Worksheet 15 Review

April 2, 2020

Question 1

a. First, we will evaluate the cost of he inner most loop.

Because the loop runs from $j = i = 1$ to $j = n - 1$, with each iteration costing 1 step, we can conclude that the inner most loop has cost of at most

$$\lceil (n-1) - (i+1) + 1 \rceil = n - i - 1 \tag{1}$$

steps.

Next, we will evaluate the cost of the outer most loop.

Because the loop runs from $i = 0$ to $i = n-1$ with each iteration costing $(n-i-1)$ steps, we can conclude the outer most loop has cost of at most

$$\sum_{i=0}^{n-1}(n-i-1) = \left[\sum_{i=0}^{n-1}(n-1) - \sum_{i=0}^{n-1} i\right] \tag{2}$$

$$= \left[\frac{2n(n-1)}{2} - \frac{(}{n}(n-1))2\right] \tag{3}$$

$$= \frac{n(n-1)}{2} \tag{4}$$

steps.

Next, we will bring everything together.

Since the lines **n = len(lst)** and **return False** have cost of 1 step each, the total cost of the algorithm is

$$\frac{n(n-1)}{2} + 2 \tag{5}$$

steps.

Then, it follows from above that the algorithm has runtime of $\Theta(n^2)$.

Correct Solution:

First, we will evaluate the cost of he inner most loop.

Because the loop runs from $j = i = 1$ to $j = n - 1$, with each iteration costing 1 step, we can conclude that the inner most loop has cost of at most

$$\lceil (n-1) - (i+1) + 1 \rceil = n - i - 1 \tag{6}$$

steps.

Next, we will evaluate the cost of the outer most loop.

Because the loop runs from $i = 0$ to $i = n - 1$ with each iteration costing $(n - i - 1)$ steps, we can conclude the outer most loop has cost of at most

$$\sum_{i=0}^{n-1}(n-i-1) = \left[\sum_{i=0}^{n-1}(n-1) - \sum_{i=0}^{n-1} i\right] \tag{7}$$

$$= \left[\frac{2n(n-1)}{2} - \frac{(}{n}(n-1))2\right] \tag{8}$$

$$= \frac{n(n-1)}{2} \tag{9}$$

steps.

Next, we will bring everything together.

Since the lines **n = len(lst)** and **return False** have cost of 1 step each, the total cost of the algorithm is at most

$$\frac{n(n-1)}{2} + 2 \tag{10}$$

steps.

Then, it follows from above that the algorithm has runtime of $\mathcal{O}(n^2)$.

Notes:

- Noticed that in here, professor considers the cost of loop variables and other lines with constant time.
- $\mathcal{O}$ used since we are determining the upper bound.
- In worksheet 14, the cost of loop variables is not required.

b. Let $n \in \mathbb{N}$, and $lst = [0, 1, 2, 3, \ldots, n-3, n-1, n-1]$.

First, we will calculate the cost of the inner most loop.

Because we know the inner most loop will terminate when **if lst[i] == lst[j]** and because we know this condition occurs when $i = n-2$, we can conclude the loop will start at $i = 0$ and run until $i = n-2$.

Because we know the condition of the inner most loop **for j in range(i+1,n)** stays true until $i = n-2$ even at the worst case, we can conclude that the cost of inner loop is the same as the cost of the inner loop at worst case, that is

$$n - i - 1 \tag{1}$$

Next, we will evaluate the cost of the outer most loop.

Since the outer most loop starts at $i = 0$ and ends at $i = n-1$ with each iteration costing $(n - i - 1)$ steps, the outer most loop has cost of

$$\sum_{i=0}^{n-1}(n - i - 1) = \frac{n(n-1)}{2} \tag{2}$$

steps.

Next, we will combine everything together.

Since each of the lines **n = len(lst)** and **return True** have cost of 1 step, we can conclude that the algorithm has total cost of

$$\frac{n(n-1)}{2} + 2 \tag{3}$$

steps.

Then, we can conclude the algorithm has runtime of $\Omega(n^2)$.

Because we know both $\mathcal{O}(n^2)$ and $\Omega(n^2)$ are true, we can also conclude the algorithm has runtime of $\Theta(n^2)$.

Notes:

- Does **if condition** counted towards the total cost?

c. Let $n \in \mathbb{N}$, and $lst = [1, 2, 3, \ldots, n-2, 0]$.

