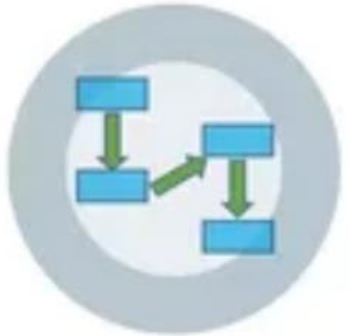
Sorting Algorithms



Data Structures and Algorithms – CSI 401

Arfan Shahzad

{ arfanskp@gmail.com }

Data Structures and Algorithms
Course Contents:
Abstract data types, complexity analysis, Big Oh notation, Stacks (linked lists and array implementations), Recursion and analyzing recursive algorithms, divide and conquer algorithms, Sorting algorithms (selection, insertion, merge, quick, bubble, heap, shell, radix, bucket), queue, dequeuer, priority queues (linked and array implementations of queues), linked list & its various types, sorted linked list, searching an unsorted array, binary search for sorted arrays, hashing and indexing, open addressing and chaining, trees and tree traversals, binary search trees, heaps, M-way tress, balanced trees, graphs, breadth-first and depth-first traversal, topological order, shortest path, adjacency matrix and adjacency list implementations, memory management and garbage collection

Sorting Algorithms

- A sorting algorithm is an algorithm that ***puts elements of an array in a certain order: ascending*** **(smallest to largest/ increasing)** or ***descending*** **(largest to smallest/ decreasing)**.
- Efficient sorting is important for optimizing the use of other algorithms, such as search and merge algorithms, which require input data to be in sorted form.

Sorting Algorithms cont…

- There are number of sorting algorithms.
- Some of them are: ***selection***, ***insertion***, ***merge***, ***quick***, ***bubble***, ***heap***, ***shell***, ***radix***, ***bucket*** sort etc.

Sorting Algorithms cont…
In-Place & Not In-Place Sorting

- Sorting algorithms may require some ***extra space for comparison*** and ***temporary storage*** of few data elements.
- Due to this characteristics, sorting algorithms are divided into types:
- **In-Place** Sorting Algorithms
- **Not In-Place** Sorting Algorithms (Out of place sorting algorithms)

Sorting Algorithms cont…
In-Place & Not In-Place Sorting

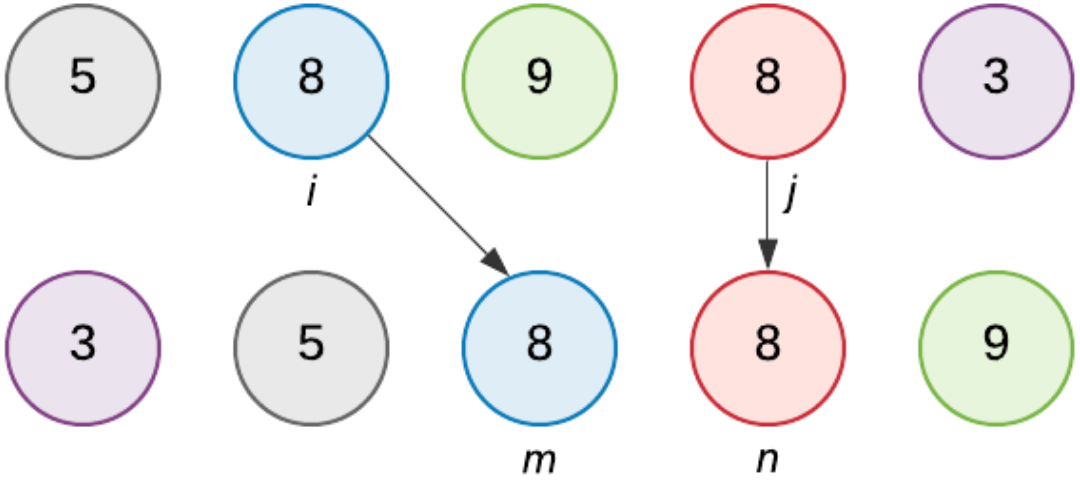
- An in-place algorithm is an algorithm that does not need an extra space and produces an output in the same memory that contains the data by transforming the input “in-place”.
- Insertion sort, Selection sort, Quick sort, Bubble sort, Heap sort, etc. are examples of in-place sorting.

Sorting Algorithms cont...
In-Place & Not In-Place Sorting

- An algorithm which is not in-place is called not-in-place or out-of-place algorithm.
- The standard merge sort algorithm is an example of out-of-place algorithm as it requires O(n) extra space for merging. The merging can be done in-place but it increases the time complexity of the sorting routine.

