

# CSC148 Worksheet 11 Solution

Hyungmo Gu

April 22, 2020

## Question 1

a. Here, the constant time means the running time of accessing and assigning element by index doesn't depend on the length of the list.

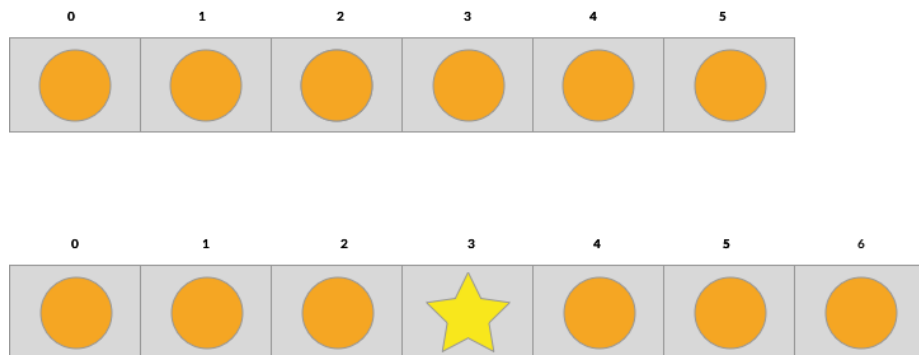
b.

$$n - i$$

many elements need to be shifted to right.

### Notes:

- The following example tells us



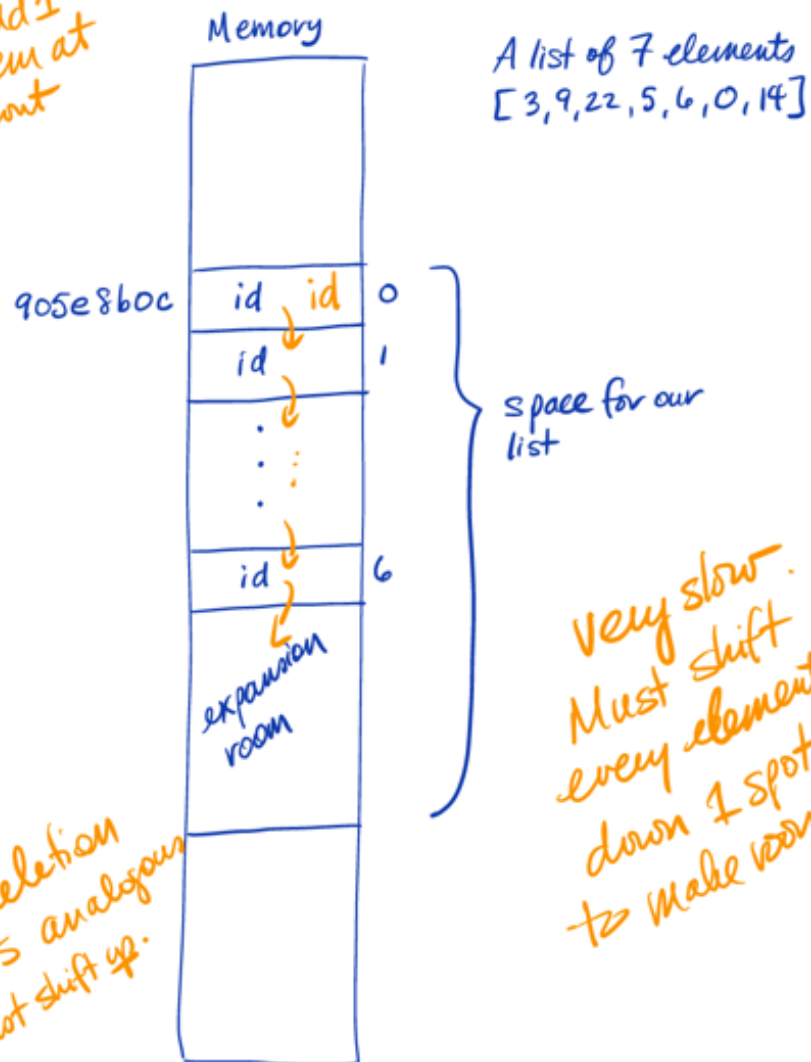
to position an element at index  $i = 3$  of the list,  $n - i = 6 - 3 = 3$  elements must be moved over.

