

# **Lecture 8 - Wednesday, February 1**

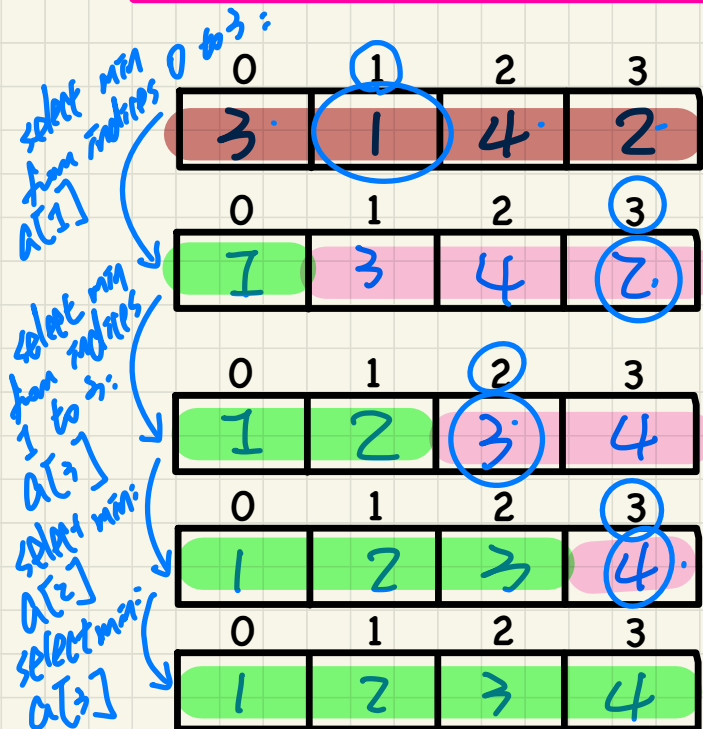
## Announcements

- **Written Test 1** guide released
  - + EECS account login (for WSC computers)
  - + PPY account + Duo Mobile (for eClass)
  - + Practice Questions & Review Session Survey
- **Assignment 1** due soon!
  - + Help: Scheduled Office Hours & TAs

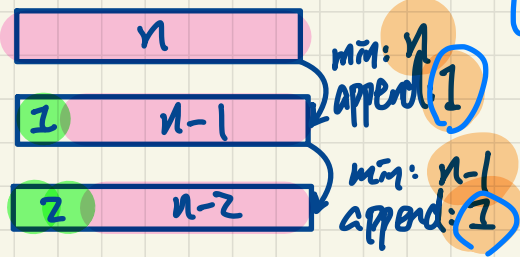
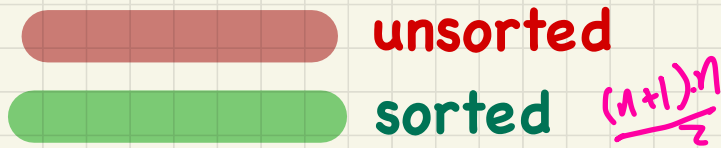
# Selection Sort

①  $n$  iterations (need to choose min  $n$  times)

Keep **selecting** minimum from the **unsorted** portion and appending it to the end of **sorted** portion.



(when started, sorted portion is empty)