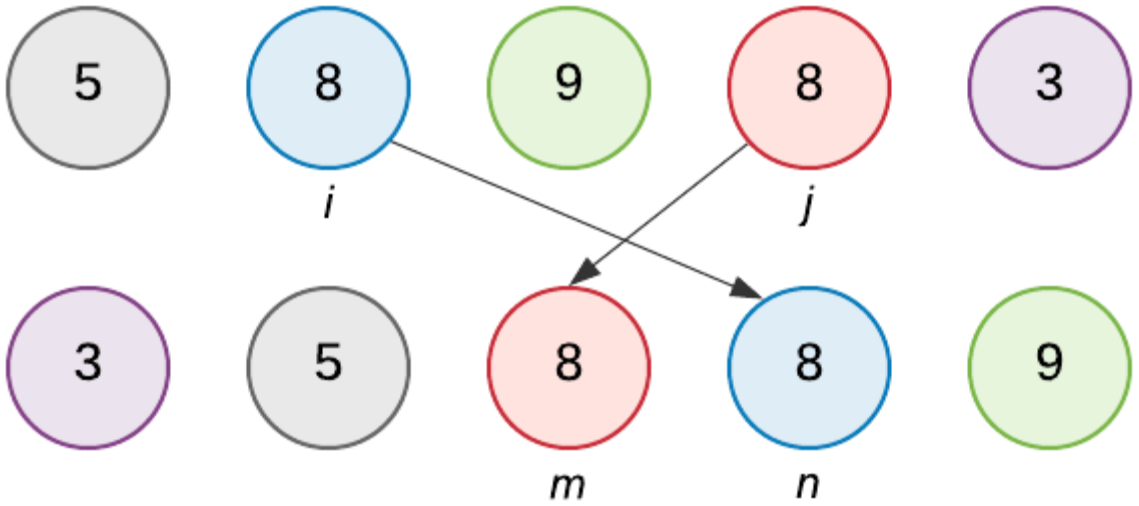
Sorting Algorithms cont...
Stable and Not Stable Sorting

- If a sorting algorithm, after sorting the contents, does not change the sequence of similar content in which they appear, it is called **stable sorting**.

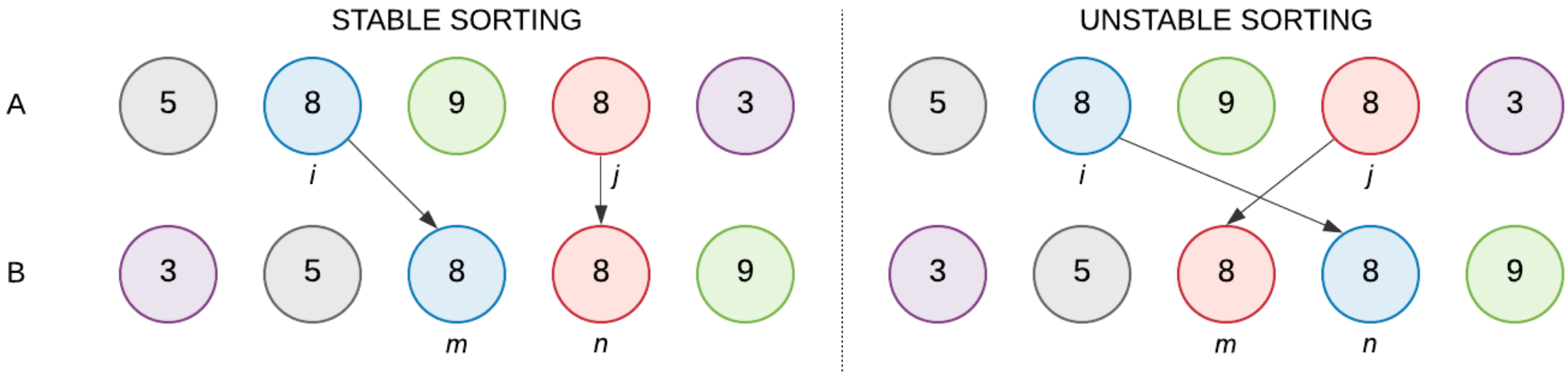


Sorting Algorithms cont...
Stable and Not Stable Sorting

- If a sorting algorithm, after sorting the contents, changes the sequence of similar content in which they appear, it is called **unstable sorting**.