We will prove that given the input group, the runtime of the algorithm is $\Theta(n)$ by computing the total cost of the algorithm, starting off with the inner most loop, then the the outer most loop, and then the rest.

First, we will calculate the cost of the inner most loop.

Because we know the loop starts at $j = i + 1$ and finishes at $j = n - 1$ with each iteration costing 1 step, we can conclude the inner most loop has cost of

$$n - 1 - (i + 1) + 1 = n - i \tag{1}$$

steps.

Next, we will calculate the cost of the outer most loop.

Because we know the outer most loop terminates by the **if condition** at $i = 0$, we can conclude the outer most loop runs only 1 iteration.

Then, since each iteration costs $(n - i - 1)$ steps, the cost of the outer most loop is

$$1 \cdot (n - 0 - 1) = n - 1 \tag{2}$$

steps.

Next, we will combine everything together, and evaluate theta.

Because we know both of the lines **n = len(lst)** and **return True** have cost of 1, the total cost of the algorithm is

$$n - 1 + 2 = n + 1 \tag{3}$$

steps.

Then, it follows from above that the runtime of the algorithm is $\Theta(n)$.

Question 2

a. Because we know the loop starts at $i = 0$ and $j = n$, we can conclude the initial value of r, in terms of n is n.

b. Since the loop condition is true as long as $i < j$, the loop termination will occur when $i \geq j$, or $r \leq 0$.

c. Let $k \in \mathbb{N}$, $x \in \mathbb{R}$, and $r_k = j - i$.

We will prove the statement $r_{k+1} \leq \frac{1}{2} r_k$ using proof by cases.

Case 1 ($lst[mid] < x$):

Proof. Assume $lst[mid] < x$.

Because we know $i_{k+1} = \lfloor \frac{i+j}{2} \rfloor + 1$ and $j_{k+1} = j$, we can conclude the new value of r or r_{k+1} is

$$r_{k+1} = j_{k+1} - i_{k+1} \tag{1}$$

$$r_{k+1} = j - \left(\left\lfloor \frac{i+j}{2} \right\rfloor + 1 \right) \tag{2}$$

$$\leq j - \left\lfloor \frac{(i+j)}{2} \right\rfloor \tag{3}$$

Then, using a fact about floor/ceil $\forall x \in \mathbb{Z}$, $\forall y \in \mathbb{R}$, $\lfloor x + y \rfloor = x + \lfloor y \rfloor$, we can calculate that

$$r_{k+1} \leq -\left(\left\lfloor \frac{i+j}{2} \right\rfloor + (-j)\right) \tag{4}$$
$$\leq -\left(\left\lfloor \frac{i+j}{2} - j \right\rfloor\right) \tag{5}$$
$$\leq -\left(\left\lfloor \frac{i-j}{2} \right\rfloor\right) \tag{6}$$

Then,

$$r_{k+1} \leq -\left(\frac{i-j}{2}\right) \tag{7}$$
$$\leq \left(\frac{j-i}{2}\right) \tag{8}$$

by using the fact $\forall x \in \mathbb{R},\ \lfloor x \rfloor \leq x < 1 + \lfloor x \rfloor$.

Because we know $r_k = j - i$, we can conclude

$$r_{k+1} \leq \frac{1}{2} r_k \tag{9}$$

□

Case 2 ($lst[mid] \geq x$):

Proof. Assume $lst[mid] \geq x$.

Because we know $j_{k+1} = \lfloor \frac{i+j}{2} \rfloor$ and $i_{k+1} = i$, we can conclude the new value of r or r_{k+1} is

$$r_{k+1} = j_{k+1} - i_{k+1} \tag{1}$$
$$r_{k+1} = \left\lfloor \frac{i+j}{2} \right\rfloor - i \tag{2}$$

Then, using a fact about floor/ceil $\forall x \in \mathbb{Z},\ \forall y \in \mathbb{R},\ \lfloor x + y \rfloor = x + \lfloor y \rfloor$, we can calculate that

$$r_{k+1} \leq \left\lfloor \frac{i+j}{2} \right\rfloor + (-i) \tag{3}$$
$$\leq \left\lfloor \frac{i+j}{2} + (-i) \right\rfloor \tag{4}$$
$$\leq \left\lfloor \frac{j-i}{2} \right\rfloor \tag{5}$$

Then,

$$r_{k+1} \leq \frac{j-i}{2} \tag{6}$$

by using the fact $\forall x \in \mathbb{R},\ \lfloor x \rfloor \leq x < 1 + \lfloor x \rfloor$.

