CSC165H1: Problem Set 4

Michael Zhou

2023-04-03

1. **Nested loops.**

   (a) Let s_t be the value of variable s and i_t be the value of variable i immediately after t iterations of Loop 1 have occured. Determine a general formula for them and find the exact number of iterations of Loop 1 given value of n.

   Looking at i_t and s_t over t iterations, we see a trend:

t	i_t	s_t
0	0	1
1	1	3
2	4	5
3	9	7
4	16	9
5	25	11

   We can see that $i_t = t^2$ and $s_t = 2t + 1$.

   Loop 1 terminates when $i \geq n$, substituting i we get

$$\begin{aligned} t^2 &\geq n \\ t &\geq \sqrt{n} \\ t &\geq \lceil \sqrt{n} \rceil \end{aligned}$$

   So Loop 1 runs for $\lceil \sqrt{n} \rceil$ iterations before terminating.

(b) Give a Theta bound on the running time function $RT(n)$ for this algorithm. Show your work.

Let $n \in \mathbb{N}$. We know that this algorithm **simply depends on the positive integer n, so the best-case would be the same as the worst-case, and the upper bound would be equal to the lower bound.**

Loop 3 breaks when $k \leq 1$, where $k = n$, and k decreases by a floor division of 2 every t iterations. So we would have $k_t = \left\lfloor \frac{n}{2^t} \right\rfloor$. Which gives us

$$\begin{aligned} \left\lfloor \frac{n}{2^t} \right\rfloor &\leq 1 \\ n &\leq 2^t && \text{(Since } \left\lfloor \tfrac{n}{2^t} \right\rfloor \leq \tfrac{n}{2^t}\text{)} \\ \log_2 n &\leq t \end{aligned}$$

Then we would have at least $\log_2 n$ iterations, or in other words $\lceil \log_2 n \rceil$ iterations for Loop 3 given a fixed number of iterations for Loop 2. This cost remains the same for each iteration of Loop 2.

Loop 2 breaks when $j \geq i$, and j_m given m iterations is $3m$. So by part (a), given t loops of Loop 1, $3m \geq i_t$, and $m \geq \frac{t^2}{3}$. Which means Loop 2 runs for $\left\lceil \frac{t^2}{3} \right\rceil$ iterations. Expressed as a sum, we get

$$\sum_{j=0}^{\left\lceil \frac{t^2}{3} \right\rceil} \lceil \log n \rceil = \lceil \log n \rceil \cdot \sum_{j=0}^{\left\lceil \frac{t^2}{3} \right\rceil} 1 = \lceil \log n \rceil \cdot \left(\left\lceil \frac{t^2}{3} \right\rceil + 1 \right)$$

steps for Loop 2 given fixed iterations of Loop 1.

Loop 1 runs for $\lceil \sqrt{n} \rceil$ iterations from part (a), so we get the sum

$$\begin{aligned} \sum_{i=0}^{\lceil \sqrt{n} \rceil} \lceil \log n \rceil \cdot \left(\left\lceil \frac{t^2}{3} \right\rceil + 1 \right) &= \lceil \log n \rceil \cdot \left(\frac{1}{3} \cdot \sum_{i=0}^{\lceil \sqrt{n} \rceil} t^2 + \sum_{i=0}^{\lceil \sqrt{n} \rceil} 1 \right) \\ &= \lceil \log n \rceil \cdot \left(\frac{\lceil \sqrt{n} \rceil \, (\lceil \sqrt{n} \rceil + 1)(2 \lceil \sqrt{n} \rceil + 1)}{18} + \lceil \sqrt{n} \rceil + 1 \right) \end{aligned}$$

We take the fastest-growing term, which is the first term, to represent asymptotic growth. This is because if $g \in \Theta(h)$ and $f \in \mathcal{O}(g)$, then $f + g \in \Theta(h)$. We can also remove constants and constant factors since it is asymptotic.

$$\begin{aligned} \lceil \log n \rceil \cdot \frac{\lceil \sqrt{n} \rceil \, (\lceil \sqrt{n} \rceil + 1)(2 \lceil \sqrt{n} \rceil + 1)}{18} &= \log n \cdot \sqrt{n} \cdot \sqrt{n} \cdot \sqrt{n} \\ &= \log n \cdot n \cdot \sqrt{n} \end{aligned}$$

Since we know that the upper bound and lower bound are equal, the tight bound on function $RT(n)$ for this algorithm is $\Theta(n \log(n) \sqrt{n})$.

