Tutorial 05: Sorting

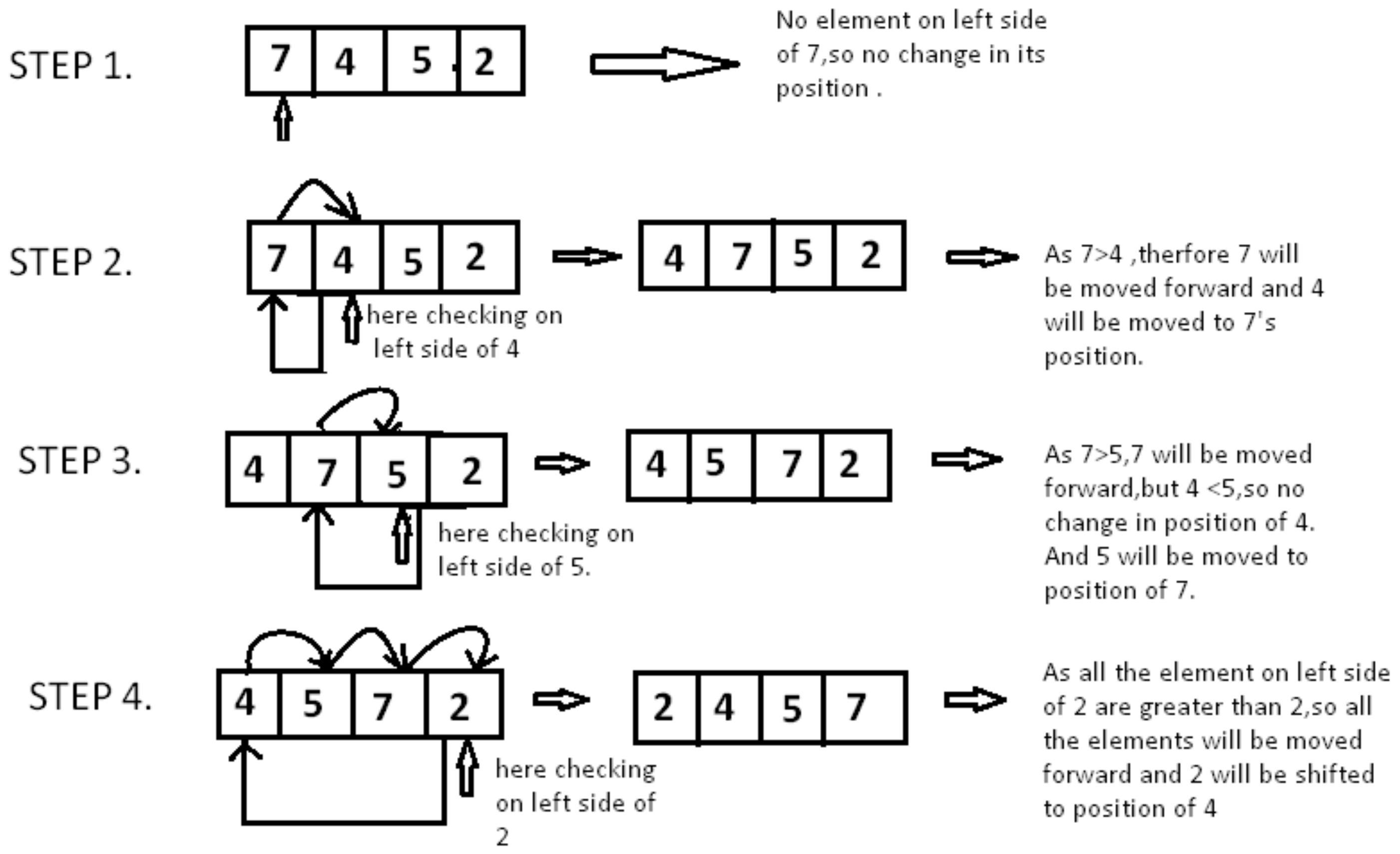
CSCI2520 - DATA STRUCTURES AND APPLICATIONS