Sorting Algorithms cont...
Stable and Not Stable Sorting



Sorting Algorithms cont…
Adaptive and Non-Adaptive Sorting

- A sorting algorithm is said to be ***adaptive***, if it takes advantage of already “sorted” elements in the list that is to be sorted.
- That is, while sorting if the source list has some element already sorted, adaptive algorithms will take this into account and will try not to re-order them.

Sorting Algorithms cont…
Adaptive and Non-Adaptive Sorting

- A non-adaptive algorithm is one which does not take into account the elements which are already sorted.
- They try to force every single element to be re-ordered to confirm their sortedness.

Sorting Algorithms cont...
Selection Sort

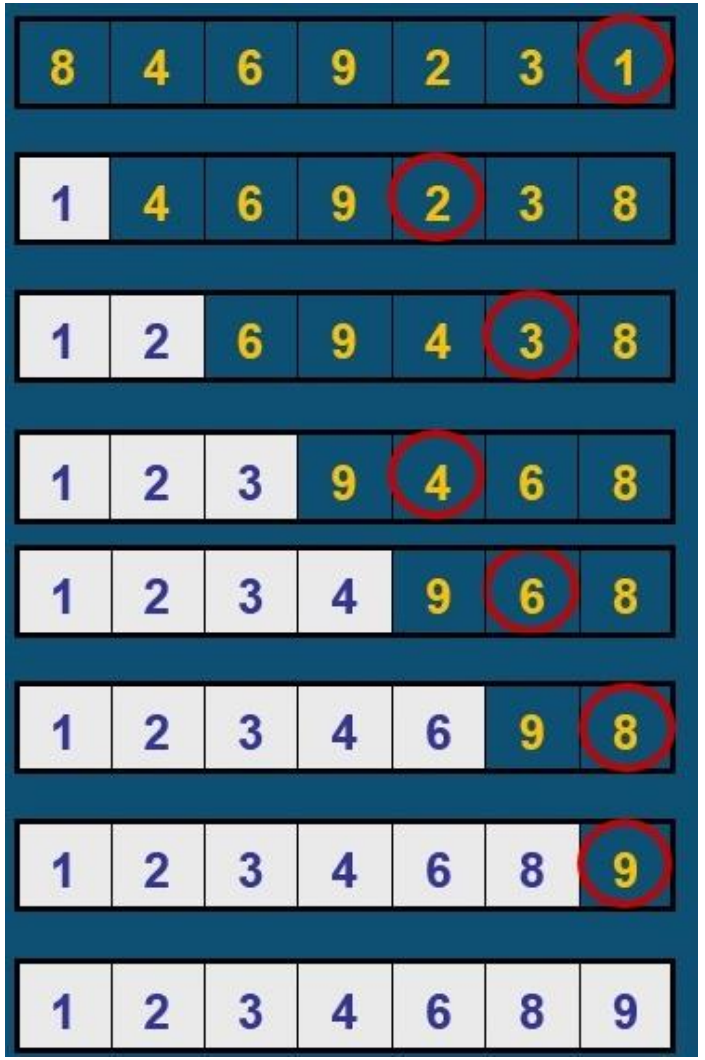
- Selection sort is an in-place comparison-based algorithm in which the list (array) is divided into two parts, the sorted part at the left end and the unsorted part at the right end.
- Initially, the sorted part is empty and the unsorted part is the entire list.

Sorting Algorithms cont...
Selection Sort

- The smallest element is selected from the unsorted array and swapped with the leftmost element, and that element becomes a part of the sorted array.
- This process continues moving unsorted array boundary by one element to the right.

Sorting Algorithms cont…
Selection Sort

- This algorithm is not suitable for large data sets as its average and worst case complexities are of $O(n^2)$, where n is the number of items.



Sorting Algorithms cont…
Selection Sort

- **Selection sort's**, working is given below:

1. Find the smallest value. Swap it with the first value (at index 0).
2. Find second-smallest value. Swap it with the second value (index 1).
3. Find the third-smallest, swap it with the third value (at index 2).
4. Repeat finding the next-smallest value, and swapping it into the correct position until the array is sorted.