2. **Worst-case analysis.**

(a) Find, with proof, an upper bound on the worst-case runtime of the algorithm matching the lower bound.

Given iterations t of the outer while loop and length of the input list n, we can see i changes by either $i_t = 2t$ or $i_t = t$, and breaks when $i_t \geq n$. So the outer loop can run for at most n iterations ($i_t = \text{t}$) and at least $\lceil \frac{n}{2} \rceil$ iterations.

The inner loop in the case that $i_t = 2t$ essentially resets the loop variable to i_t before running each time, it breaks when $\lfloor \frac{i_t}{2^t} \rfloor$. So in terms of i_t, this loop would run for $\lceil \log i_t \rceil$ times. Writing this as a cost of the outer loop in the worst case we get

$$\begin{aligned}
\sum_{t=1}^{n} \log 2t &= n \cdot \log 2 + (\log 1 + \log 2 + \ldots + \log n) \\
&= n \cdot \log 2 + \log(1 \cdot 2 \cdot 3 \cdot \ldots n) && \text{(Since } \log a + \log b = log(ab)\text{)} \\
&= n \cdot \log 2 + \log(n!)
\end{aligned}$$

So counting the variable initialization steps outside the loop, we have $1 + n \cdot \log 2 + \log(n!)$. Taking the fastest growting term we have

$$\log(n!) = 1 \cdot 2 \cdot \ldots \cdot n$$

An upper bound on $\log(n!)$ would be assuming all terms were n, which is $\log(n^n) = n \cdot \log n$. We will now prove that this is True.

Let $c = 1$, $n_0 = 1$. Assume $n \geq n_0$. We will show that $\log(n!) \in \mathcal{O}(n \cdot \log n)$.

$$\begin{aligned}
\log(n!) &= \log n + \ldots + \log 1 \\
&\leq \log n + \ldots + \log n \\
&= \log(n^n) && \text{(Since } \log a + \log b = \log(ab)\text{)} \\
&= 1 \cdot n \cdot \log n \\
&= c \cdot n \cdot \log n \\
&\in \mathcal{O}(n \cdot \log n)
\end{aligned}$$

Thus $\log(n!) \in \mathcal{O}(n \cdot \log n)$.

So the upper bound on the worst-case runtime would be $\mathcal{O}(n \cdot \log n)$.

■

(b) Find, with proof, a lower bound on the worst-case runtime of the algorithm matching the upper bound.

Let $n \in \mathbb{N}$. Consider a list A of length n containing only the integer 3, such that it always enters into the first condition of the algorithm. We want to show that the lower bound on the worst case runtime of this algorithm is $\Omega(n \cdot \log n)$.

Since all items in A are divisible by 3, it would always enter the first condition in the while loop.

From part (a), we know this will run for at least $\lceil \frac{n}{2} \rceil$ iterations, with a cost of $\log 2t$ for each t iterations.

$$\begin{aligned}\sum_{t=1}^{\lceil \frac{n}{2} \rceil} \log 2t &= \left\lceil \frac{n}{2} \right\rceil \cdot \log 2 + \log(1 \cdot 2 \cdot 3 \cdot \ldots \cdot \cdots \cdot \left\lceil \frac{n}{2} \right\rceil) \\ &= \left\lceil \frac{n}{2} \right\rceil \cdot \log 2 + \log(\left\lceil \frac{n}{2} \right\rceil !)\end{aligned}$$

Taking the fastest growing term of $\log(\lceil \frac{n}{2} \rceil !)$, we will prove that this term is $\Omega(n \cdot \log n)$.

