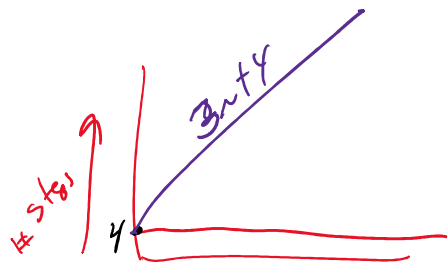


# CSE12 - Lecture 9

Wednesday, April 17, 2024 10:00 AM

## Exam 1 - Wednesday

- <https://ucsd-cse12-sp24.github.io/lectures/exam1.html>



### Lecture 9

#### Counting Steps

##### ArrayList Insert - ignore ExpandCapacity

```
public void insert(int index, String s) {
    //expandCapacity(); //ignore
    for (int i = size - 1; i >= index; i--) {
        this.elements[i+1] = this.elements[i];
    }
    this.elements[index] = s;
    this.size += 1;
}
```

Worst case  
index = 0  
size = N = 5  
i = 4

cond 1  
body 1  
update 1  
N+1 N N

$N \rightarrow$  # of elements in the structure

best case  $\rightarrow$  index at the end (add)  
index = 5 = N  
i = 4

cond 1  
body 0  
update 0

Best Case Worst Case Avg Case

Best Case	Worst Case	Avg Case
0	0	0
1 + 1 + 0	1 + (N+1) + N	1 + ( $\frac{N}{2} + 1$ ) + $\frac{N}{2}$
0	N	$\frac{N}{2}$
1	1	1
1	1	1
4	3N + 4	$\frac{3}{2}N + 4$

##### ArrayList ExpandCapacity

```
private void expandCapacity() {
    int currentCapacity = this.elements.length;
    if (this.size < currentCapacity) { return; }
    String[] expanded = new String[currentCapacity * 2];
    for (int i = 0; i < this.size; i += 1) {
        expanded[i] = this.elements[i];
    }
    this.elements = expanded;
}
```

allocate memory  
init default

Best Case Worst Case Avg Case

Best Case	Worst Case	Avg Case
1	1	
1 + 1	1 + 0	
0	1 + 2N + 2N	
0	1 + (N+1) + N	
0	N	
0	1	
3	7N + 6	

##### ArrayList Insert - with ExpandCapacity

```
public void insert(int index, String s) {
    expandCapacity();
    for (int i = size - 1; i >= index; i--) {
        this.elements[i+1] = this.elements[i];
    }
    this.elements[index] = s;
    this.size += 1;
}
```

Best Case Worst Case Avg Case

Best Case	Worst Case	Avg Case
3	7N + 6	
1 + 1 + 0	1 + (N+1) + N	
0	N	
1	1	
1	1	
7	10N + 10	

$\hookrightarrow dd()$

3 + 2 = 5  $\quad 7N + 6 + 2 = 7N + 8$

6891

Name: \_\_\_\_\_ PID: \_\_\_\_\_ Code: \_\_\_\_\_

## Counting Steps - where size of the contents is $n$

### LinkedList Add

```
public void add(String s) {
    Node current = this.front;
    while(current.next != null) {
        current = current.next;
    }
    current.next = new Node(s, null);
    this.size += 1;
}
```

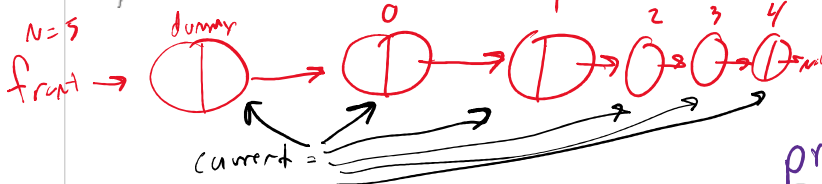
Best Case

Worst Case

Avg Case

$$\begin{array}{c} 1 \\ n+1 \\ n \\ 1 \end{array}$$

$$2n+4$$



### LinkedList Insert

```
public void insert(int index, String s) {
    Node current = this.front;
    for(int i = 0; i < index; i += 1) {
        current = current.next;
    }
    current.next = new Node(s, current.next);
    this.size += 1;
}
```

Best Case

Worst Case

Avg Case

*prepend()*

$$\begin{array}{cc} \begin{array}{c} 1 \\ 1+1+0 \\ 0 \\ 1 \\ 1 \end{array} & \begin{array}{c} 1 \\ 1+(n+1)+n \\ n \\ 1 \\ 1 \end{array} \end{array}$$

$$5 \quad 3n+5$$



### LinkedList Get

```
public String get(int index) {
    Node current = this.front.next;
    for(int i = 0; i < index; i += 1) {
        current = current.next;
    }
    return current.value;
}
```

Best Case

Worst Case

Avg Case

$$\begin{array}{cc} \begin{array}{c} 1 \\ 1+0+0 \\ 0 \\ 1 \end{array} & \begin{array}{c} 1 \\ 1+(n+1)+n \\ n \\ 1 \end{array} \end{array}$$

$$4 \quad 3n+4$$

### ArrayList Get

```
public String get(int index) {
    return this.elements[index];
}
```

Best Case

Worst Case

Avg Case

$$1 \quad 1 \quad 1$$