Using this fact, we can generalize that to position an element at index  $i$  of the list,  $n - i$  many elements must be shifted.

- Learned that when items shifts, it shifts into the expansion room.

## Updates at the front of our list

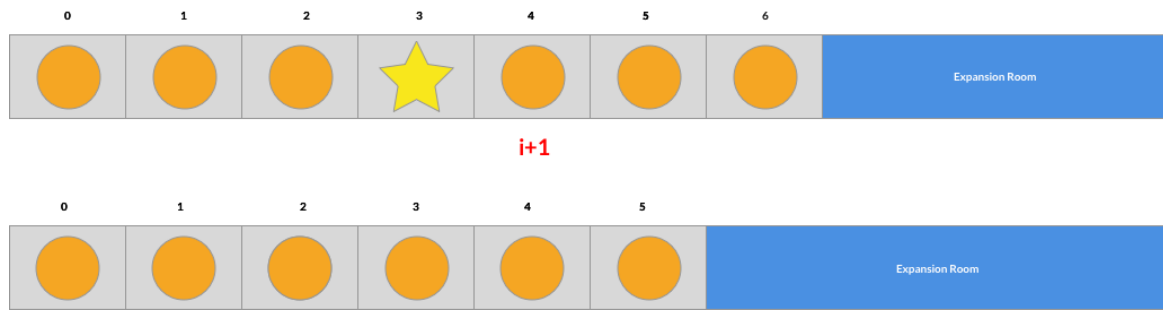
Add 1  
item at  
front



- c. Because we know the list size stays as is when an element is removed, we can conclude 0 many list elements must be moved.

### Correct Solution:

The following example tells us



when an element at index  $i = 3$  is removed from the list  $n - (i + 1) = 7 - (3 + 1) = 3$  many elements must be moved.

Using this fact, we can generalize that when an element is removed,  $n - (i + 1) = n - i - 1$  many elements must be shifted to left.

- d. i. A solution is *LIST.remove(...)*.

The answer to question 1.d tells us when an element is removed,  $n - i$  must be shifted to left.

Using this fact, we can write a list of smaller size needs to shift elements less.

Then, it follows from this fact that  $n = 100$  works faster than  $n = 1,000,000$ .

- ii. A solution is *LIST.append(...)*

The definition of append tells us that upon call, an element is added to the end of a list, it takes a constant time to add an element as long as the expansion room is not filled.

It follows from this fact that  $n = 100$  and  $n = 1,000,000$  takes roughly the same amount of time.

- e. The definition of *Queue* tells us *Queue* is FIFO. That is, the first element inserted is the first to come out.

Since we know the front of *Queue* is the front of the list, we can conclude *LIST.insert(...)* and *LIST.pop(0)* are used to support *QUEUE.enqueue* and *QUEUE.dequeue*, respectively.

Since we know *LIST.insert(...)* requires shifting of elements by  $n - i = n - 0 = n$  and *LIST.pop(0)* requires shifting of  $n - i - 1 = n - 1$  many elements, we can conclude both *QUEUE.enqueue* and *QUEUE.dequeue* takes longer time as size increases.

### Correct Solution:

The definition of *Queue* tells us *Queue* is FIFO. That is, the first element inserted is the first to come out.

Since we know the front of *Queue* is the front of the list, we can conclude *LIST.append(...)* and *LIST.pop(0)* are used to support *QUEUE.enqueue* and *QUEUE.dequeue*, respectively.

Since we know *LIST.append(...)* requires the shifting of elements by  $n - i = n - n = 0$  and *LIST.pop(0)* requires shifting of  $n - i - 1 = n - 1$  many elements, we can conclude *QUEUE.dequeue* takes longer time as size increases.

### Notes:

- Learned that the **front of queue** is where *QUEUE.dequeue* occurs.



## Question 2

a. Stack 1:

$$\begin{aligned} \text{Number of steps for } s.\text{push}(1) + \text{Number of steps for } s.\text{pop}() &= 1 + 1 \\ &= 2 \end{aligned}$$

Stack 2:

$$\begin{aligned} \text{Number of steps for } s.\text{push}(1) + \text{Number of steps for } s.\text{pop}() &= (n + 1) + (n + 1) \\ &= 2n + 1 \end{aligned}$$