Because we know $r_k = j - i$, we can conclude

$$r_{k+1} \leq \frac{1}{2} r_k \tag{7}$$

□

Notes:

- How do we introduce the loop variables i and j with r?
  - Ah. professor assumes that we know i and j. Same with r_k and r_{k+1}. He introduces i_k, j_k, i_{k+1}, j_{k+1} as follows:

    > Let i_k, j_k, i_{k+1}, and j_{k+1} As the values of i and j immediately before and after the k^{th} iteration, respectively. So then $r_k = j_k - i_k$ and $r_{k+1} = j_{k+1} - i_{k+1}$.

- $\forall x \in \mathbb{Z},\ \forall y \in \mathbb{R},\ \lfloor x + y \rfloor = x + \lfloor y \rfloor$
- Professor wrote $-\lfloor x \rfloor = \lceil -x \rceil$ is true. Is $x \in \mathbb{R}$? Also, how does j get embedded to $-\left\lfloor \frac{j+i}{2} \right\rfloor$?

d. *Proof.* Let $k \in \mathbb{N}$, and r_k be the value of r at k^{th} iteration.

We will prove the statement by evaluating the total cost of the algorithm, and then finding theta.

First, we will calculate the total cost of the algorithm.

The result in previous problem tells us that

$$r_{k+1} \leq \frac{1}{2} \cdot r_k \tag{1}$$

Using this fact, along with the fact that $r_0 = n$, the value of r at k^{th} iteration is

$$r_k = \left\lfloor \frac{n}{2^k} \right\rfloor \tag{2}$$

Because we know the loop terminates when $r_k \leq 0$, we can conclude this condition occurs when

$$2^k > n \tag{3}$$
$$k > \log n \tag{4}$$

Since $\lceil \log n \rceil$ is the boundary of loop termination, the value of k at which it occurs is

$$\lceil \log n \rceil + 1 \tag{5}$$

Since each of the lines **i = 0**, **j = len(lst)** and **return False** have cost of 1 step, the total cost of the algorithm is

$$\lceil \log n \rceil + 4 \tag{6}$$

Then, we can conclude the algorithm has runtime of $\Theta(\log n)$.

□

Correct Solution:

Let $k \in \mathbb{N}$, and r_k be the value of r at k^{th} iteration.

We will prove the statement by evaluating the total cost of the iteration, and then finding theta.

First, we will calculate the total cost of the algorithm.

The result in previous problem tells us that

$$r_{k+1} \leq \frac{1}{2} \cdot r_k \tag{1}$$

Using this fact, along with the fact that $r_0 = n$, the value of r at k^{th} iteration is

$$r_k = \left\lfloor \frac{n}{2^k} \right\rfloor \tag{2}$$

Because we know the loop terminates when $r_k \leq 0$, we can conclude this condition occurs when

$$2^k > n \tag{3}$$
$$k > \log n \tag{4}$$

Since $\lfloor \log n \rfloor$ is the boundary of loop termination, the value of k at which it occurs is

$$\lfloor \log n \rfloor + 1 \tag{5}$$

Then, we can conclude the algorithm has runtime of $\mathcal{O}(\log n)$.

Notes:

- I feel I need help here. In previous problem ceiling notation is used to obtain the closest value at which loop termination occurs. Here floor is used. How come a floor notation chosen as opposed to ceiling notation? Are there rules when deciding which notation to use?
- Here, professor omits the consideration of the lines **i = 0, j = len(lst)** and **return False** before computing the big-oh. In some questions, the lines are included. Are there general rules of a thumb when deciding what to include and not to include when determining total cost of the algorithm?

e. *Proof.* Let $n \in \mathbb{N}$, $lst = [0, 1, \ldots, n-1]$ and $x = n + 1$.

Since x doesn't exist in lst, the loop will continue to run until termination.

Because we know the loop behaves the same was when finding the big oh, we can conclude the loop has total cost of

$$\lfloor \log n \rfloor + 1 \tag{6}$$

Then, we can conclude this algorithm has asymptotic lower bound of $\Omega(\log n)$. □