Sorting Algorithms cont...
Selection Sort

```
procedure selection sort
   list  : array of items
   n     : size of list

   for i = 1 to n - 1
   /* set current element as minimum*/
      min = i

      /* check the element to be minimum */

      for j = i+1 to n
         if list[j] < list[min] then
            min = j;
         end if
      end for

      /* swap the minimum element with the current element*/
      if indexMin != i  then
         swap list[min] and list[i]
      end if
   end for

end procedure
```

Sorting Algorithms cont...
Selection Sort

```
for(i=0;i<n-1;i++)
{
    min=a[i];
    loc=i;
    for(j=i+1;j<n;j++)
    {
        if(min>a[j])
        {   min=a[j];      loc=j;
        }
    }
    //Swapping
    temp=a[i];    a[i]=a[loc];      a[loc]=temp;
}
```

Sorting Algorithms cont...
Selection Sort

Algorithm	Time Complexity			Space Complexity
	Best	Average	Worst	Worst
Quicksort	Ω(n log(n))	Θ(n log(n))	O(n^2)	O(log(n))
Mergesort	Ω(n log(n))	Θ(n log(n))	O(n log(n))	O(n)
Timsort	Ω(n)	Θ(n log(n))	O(n log(n))	O(n)
Heapsort	Ω(n log(n))	Θ(n log(n))	O(n log(n))	O(1)
Bubble Sort	Ω(n)	Θ(n^2)	O(n^2)	O(1)
Insertion Sort	Ω(n)	Θ(n^2)	O(n^2)	O(1)
Selection Sort	Ω(n^2)	Θ(n^2)	O(n^2)	O(1)
Tree Sort	Ω(n log(n))	Θ(n log(n))	O(n^2)	O(n)
Shell Sort	Ω(n log(n))	Θ(n(log(n))^2)	O(n(log(n))^2)	O(1)
Bucket Sort	Ω(n+k)	Θ(n+k)	O(n^2)	O(n)
Radix Sort	Ω(nk)	Θ(nk)	O(nk)	O(n+k)
Counting Sort	Ω(n+k)	Θ(n+k)	O(n+k)	O(k)
Cubesort	Ω(n)	Θ(n log(n))	O(n log(n))	O(n)

Sorting Algorithms cont...
Insertion Sort

- This is an in-place comparison-based sorting algorithm.
- Here, a sub-list is maintained which is always sorted.
- For example, the lower part of an array is maintained to be sorted.
- An element which is to be “inserted” in this sorted sub-list, has to find its appropriate place and then it has to be inserted there.
- Hence the name, **insertion sort**.

Sorting Algorithms cont...
Insertion Sort

- The array is searched sequentially and unsorted items are moved and inserted into the sorted sub-list (in the same array).
- This algorithm is not suitable for large data sets as its average and worst case complexity are of $O(n^2)$, where **n** is the number of items.

Sorting Algorithms cont...
Insertion Sort

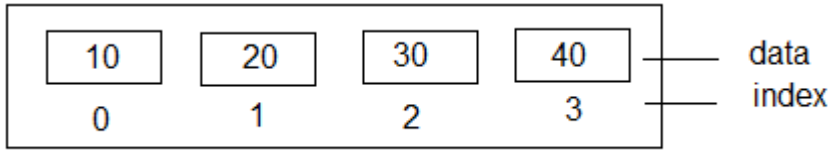
6 5 3 1 8 7 2 4

Sorting Algorithms cont...
Insertion Sort

- Simple steps of insertion sort are given below:

1. If it is the first element, it is already sorted. return 1;
2. Pick next element
3. Compare with all elements in the sorted sub-list
4. Shift all the elements in the sorted sub-list that is greater than the value to be sorted
5. Insert the value
6. Repeat until list is sorted

Sorting Algorithms cont...
Insertion Sort




```
procedure insertionSort( A : array of items )
   int holePosition
   int valueToInsert

   for i = 1 to length(A) inclusive do:

      /* select value to be inserted */
      valueToInsert = A[i]
      holePosition = i

      /*locate hole position for the element to be inserted */

      while holePosition > 0 and A[holePosition-1] > valueToInsert do:
         A[holePosition] = A[holePosition-1]
         holePosition = holePosition -1
      end while

      /* insert the number at hole position */
      A[holePosition] = valueToInsert

   end for

end procedure
```

