 3, which is $\sum\limits_{i=1}^{\left\lfloor\frac{n}{5}\right\rfloor}5i+\sum\limits_{d=1}^{n}\left\lceil\frac{n}{d}\right\rceil$.
Since there are no extra steps besides loop 1 in **print-multiples2**, the total runtime of **print-multiples2** is $\sum\limits_{i=1}^{\left\lfloor\frac{n}{5}\right\rfloor}5i+\sum\limits_{d=1}^{n}\left\lceil\frac{n}{d}\right\rceil$.

$$
\begin{aligned}
\sum_{i=1}^{\lfloor \frac{n}{5} \rfloor} 5i &= 5 \sum_{i=1}^{\lfloor \frac{n}{5} \rfloor} i \\
&= \frac{\lfloor \frac{n}{5} \rfloor * (\lfloor \frac{n}{5} \rfloor + 1)}{2} \in \Theta(n^2)
\end{aligned} \tag{3}
$$

Also, from part (a), we know that $\sum\limits_{i=1}^{n} \lceil \frac{n}{i} \rceil \in \Theta(n \log n)$.

So, the total runtime of **print-multiples2** is $\Theta(n \log n) + \Theta(n^2)$, which is $\Theta(n^2)$.

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

(a) *Proof.*

The input family whose runtime is (2^n) is the list whose length is n and every element is oven, except for the last one. For example, lst = $[2, 2, ..., 2, 1]$.

For the first $n-1$ items, lst$[i]\%2 == 0$ is always true until $i = n-1$. So if statement will run execute $n-2$ times and every time takes constant time. So the runtime for if statement is $n-2$.

When $i = n-1$, the iterations run $n-2$ times. So j= 2^{n-2}. And at this time, lst$[i]\%2 == 0$ is false. So the else statement will excuse.

So for loop 2, there will be 2^{n-2} iterations and every iteration takes 1 step. So the runtime of loop 2 is 2^{n-2}.

After loop 2, i will be $2(n-1) \geq n$. So the Loop 1 will stop.

Thus, the runtime of loop 1 is the sum of if and else statement, which is $(n-2) + 2^{n-2}$.

Since there is 1 extra steps in **alg** except loop 1, the total running time of **alg** is t$(n-2) + 2^{n-2} + 1$, which is $\Theta(2^n)$

(b) *Proof.*

An input family whose running time is $\Theta(\log n \times 2^{\sqrt{n}})$ is a list of size n whose first $\lceil \sqrt{n} \rceil$ items are even, then the rest are odd.

So for the first $\lceil \sqrt{n} \rceil - 1$ iterations of loop 1, $lst[i]\%2 == 0$ is always true.

After $\lceil \sqrt{n} \rceil - 1$ iterations, $i = \lceil \sqrt{n} \rceil, j = 2^{\lceil \sqrt{n} \rceil - 1}$. And then, all elements are even, so lst$[i]\%2 == 0$ is false.

So for Loop 2, there will be $2^{\lceil \sqrt{n} \rceil - 1}$iterations, and every iteration take 1 step. And Loop 1 will stop until i.

Let i_k be the value of i after k iterations of else step. $i_k = 2^k * \sqrt{n}$. So when $k > \frac{1}{2} \log n$, the loop will stop.

So the if statement executes $\lceil \sqrt{n} \rceil - 1$ times and every time has 1 step. And the else statement executes $\lceil \frac{1}{2} \log n \rceil$ time and every time costs $2^{\lceil \sqrt{n} \rceil - 1}$ step.

Thus, the runtime of loop 1 is $\lceil \sqrt{n} \rceil - 1 + \lceil \frac{1}{2} \log n \rceil * 2^{\lceil \sqrt{n} \rceil - 1}$.

