Worksheet 13 Review

March 31, 2020

Question 1

a. Since the loop starts from $i = 0$ and ends at $i = n - 1$. The loop has

$$n - 1 - 0 + 1 = n \tag{1}$$

iterations.

Since each iteration runs 5 steps, the loop has total cost of

$$5 \cdot n = 5n \tag{2}$$

steps.

Because we know $i = 0$ at line 2 has cost of 1, we can conclude that the algorithm has total cost of $5n + 1$ steps.

Correct Solution:

Because we know the loop starts from $i = 0$ and ends at $i = n - 1$ with i increasing by 5 per iteration, we can conclude the loop has

$$\left\lceil \frac{n}{5} \right\rceil \tag{3}$$

iterations.

Since each iteration takes constant time, the loop has runtime of $\Theta(n)$

Notes:

- How does professor begin a proof after 'We will prove that...' or at the beginning of each case/parts?
- Noticed professor doesn't provide a detailed explanation for the number of iterations.
- Realized the goal of this problem is to determine the exact cost and runtime of each loop.

There are $\lceil \frac{n}{5} \rceil$ iterations. $\cdots$

b. Because we know the loop starts at $i = 4$ and ends at $i = n - 1$ with i increasing by 1 per iteration, we can conclude that the loop has

$$\lceil n - 14 + 1 \rceil = n - 4 \tag{1}$$

iterations.

Since each iteration takes a constant time, we can conclude that the loop has runtime of $\mathcal{O}(n)$.

Correct Solution:

Because we know the loop starts at $i = 4$ and ends at $i = n - 1$ with i increasing by 1 per iteration, we can conclude that the loop has **at most**

$$\lceil n - 14 + 1 \rceil = n - 4 \tag{1}$$

iterations.

Since each iteration takes a constant time, we can conclude that the loop has runtime of $\Theta(n)$.

Notes:

- Noticed professor doesn't count $i = 4$ or $i = 0$ in question 1.a to the total cost of algorithm. But in later parts of the question, the two are considered. I wonder if the line with constant runtime should always be accounted for, or if it can be ignored in certain circumstances. If latter, when can the constants be ignored?

c. Since the loop starts at $i = 0$ and ends at $i = n - 1$ with i increasing by $\frac{n}{10}$ per iteration, we can conclude that the loop has at most

$$\left\lceil \frac{(n-1-0+1)}{\frac{n}{10}} \right\rceil = \left\lceil \frac{n \cdot 10}{n} \right\rceil \tag{1}$$
$$= 10 \tag{2}$$

iterations.

Since each iteration takes constant time, we can conclude that the loop has runtime of $\Theta(1)$.

d. For the case where $n^2 \leq 20$, it's omitted because we are assuming the value of n is asymptotically large.

For the case where $n^2 > 20$, because we know the loop runs from $i = 20$ to $i = n^2 - 1$ with i increasing by 3 per iteration, we can conclude that the loop has at most

$$\left\lceil \frac{n^2 - 1 - 20 + 1}{3} \right\rceil = \left\lceil \frac{n^2 - 20}{3} \right\rceil \tag{1}$$

iterations.

Since the loop takes constant time per iteration, the loop has total runtime of $\Theta(n^2)$.

e. Since we are considering asymptotic runtime or the case where n is very large, the case where $n^2 \leq 20$ will be omitted.

For the case where $n^2 > 20$, because we know the first loop runs from $i = 20$ and ends at $i = n^2 - 1$ with i increasing by 3 per iteration, we can conclude that the first loop runs

$$\left\lceil \frac{n^2 - 1 - 20 + 1}{3} \right\rceil = \left\lceil \frac{n^2 - 20}{3} \right\rceil \tag{1}$$

iterations

For the second loop, because we know the loop runs from $j = 0$ to $j = n-1$ with j increasing by 0.01 per iteration, we can conclude that the second loop runs at most

$$\begin{aligned} \left\lceil \frac{n - 1 - 0 + 1}{\frac{1}{100}} \right\rceil &= \lceil 100 \cdot n \rceil &(2) \\ &= 100 \cdot n &(3) \end{aligned}$$

iterations

Since each iteration takes a constant time in both of the loops, the total runtime of the loops in this algorithm is

$$\Theta\left(\left\lceil \frac{n^2 - 20}{3} \right\rceil + 100 \cdot n\right) = \Theta(n^2) \tag{4}$$

Correct Solution:

For the case where $n^2 \leq 20$, it's omitted because we are assuming the value of n is asymptotically large.