Sorting Algorithms cont...
Insertion Sort

Algorithm	Time Complexity			Space Complexity
	Best	Average	Worst	Worst
Quicksort	Ω(n log(n))	Θ(n log(n))	O(n^2)	O(log(n))
Mergesort	Ω(n log(n))	Θ(n log(n))	O(n log(n))	O(n)
Timsort	Ω(n)	Θ(n log(n))	O(n log(n))	O(n)
Heapsort	Ω(n log(n))	Θ(n log(n))	O(n log(n))	O(1)
Bubble Sort	Ω(n)	Θ(n^2)	O(n^2)	O(1)
Insertion Sort	Ω(n)	Θ(n^2)	O(n^2)	O(1)
Selection Sort	Ω(n^2)	Θ(n^2)	O(n^2)	O(1)
Tree Sort	Ω(n log(n))	Θ(n log(n))	O(n^2)	O(n)
Shell Sort	Ω(n log(n))	Θ(n(log(n))^2)	O(n(log(n))^2)	O(1)
Bucket Sort	Ω(n+k)	Θ(n+k)	O(n^2)	O(n)
Radix Sort	Ω(nk)	Θ(nk)	O(nk)	O(n+k)
Counting Sort	Ω(n+k)	Θ(n+k)	O(n+k)	O(k)
Cubesort	Ω(n)	Θ(n log(n))	O(n log(n))	O(n)

Sorting Algorithms cont...
Merge Sort

- Merge sort is a sorting technique based on divide and conquer technique.
- With worst-case time complexity being O(n log n), it is one of the most respected algorithms.
- Merge sort first divides the array into equal halves and then combines them in a sorted manner.

Sorting Algorithms cont...
Merge Sort

6 5 3 1 8 7 2 4

Sorting Algorithms cont...
Merge Sort

- How merge sort takes place, described here:

1. If it is only one element in the list it is already sorted, return.
2. Divide the list recursively into two halves until it can no more be divided.
3. Merge the smaller lists into new list in sorted order.

Sorting Algorithms cont…
Merge Sort

Algorithm	Time Complexity			Space Complexity
	Best	Average	Worst	Worst
Quicksort	Ω(n log(n))	Θ(n log(n))	O(n^2)	O(log(n))
Mergesort	Ω(n log(n))	Θ(n log(n))	O(n log(n))	O(n)
Timsort	Ω(n)	Θ(n log(n))	O(n log(n))	O(n)
Heapsort	Ω(n log(n))	Θ(n log(n))	O(n log(n))	O(1)
Bubble Sort	Ω(n)	Θ(n^2)	O(n^2)	O(1)
Insertion Sort	Ω(n)	Θ(n^2)	O(n^2)	O(1)
Selection Sort	Ω(n^2)	Θ(n^2)	O(n^2)	O(1)
Tree Sort	Ω(n log(n))	Θ(n log(n))	O(n^2)	O(n)
Shell Sort	Ω(n log(n))	Θ(n(log(n))^2)	O(n(log(n))^2)	O(1)
Bucket Sort	Ω(n+k)	Θ(n+k)	O(n^2)	O(n)
Radix Sort	Ω(nk)	Θ(nk)	O(nk)	O(n+k)
Counting Sort	Ω(n+k)	Θ(n+k)	O(n+k)	O(k)
Cubesort	Ω(n)	Θ(n log(n))	O(n log(n))	O(n)

Sorting Algorithms cont...
Quick Sort

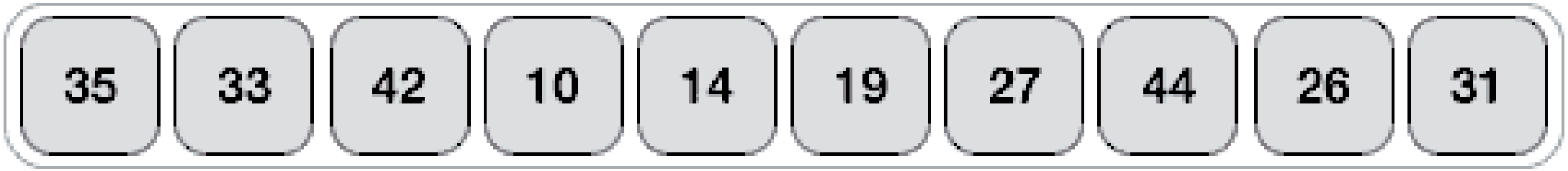
- Quick sort is a highly efficient sorting algorithm and is based on partitioning of array of data into smaller arrays.
- A large array is partitioned into two arrays one of which holds values smaller than the specified value, say pivot, based on which the partition is made and another array holds values greater than the pivot value.

Sorting Algorithms cont...
Quick Sort

- Quicksort partitions an array and then calls itself recursively twice to sort the two resulting subarrays.
- This algorithm is quite efficient for large-sized data sets as its average and worst-case complexity are $O(n^2)$, respectively.