$$O(n \cdot 1 + \frac{(n+(n-1)+\dots+1)}{2})$$

$$\# \text{ of appendings} = O(?) = O(n^2)$$



in-place sorting

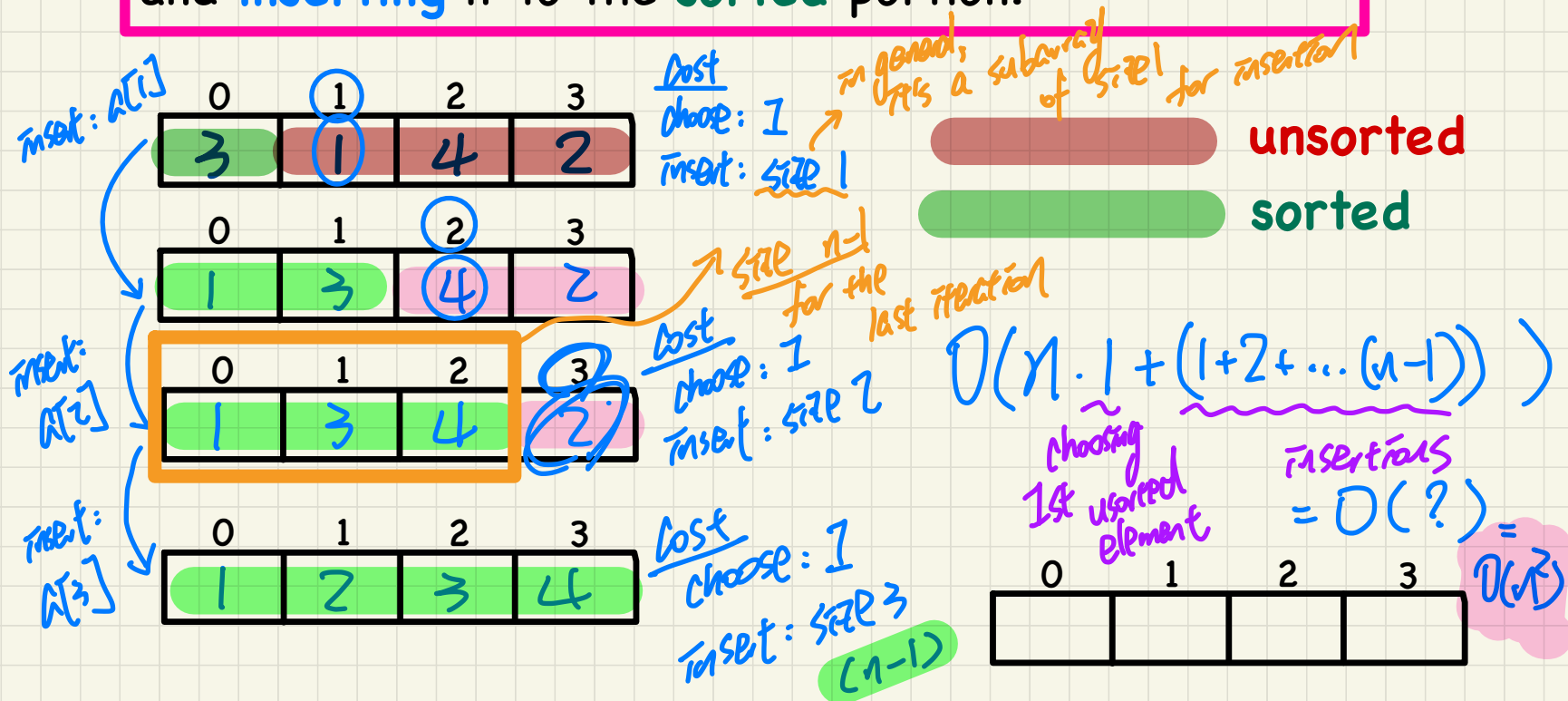
↳ sorting procedure operates

directly on the original input array.

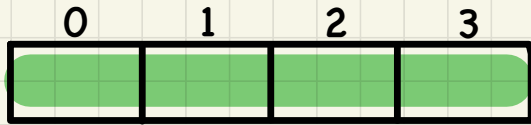
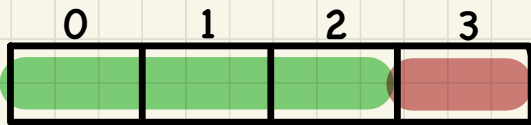
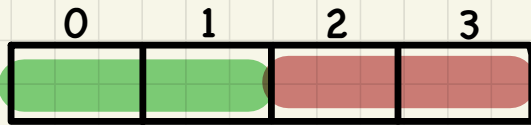
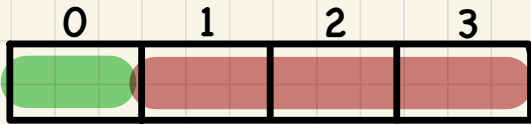
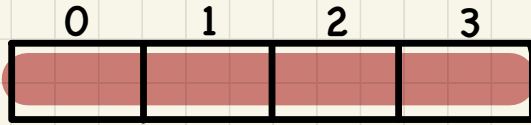
# Insertion Sort