Let $c = \frac{1}{8}$, $n_0 = 16$. Let $n \in \mathbb{N}$, and assume $n \geq n_0$. We will show that $\log(\lceil \frac{n}{2} \rceil !) \in \Omega(n \log n)$.

$$\begin{aligned}\log(\left\lceil \frac{n}{2} \right\rceil !) &= \log(\left\lceil \frac{n}{2} \right\rceil) + \ldots + \log 1 \\ &\geq \log(\left\lceil \frac{n}{2} \right\rceil) + \ldots + \log(\left\lceil \frac{n}{4} \right\rceil) && \text{(Taking half the terms)} \\ &\geq \left\lceil \frac{n}{4} \right\rceil \cdot \log(\left\lceil \frac{n}{4} \right\rceil) && \text{(All cases are less than equal to original)} \\ &\geq \frac{n}{4} \cdot \log(\frac{n}{4}) && (\lceil \tfrac{n}{4} \rceil \geq \tfrac{n}{4}) \\ &= \frac{n}{4} \log n - \frac{n}{4} \log 4 \\ &\geq \frac{n}{4} \log n - \frac{n}{8} \log n && \text{(Since } n \geq 16) \\ &= \log n \cdot \left(\frac{n}{4} - \frac{n}{8}\right) \\ &= \frac{n}{8} \log n \\ &= \frac{1}{8} n \cdot \log n \\ &= c \cdot n \log n \\ &\in \Omega(n \log n)\end{aligned}$$

Thus we have proven that $\log(\lceil \frac{n}{2} \rceil !)$ is $\Omega(n \cdot \log n)$.

Since both the upper bound and the lower bound are the same, this algorithm is $\Theta(n \cdot \log n)$

■

3. **Average-case analysis.**

(a) Find a set of inputs $\mathcal{I}_{has\_even,n}$ for which $Avg_{has\_even}(n)$ is $\Theta(1)$.

Let $\mathcal{I}_{has\_even,n} = \{\text{Input lists } A \mid \text{len}(A) = n \wedge (\exists k \in \mathbb{Z}, A[0] = 2k)\}$. Since the first index is always even, has_even will always return on the first index, which makes the algorithm have a constant runtime, which means $Avg_{has\_even}(n) \in \Theta(1)$.

(b) Find a set of inputs $\mathcal{I}_{has\_even,n}$ for which $Avg_{has\_even}(n)$ is $\Theta(n)$.

Let $\mathcal{I}_{has\_even,n} = \{\text{Input lists } A \mid \text{len}(A) = n \wedge (\forall i \in \text{range}(n), \exists k \in \mathbb{Z}, A[i] = 2k+1)\}$. Since all numbers in the lists are odd, the algorithm loops for n iterations on any of the input lists described in this set before returning False, which is $\Theta(n)$.

(c) $S_n = \{\text{Input lists } numbers \text{ to } has\_even \mid \forall i \in \text{range}(n), 1 \leq numbers[i] \leq n\}$

Consider the set of allowable inputs $\mathcal{I}_{has\_even,n} = S_n$. Give an expression for $|\mathcal{I}_{has\_even,n}|$.

Since we can have repeats within the pairs, we know that each index of a list would have n choices, and the list is n length.

$$n \times n \times \ldots \times n \times n \ (n \text{ times})$$

So $|\mathcal{I}_{has\_even,n}| = n^n$.

(d) Calculate an exact expression for $Avg_{has\_even}(n)$ for the set of allowable inputs $\mathcal{I}_n = S_n$, where S_n is as defined in the previous part.

To find the average-case for has_even, we need to take the weighted average of the runtimes for the set of allowable inputs S_n. For every input list in S_n where the first even number is at index i, the algorithm runs exactly $i+1$ times. When there does not exist an even number, there are $(n+1)$ steps of the loop and the return statement. Substituing these steps into the weighted average we have

$$Avg_{has\_even}(n) = \frac{1}{|S_n|} \sum_{i=0}^{n-1} \sum_{lst \in S_n} (1+i) \qquad \text{(Where the first even number is at } i\text{)}$$

We know from part (c) $|S_n| = n^n$. Consider a list where the first even number is at i, then at that index, there would be $\left\lfloor \frac{n}{2} \right\rfloor$ possibilities where $lst[i]$ is even. Counting up all the previous indices, then the possibilties where all indices up to and including i are even can be expressed as $\left\lfloor \frac{n}{2} \right\rfloor^i$. We also have to consider the indices after i, which could be either even or odd, so n^{n-i-1} possibilities.