Sorting Algorithms cont...
Quick Sort

Unsorted Array



Sorting Algorithms cont...
Quick Sort

6 5 3 1 8 7 2 4

Sorting Algorithms cont...
Quick Sort

Algorithm	Time Complexity			Space Complexity
	Best	Average	Worst	Worst
Quicksort	Ω(n log(n))	Θ(n log(n))	O(n^2)	O(log(n))
Mergesort	Ω(n log(n))	Θ(n log(n))	O(n log(n))	O(n)
Timsort	Ω(n)	Θ(n log(n))	O(n log(n))	O(n)
Heapsort	Ω(n log(n))	Θ(n log(n))	O(n log(n))	O(1)
Bubble Sort	Ω(n)	Θ(n^2)	O(n^2)	O(1)
Insertion Sort	Ω(n)	Θ(n^2)	O(n^2)	O(1)
Selection Sort	Ω(n^2)	Θ(n^2)	O(n^2)	O(1)
Tree Sort	Ω(n log(n))	Θ(n log(n))	O(n^2)	O(n)
Shell Sort	Ω(n log(n))	Θ(n(log(n))^2)	O(n(log(n))^2)	O(1)
Bucket Sort	Ω(n+k)	Θ(n+k)	O(n^2)	O(n)
Radix Sort	Ω(nk)	Θ(nk)	O(nk)	O(n+k)
Counting Sort	Ω(n+k)	Θ(n+k)	O(n+k)	O(k)
Cubesort	Ω(n)	Θ(n log(n))	O(n log(n))	O(n)

Sorting Algorithms cont...
Bubble Sort

- Bubble sort is comparison-based algorithm in which each pair of adjacent elements is compared and the elements are swapped if they are not in order.
- It sometimes referred to as **sinking sort**
- This algorithm is not suitable for large data sets as its worst case complexity is of $O(n^2)$ where **n** is the number of items.

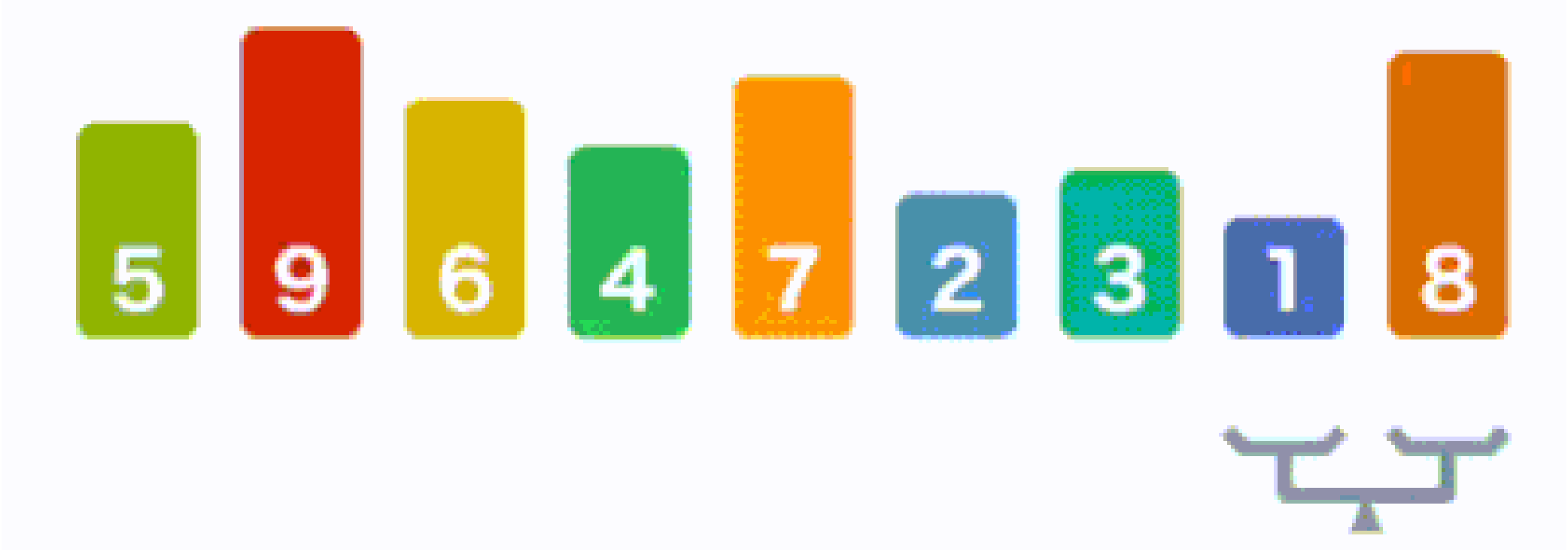
Sorting Algorithms cont...
Bubble Sort

- Although the algorithm is simple, it is too slow and impractical for most problems even when compared to **insertion sort**.
- It can be practical if the input is in sorted order but may have some out-of-order elements nearly in position.