For the case where $n^2 > 20$, because we know the loop runs from $i = 20$ to $i = n^2 - 1$ with i increasing by 3 per iteration, we can conclude that the loop has at most

$$\left\lceil \frac{n^2 - 1 - 20 + 1}{3} \right\rceil = \left\lceil \frac{n^2 - 20}{3} \right\rceil \tag{1}$$

iterations.

Since the loop takes constant time per iteration, the loop has total runtime of $\Theta(n^2)$.

Since we are considering asymptotic runtime or the case where n is very large, the case where $n^2 \leq 20$ will be omitted.

For the case where $n^2 > 20$, because we know the first loop runs from $i = 20$ and ends at $i = n^2 - 1$ with i increasing by 3 per iteration, we can conclude that the first loop runs

$$\left\lceil \frac{n^2 - 1 - 20 + 1}{3} \right\rceil = \left\lceil \frac{n^2 - 20}{3} \right\rceil \tag{1}$$

iterations

Since the first loop takes a constant time per iteration, we can conclude the first loop has runtime of $\Theta(n^2)$.

For the second loop, because we know the loop runs from $j = 0$ to $j = n - 1$ with j increasing by 0.01 per iteration, we can conclude that the second loop runs at most

$$\left\lceil \frac{n - 1 - 0 + 1}{\frac{1}{100}} \right\rceil = \lceil 100 \cdot n \rceil \tag{2}$$
$$= 100 \cdot n \tag{3}$$

iterations

Since the first loop takes a constant time per iteration, we can conclude the second loop has runtime of $\Theta(n)$.

Since $n \in \Theta(n^2)$, the total runtime of algorithm is $\Theta(n^2)$.

Notes:

- Noticed professor computes the theta of each loop, and then compare to choose the biggest theta. Professor calls it **one version of the "sum" Big-Oh/Omega/Theta theorem.**
  - Ah. this is also how $\in$ in $n \in \Theta(n)$ is used.

Question 2

a. It follows from the fact $i_k = i \cdot 2$ that we can conclude

$$i_3 = 8$$
$$i_4 = 16$$
$$i_k = 2^k$$

b. Using the fact that loop termination occurs when $i_k \geq n$, we can calculate

$$2^k \geq n \tag{1}$$
$$\log 2^k \geq \log n \tag{2}$$
$$k \geq \log n \tag{3}$$

Since the loop terminates when k is greater than or equal to $\log n$, we can conclude that the loop has $\lceil \log n \rceil$ iterations.

c. Since the loop runs $\lceil \log n \rceil$ iterations, and constant time is taken per iteration, we can conclude that the algorithm has runtime of $\Theta(\log n)$.

d. We did not initialize $i = 0$ to prevent the loop from running indefinitely.

Question 3

- *Proof.* Let $k \in \mathbb{Z}^+$.

  Using the fact $i = i \cdot i$, we can calculate

$$i_1 = 4 = 2^2 \tag{1}$$
$$i_2 = 16 = 2^4 \tag{2}$$
$$i_3 = 256 = 2^8 \tag{3}$$
$$i_k = 2^{2^k} \tag{4}$$

  Since the loop terminates when $i_k \geq n$, we can conclude

$$2^{2^k} \geq n \tag{5}$$
$$\log 2^{2^k} \geq \log n \tag{6}$$
$$2^k \geq \log n \tag{7}$$
$$\log 2^k \geq \log \log n \tag{8}$$
$$k \geq \log \log n \tag{9}$$

Since we are looking for the smallest k when the loop terminates, the smallest k possible is $\lceil \log \log n \rceil$.

Since the loop takes constant time per iteration, we can conclude the runtime of the algorithm is $\Theta(\log \log n)$ $\square$