Except the loop 1, **alg** has a constant step (since they are independent with the input size). So the total run time of **alg** is $\lceil \sqrt{n} \rceil - 1 + \lceil \frac{1}{2} \log n \rceil * 2^{\lceil \sqrt{n} \rceil - 1} + 1$, which is $\Theta(\log n * 2^{\sqrt{n}})$

(c) *Proof.*
In the worst case, we want the loop 2 executes as much as possible. We can know that the larger j is, the more executing time that loop 2 have.
When else statement runs 1 time and in order to get a larger j, if statement runs $n-2$ times. So, j will be 2^{n-2} and the runtime of if statement is $n-2$. So, loop 2 will run 2^{n-2} times. Also, every iterations takes constant time (since the runtime doesn't depend on the input size). So the runtime of loop 2 is 2^{n-2}, which means that the runtime of else statement is 2^{n-2}. Thus the total runtime of **alg** is $(n-2)+2^{n-2}$
When else statement runs 2 times and in order to get a larger j, if statement runs $\lfloor \frac{n-1}{2} \rfloor - 1$ times. So, j will be $2^{\lfloor \frac{n-1}{2} \rfloor - 1}$ and the runtime of if statement is $\lfloor \frac{n-1}{2} \rfloor - 1$. So, loop 2 will run $2^{\lfloor \frac{n-1}{2} \rfloor - 1}$ times. Also, every iterations takes constant time (since the runtime doesn't depend on the input size). So the runtime of loop 2 is $2^{\lfloor \frac{n-1}{2} \rfloor - 1}$. And else statement runs twice, so the runtime of else statement is $2 * 2^{\lfloor \frac{n-1}{2} \rfloor - 1}$, which is $2^{\lfloor \frac{n-1}{2} \rfloor}$. Thus the total runtime of **alg** is $\lfloor \frac{n-1}{2} \rfloor - 1 + 2^{\lfloor \frac{n-1}{2} \rfloor}$.
We can find that $n - 2 + 2^{n-2} \geq \lfloor \frac{n-1}{2} \rfloor - 1 + 2^{\lfloor \frac{n-1}{2} \rfloor}$. Thus, we can know that the larger j is, the more runtime is. SO, when id statement executes $n-2$ times, else statement executes 1 time, the runtime of **alg** will be the largest.
So, the total runtime of the worst case of **alg** is $n - 2 + 2^{n-2}$, which is $\mathcal{O}(2^n)$.

3. Rearrangements, best-case analysis

(a) i. $\forall n \in \mathbb{N}, BC_{func}(n) \leq f(n)$
$\iff \forall n \in \mathbb{N}, \min\{\text{running time of executing } func(x) | \, x \in \mathcal{I}n\} \leq f(n)$
$\iff \forall n \in \mathbb{N}, \exists x \in \mathcal{I}n, \text{ running time of executing } func(n) \leq f(n)$

ii. $\forall n \in \mathbb{N}, BC_{func}(n) \geq f(n)$
$\iff \forall n \in \mathbb{N}, \min\{\text{running time of executing } func(x) | \, x \in \mathcal{I}n\} \geq f(n)$
$\iff \forall n \in \mathbb{N}, \forall x \in \mathcal{I}n, \text{ running time of executing } func(x) \geq f(n)$

(b) Part 1. Upper bound
Let lst be the list with length n. And all the elements of lst are 1. Since every element are equal, so loop 2 and loop 3 will never execute. So for loop 1, there are $n-2$ iterations, and every iteration takes 1 step (since they are independent with input size). So the total runtime is $n-2$, which is $\mathcal{O}(n)$
Part 2. Lower bound
For a list which length is n, no matter what happen, Loop 1 has $n-2$ iteration. And in the best case, the running time is minimum when Loop 2 and Loop 3 do not executed. And that will happen when the elements are all the same.
So there are at least $n-2$ iterations will occur, and each iteration takes 1 step.
This gives us a lower bound on the number of steps as $n-2 \in \Omega(n)$
In conclusion, the best-case running time of **rearrange** is $\Theta(n)$