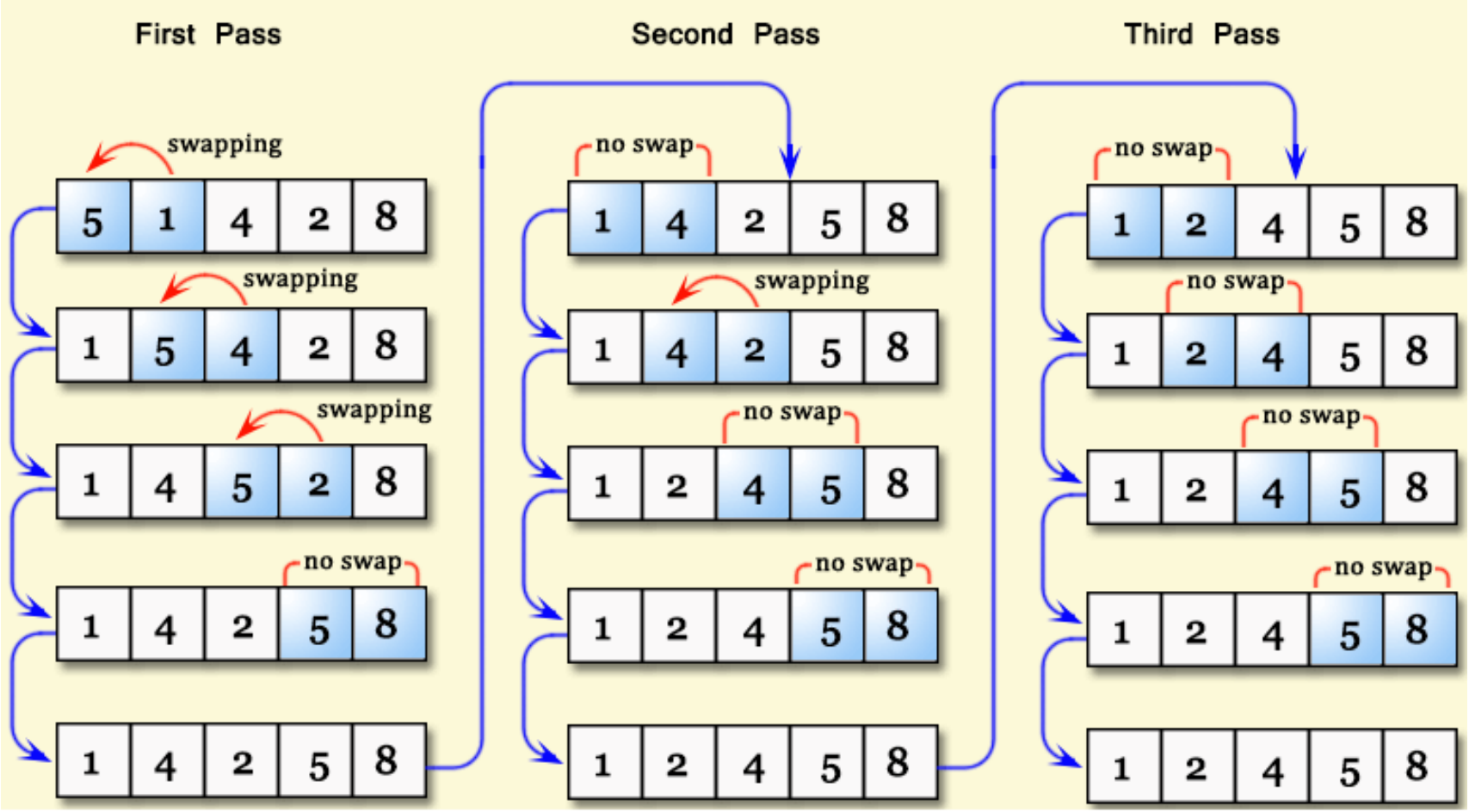
Sorting Algorithms cont...
Bubble Sort

6 5 3 1 8 7 2 4

Sorting Algorithms cont…
Bubble Sort



Sorting Algorithms cont...
Bubble Sort



Sorting Algorithms cont...
Bubble Sort

Algorithm	Time Complexity			Space Complexity
	Best	Average	Worst	Worst
Quicksort	Ω(n log(n))	Θ(n log(n))	O(n^2)	O(log(n))
Mergesort	Ω(n log(n))	Θ(n log(n))	O(n log(n))	O(n)
Timsort	Ω(n)	Θ(n log(n))	O(n log(n))	O(n)
Heapsort	Ω(n log(n))	Θ(n log(n))	O(n log(n))	O(1)
Bubble Sort	Ω(n)	Θ(n^2)	O(n^2)	O(1)
Insertion Sort	Ω(n)	Θ(n^2)	O(n^2)	O(1)
Selection Sort	Ω(n^2)	Θ(n^2)	O(n^2)	O(1)
Tree Sort	Ω(n log(n))	Θ(n log(n))	O(n^2)	O(n)
Shell Sort	Ω(n log(n))	Θ(n(log(n))^2)	O(n(log(n))^2)	O(1)
Bucket Sort	Ω(n+k)	Θ(n+k)	O(n^2)	O(n)
Radix Sort	Ω(nk)	Θ(nk)	O(nk)	O(n+k)
Counting Sort	Ω(n+k)	Θ(n+k)	O(n+k)	O(k)
Cubesort	Ω(n)	Θ(n log(n))	O(n log(n))	O(n)

Sorting Algorithms cont...
Heap Sort

- Heap sort is a comparison-based sorting technique based on Binary Heap data structure.
- It is similar to selection sort where we first find the minimum element and place the minimum element at the beginning.
- We repeat the same process for the remaining elements.

Sorting Algorithms cont…
Heap Sort (Complete Binary Tree & Binary Heap)

- A complete binary tree is a binary tree in which every level, except possibly the last, is completely filled, and all nodes are as far left as possible
- A Binary Heap is a Complete Binary Tree where items are stored in a special order such that the value in a parent node is greater (max heap) or smaller (min heap) than the values in its two children nodes.

Sorting Algorithms cont...
