🎯 Today’s Learning Objectives

❑ Wrap-up Comparators

❑ Insertion Sort (pseudocode)

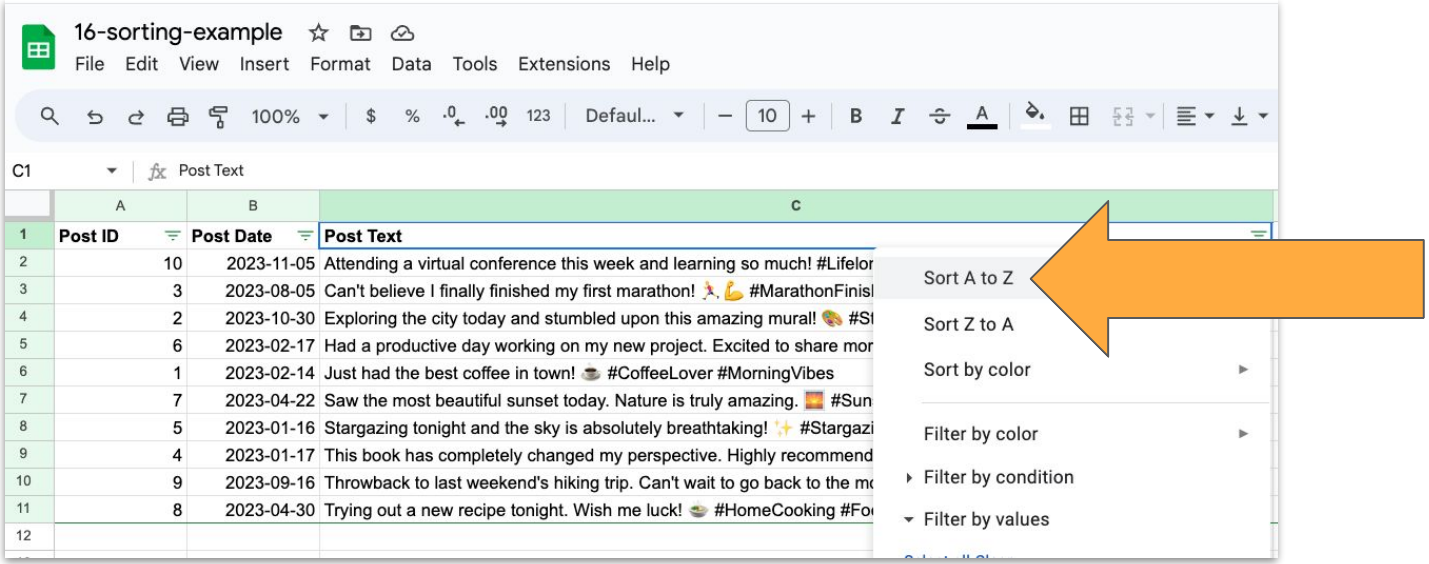
❑ Merge Sort

📚Readings

- Sedgewick and Wayne. Algorithms. Section 2.1
- Sedgewick and Wayne. Algorithms. Section 2.2.

Review: Sorting algorithms in modern applications

Using Google Sheets to sort (fake) X/Twitter posts by different variables/columns. Data here.



	Post ID	Post Date	Post Text
2	10	2023-11-05	Attending a virtual conference this week and learning so much! #Lifelon
3	3	2023-08-05	Can't believe I finally finished my first marathon! 🏃 💪 #MarathonFinisl
4	2	2023-10-30	Exploring the city today and stumbled upon this amazing mural! 🎨 #St
5	6	2023-02-17	Had a productive day working on my new project. Excited to share mor
6	1	2023-02-14	Just had the best coffee in town! ☕ #CoffeeLover #MorningVibes
7	7	2023-04-22	Saw the most beautiful sunset today. Nature is truly amazing. 🌅 #Sun
8	5	2023-01-16	Stargazing tonight and the sky is absolutely breathtaking! ✨ #Stargazi
9	4	2023-01-17	This book has completely changed my perspective. Highly recommend
10	9	2023-09-16	Throwback to last weekend's hiking trip. Can't wait to go back to the mo
11	8	2023-04-30	Trying out a new recipe tonight. Wish me luck! 🍲 #HomeCooking #Fo

Review: Selection Sort Pseudocode

Overview: We sort the array "in place."