Let $a = \left\lceil \frac{n}{2} \right\rceil$, let $b = \left\lfloor \frac{n}{2} \right\rfloor$, and $r = \frac{a}{n}$ Then we have

$$\begin{aligned}
Avg_{has_even}(n) &= \frac{1}{n^n} \left[\sum_{i=0}^{n-1} (i+1) \left\lceil \frac{n}{2} \right\rceil^i \left\lfloor \frac{n}{2} \right\rfloor n^{n-i-1} \right] + \frac{1}{n^n}(n+1) \left\lceil \frac{n}{2} \right\rceil^n \\
&= \left[\sum_{i'=1}^{n} i' a^{i'-1} b \frac{n^{n-i'}}{n^n} \right] + \frac{a^n}{n^n}(n+1) \qquad (i' = i+1) \\
&= \left[\sum_{i'=1}^{n} i' \frac{b}{a} \left(\frac{a}{n} \right)^{i'} \right] + r^n(n+1) \\
&= \frac{b}{a} \cdot \left[\sum_{i'=1}^{n} i' r^{i'} \right] + r^n(n+1)
\end{aligned}$$

We will now split the calculations to the cases where n is even and n is odd.

Case 1. Assume $n = 2k$, where $k \in \mathbb{Z}$.

Then we know that $\lceil \frac{2k}{2} \rceil = \lfloor \frac{2k}{2} \rfloor = \frac{2k}{2} = a = b$. We also know that $r = \frac{k}{2k} = \frac{1}{2}$. We will simplify the first term then the second term before combining:

$$\begin{aligned}
1 \cdot \sum_{i=1}^{n} i' \left(\frac{1}{2}\right)^{i'} &= \sum_{i'=0}^{n} i' \left(\frac{1}{2}\right)^{i'} && \text{(Term at } i = 0 \text{ is 0)} \\
&= \frac{n(\frac{1}{2})^{n+1}}{(\frac{1}{2}) - 1)} - \frac{\frac{1}{2}((\frac{1}{2})^n - 1)}{(\frac{1}{2} - 1)^2} && \text{(Closed-form)} \\
&= -\frac{n}{2^n} - 2\left(\left(\frac{1}{2}\right)^n - 1\right) \\
\frac{(n+1)(\frac{n}{2})^n}{n^n} &= \frac{(n+1)n^n}{2^n n^n} \\
&= \frac{(n+1)}{2^n} \\
Avg_{has_even(even\, n)}(n) &= -\frac{n}{2^n} - 2\left(\left(\frac{1}{2}\right)^n - 1\right) + \frac{(n+1)}{2^n} \\
&= \frac{1}{2^n} - 2(\frac{1}{2^n} - 1) \\
&= 2 - \frac{1}{2^n}
\end{aligned}$$

So we have $2 - \frac{1}{2^n}$ steps for $Avg_{has\_even}(n)$ when n is even.

Case 2. Assume $n = 2k + 1$, where $k \in \mathbb{Z}$.

Then we know $a = \frac{n+1}{2}$,and $b = \frac{n-1}{2}$. We will simplify the first term before combining:

$$\begin{aligned}
\frac{b}{a} \sum_{i'=1}^{n} i' r^{i'} &= \frac{b}{a} \sum_{i'=0}^{n} i' r^{i'} && \text{(Term at } i' = 0 \text{ is 0)} \\
&= \frac{n-1}{n+1}\left(\frac{nr^{n+1}}{r-1} - \frac{r(r^n - 1)}{(r-1)^2}\right) \\
Avg_{has_even(odd\, n)}(n) &= (n+1)r^n + \frac{n-1}{n+1}\left(\frac{nr^{n+1}}{r-1} - \frac{r(r^n - 1)}{(r-1)^2}\right)
\end{aligned}$$

Thus the exact expression would be

$$Avg_{has\_even}(n) = \begin{cases} 2 - \frac{1}{2^n}, & \text{when } n \text{ is even} \\ (n+1)r^n + \frac{n-1}{n+1}\left(\frac{nr^{n+1}}{r-1} - \frac{r(r^n-1)}{(r-1)^2}\right), & \text{when } n \text{ is odd} \end{cases} \tag{1}$$

where $r = \frac{a}{n} = \frac{\lceil \frac{n}{2} \rceil}{n}$.