Heap Sort (Heapify)

- The process of reshaping a binary tree into a Heap data structure is known as ‘heapify’.
- A binary tree is a tree data structure that has two child nodes at max.
- If a node’s children nodes are ‘heapified’, then only ‘heapify’ process can be applied over that node.
- A heap should always be a complete binary tree.

Sorting Algorithms cont...
Heap Sort (Heapify)

- Starting from a complete binary tree, we can modify it to become a Max-Heap by running a function called ‘heapify’ on all the non-leaf elements of the heap. i.e. ‘heapify’ uses recursion.

Sorting Algorithms cont...
Heap Sort (Algorithm increasing order)

1. Build a max heap from the input data.
2. At this point, the largest item is stored at the root of the heap. Replace it with the last item of the heap followed by reducing the size of heap by 1. Finally, heapify the root of the tree.
3. Repeat step 2 while the size of the heap is greater than 1.

Sorting Algorithms cont...
Heap Sort

Algorithm	Time Complexity			Space Complexity
	Best	Average	Worst	Worst
Quicksort	Ω(n log(n))	Θ(n log(n))	O(n^2)	O(log(n))
Mergesort	Ω(n log(n))	Θ(n log(n))	O(n log(n))	O(n)
Timsort	Ω(n)	Θ(n log(n))	O(n log(n))	O(n)
Heapsort	Ω(n log(n))	Θ(n log(n))	O(n log(n))	O(1)
Bubble Sort	Ω(n)	Θ(n^2)	O(n^2)	O(1)
Insertion Sort	Ω(n)	Θ(n^2)	O(n^2)	O(1)
Selection Sort	Ω(n^2)	Θ(n^2)	O(n^2)	O(1)
Tree Sort	Ω(n log(n))	Θ(n log(n))	O(n^2)	O(n)
Shell Sort	Ω(n log(n))	Θ(n(log(n))^2)	O(n(log(n))^2)	O(1)
Bucket Sort	Ω(n+k)	Θ(n+k)	O(n^2)	O(n)
Radix Sort	Ω(nk)	Θ(nk)	O(nk)	O(n+k)
Counting Sort	Ω(n+k)	Θ(n+k)	O(n+k)	O(k)
Cubesort	Ω(n)	Θ(n log(n))	O(n log(n))	O(n)