We maintain a sorted part of the array (front of the array) and unsorted part (back of the array).

Outer loop:

Consider each element, array[i]

> **Inner loop:**
>
> For all elements with index j > i, find the smallest element
>
> If this smallest element is less than array[i], swap the elements.

Sorting Algorithms

Algorithm	**When?**	**Time Big-O?**	**(Auxiliary) Space Big-O?**
Selection Sort	Monday Lecture	$O(n^2)$	O(1)
Insertion Sort	Wednesday Lecture (pseudocode) Lab 5 (implement)		
Merge Sort	Wednesday Lecture		
Quick Sort	Friday Lecture (pseudocode) Lab 5 extension (implement)		

💻

SelectionSort.java

Review: Example 1 of Java’s Comparator interface

The **compare** method is required by the Comparator interface.

Returns:

A **positive integer** if the first argument is **greater than** the second argument

A **negative integer** if the first argument is **less than** the second argument

Zero if the first argument is **equal to** the second argument

Import statement

Interface

Reference type to compare

```
import java.util.Comparator;

class IntComparator implements Comparator<Integer>{

    public int compare(Integer number1, Integer number2){
       if (number1 > number2) return 1;
       if (number1 < number2) return -1;
       return 0;
    }
}
```

Java docs here.

Review: Example 2 of Java’s Comparator Interface

The **Comparator** interface is helpful when you need a different comparison logic that is not the “natural ordering” (e.g., 1 < 2).

```
import java.util.Comparator;

class ChronologicalOrder implements Comparator<Date>{

    public int compare(Date date1, Date date2){
        if (date1.year  < date2.year)  return -1;
        if (date1.year  > date2.year)  return +1;

        if (date1.month < date2.month) return -1;
        if (date1.month > date2.month) return +1;

        if (date1.day   < date2.day)   return -1;
        if (date1.day   > date2.day)   return +1;
        return 0;
    }
}
```

We will compare our custom **Date** class

💻

`Date.java`

`SelectionSort.java`

Example 3 of Java’s Comparator Interface

We can make several Comparator classes for the same reference type and choose which one to use with the **same implementation of the sorting algorithm.**

```
import java.util.Comparator;

// Sort by the day only
class DayOrder implements Comparator<Date>{
    public int compare(Date date1, Date date2){
        if (date1.day   < date2.day)   return -1;
        if (date1.day   > date2.day)   return +1;
        return 0;
    }
}
```

🎯 Today's Learning Objectives

- ✅ Wrap-up Comparators
- ❏ Insertion Sort (pseudocode)
- ❏ Merge Sort

Sorting Algorithms

Algorithm	**When?**	**Time Big-O?**	**(Auxiliary) Space Big-O?**
Selection Sort	Monday Lecture	O(n^2)	O(1)
Insertion Sort	Wednesday Lecture (pseudocode) **Lab 5 (implement)**		
Merge Sort	Wednesday Lecture		
Quick Sort	Friday Lecture (pseudocode) Lab 5 extension (implement)		

Insertion Sort

Card playing analogy:

Pick up a card from the unsorted section. Figure out where it goes in the sorted section and insert it.