# iterations (for choosing): n

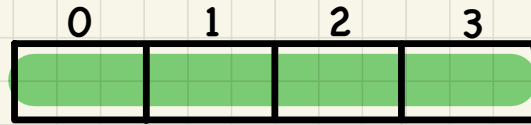
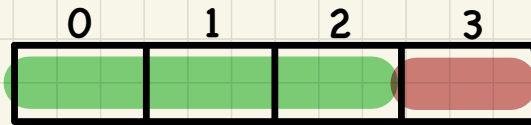
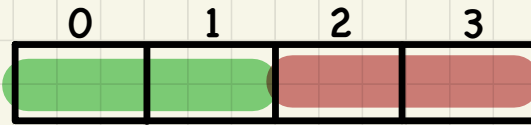
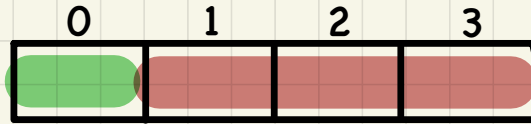
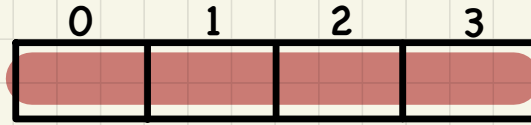
Keep getting 1st element from the **unsorted** portion and **inserting** it to the **sorted** portion.



## Selection Sort



## Insertion Sort



# Selection Sort: Deriving Asymptotic Upper Bound

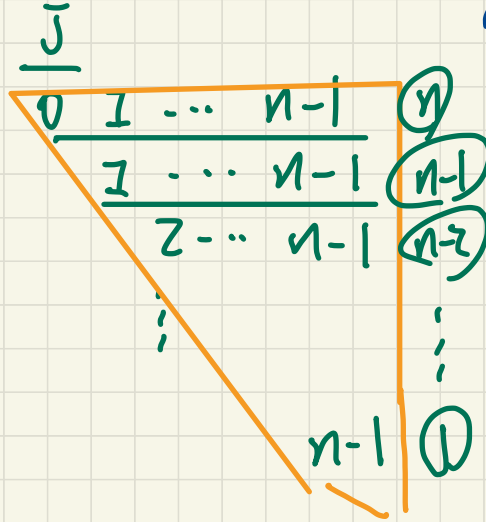
```

1 void selectionSort(int[] a, int n)
2   for (int i = 0; i <= (n - 2); i++)
3     int minIndex = i;
4     for (int j = i; j <= (n - 1); j++)
5       if (a[j] < a[minIndex]) { minIndex = j; }
6     int temp = a[i];
7     a[i] = a[minIndex];
8     a[minIndex] = temp;
  
```

$$\frac{(n+1) \cdot n}{2}$$

Annotations on code:  
 - Line 1:  $n$  is circled in purple, with a note  $\rightarrow a.length$ .  
 - Line 2:  $i = 0$  and  $i <= (n - 2)$  are highlighted in yellow.  
 - Line 3:  $int minIndex = i;$  is highlighted in purple.  
 - Line 4:  $for (int j = i; j <= (n - 1); j++)$  is highlighted in yellow.  
 - Line 5:  $if (a[j] < a[minIndex]) { minIndex = j; }$  is highlighted in blue, with a checkmark  $\checkmark$  to its right.  
 - Line 6:  $int temp = a[i];$  is highlighted in purple.  
 - Line 7:  $a[i] = a[minIndex];$  is highlighted in purple.  
 - Line 8:  $a[minIndex] = temp;$  is highlighted in purple.  
 - A purple  $I$  is written next to line 7.

Diagram showing the range of  $i$  values from 0 to  $n-2$ .  
 $[0, n-2]$   
 $= (n-2) - 0 + 1$   
 $= n-1$



$$O\left( \underbrace{(n-1)}_{\substack{\# \text{ of iterations} \\ \text{of outer} \\ \text{loop}}} \cdot \underbrace{I}_{\substack{\text{L4,} \\ \text{L6-8}}} + \underbrace{\frac{(n+(n-1)+\dots+1)}{2}}_{\substack{\# \text{ combinations} \\ \text{of } i, j}} \cdot \underbrace{J}_{\substack{\text{L5}}} \right)$$

$= O(?) = O(n^2)$

# Insertion Sort: Deriving Asymptotic Upper Bound

Exercise

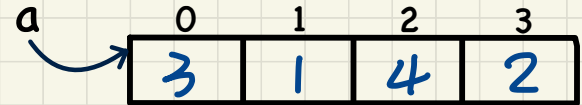
```
1 void insertionSort(int[] a, int n)
2   for (int i = 1; i < n; i++)
3     int current = a[i];
4     int j = i;
5     while (j > 0 && a[j - 1] > current)
6       a[j] = a[j - 1];
7       j--;
8     a[j] = current;
```



# Selection Sort in Java

```
1 void selectionSort(int[] a, int n)
2   for (int i = 0; i <= (n - 2); i++)
3     int minIndex = i;
4     for (int j = i; j <= (n - 1); j++)
5       if (a[j] < a[minIndex]) { minIndex = j; }
6     int temp = a[i];
7     a[i] = a[minIndex];
8     a[minIndex] = temp;
```

Inner Loop: select the next min from  $a[i]$  to  $a[n - 1]$  and put it to the end of the sorted region.



Outer Loop:

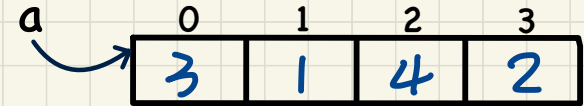
At the end of each iteration of the for-loop,  $a$  is sorted from  $a[0]$  to  $a[i]$ .

$i$	inner loop: $j$ from ? to ?	midIndex at L6	after L6 - L8, $a$ becomes?

# Insertion Sort in Java

```
1 void insertionSort(int[] a, int n)
2   for (int i = 1; i < n; i++)
3     int current = a[i];
4     int j = i;
5     while (j > 0 && a[j - 1] > current)
6       a[j] = a[j - 1];
7       j--;
8     a[j] = current;
```

Inner Loop: find out where to insert current into a[0] to a[i] s.t. that part of a becomes sorted.



Outer Loop:

At the end of each iteration of the for-loop, a is sorted from a[0] to a[i].

i	current after L3	j at L8	after L8, a becomes?

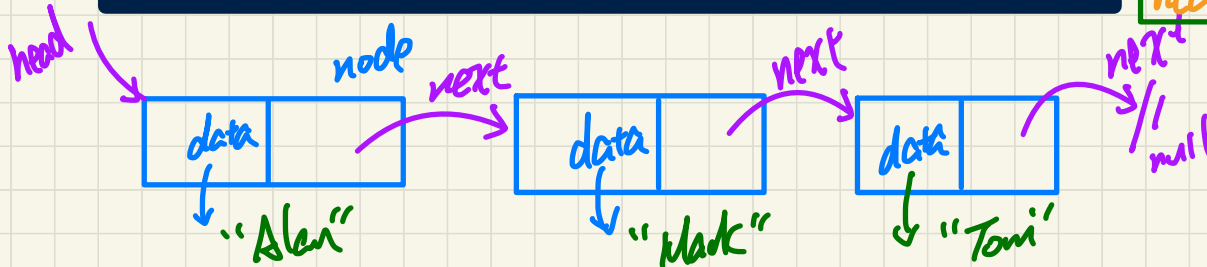
# Lecture

## Arrays vs. Linked Lists

### ***Singly-Linked Lists - Intuitive Introduction***

# Singly-Linked Lists (SLL): Visual Introduction

- A chain of connected nodes
- Each node contains:
  - + reference to a data object
  - + reference to the next node
- Accessing a node in a list:
  - \* Relative positioning:  $O(n)$
  - \* Absolute indexing:  $O(1)$
- The chain may grow or shrink dynamically.
- Head vs. Tail



head  $\neq$  null : 1st node

head.next  $\neq$  null : 2nd node

head.next.next  $\neq$  null : 3rd node

head.data : "Alan"

head.next.data : "Mark"

head.next.next.data : "Tom"

head.next.next.next (null)

head.next.next.next.data

null

NullPointerExcep