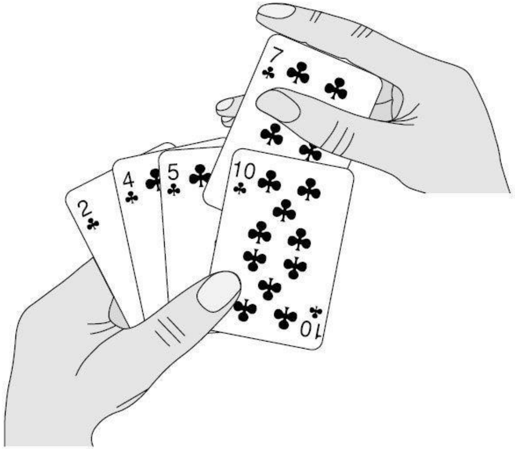


Figure credit: Rebekah Zhou

Insertion Sort: Pseudocode

Overview: We sort the array “in place.”

Maintain a “sorted” part of the array (beginning indices) and “unsorted” part (later indices).

```
Outer loop: Iterate through all indices in the array, from i=1  to i=array.length-1
    Assign  array[i] to a local variable temp. Let j=i.
    Inner loop: While j>0 and array[j-1] > temp
        Assign array[j]=array[j-1]
        Decrement j.
    Assign array[j] = temp.
```

💡 Think-pair-share

Follow the insertion sort pseudocode from the previous slide.

For the following array, use **pen & paper** to keep track of **i, j, temp** and the contents of the array

‘B’	‘D’	‘A’	‘C’

Big-O for Insertion sort

Auxiliary space:

- O(1)
- Because we're sorting in place.

Time:

- $O(n^2)$
- The algorithm has two nested loops.
- Worst case scenario (array in reverse order), every iteration of the inner loop will scan and shift the entire sorted subsection of the array before inserting the next element.

Sorting Algorithms

Algorithm	**When?**	**Time Big-O?**	**(Auxiliary) Space Big-O?**
Selection Sort	Monday Lecture	$O(n^2)$	O(1)
Insertion Sort	Wednesday Lecture (pseudocode) Lab 5 (implement)	$O(n^2)$	O(1)
Merge Sort	Wednesday Lecture		
Quick Sort	Friday Lecture (pseudocode) Lab 5 extension (implement)		

💡 Think-pair-share

Selection sort and insertion sort both have the same big-O for time and space.

Questions:

When would you choose to use insertion sort over selection sort?

Hint: Consider the following array

[2, 1, 3, 4, 5]

🎯 Today’s Learning Objectives

- ✅ Wrap-up Comparators
- ✅ Insertion Sort (pseudocode)
- ❏ Merge Sort

Merge Sort

Invented by John von Neumann in 1945.

Divide & Conquer

Merge sort uses the **divide and conquer** paradigm, a paradigm that is helpful for many different algorithms in computer science.

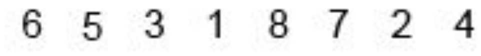
- **Divide:** Recursively break down problems into smaller, more manageable subproblems.
- **Conquer:** Combine the solutions to these subproblems to solve the original problem.

Merge Sort Pseudocode

- **Divide:** Recursively divide the unsorted array into two different arrays until each smaller array is a single element (base case).

- **Conquer:** Repeatedly merge smaller arrays to produce new sorted arrays.
    - To **merge**, compare the smallest element in the left array with the smallest element in the right array and repeat. Do so until there is only one large sorted array remaining.

Example



GIF credit: [Wikipedia]

Sorting Algorithms

Algorithm	**When?**	**Time Big-O?**	**(Auxiliary) Space Big-O?**
Selection Sort	Monday Lecture	$O(n^2)$	O(1)
Insertion Sort	Wednesday Lecture (pseudocode) Lab 5 (implement)	$O(n^2)$	O(1)
Merge Sort	Wednesday Lecture	O(n logn)	O(n)
Quick Sort	Friday Lecture (pseudocode) Lab 5 extension (implement)		

Preview: We'll do a proof sketch for this on Friday.

🎯 Today’s Learning Objectives

- ✅ Wrap-up Comparators
- ✅ Insertion Sort (pseudocode)
- ✅ Merge Sort