TUTOR: ZHANG KAI

Outlines

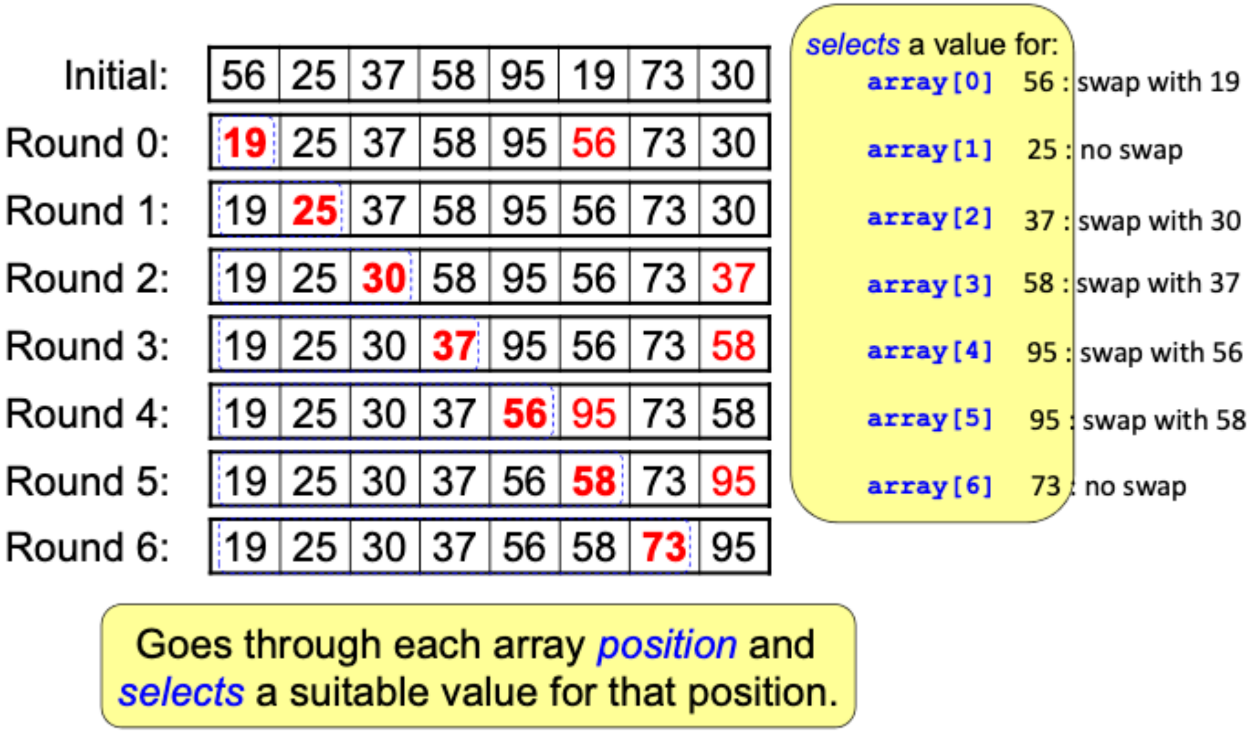
1. Several Sorting Algorithms
   - Insertion Sort
   - Selection Sort
   - Merge Sort
   - Quicksort
   - Counting Sort
   - Radix Sort
2. Exercises

Insertion Sort $O(N^2)$



Selection Sort $O(N^2)$

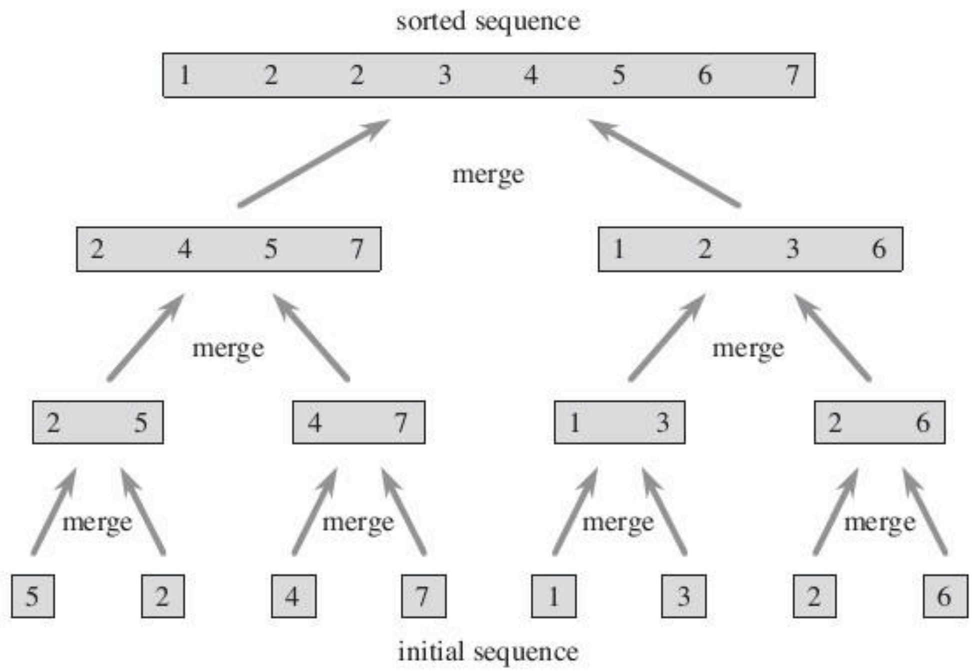
The algorithm goes through each array **position** and **selects** a suitable value for that position.



Merge Sort *O(N logN)*

Divide-and-Conquer

- Divide the array into two (or more) subarrays
- Sort each subarray (Conquer)
- Merge them into one (in a smart way!)



Pseudo Code:

```
MERGE_SORT(A, p, r)
    if p < r
        q = (p + r) / 2
        MERGE_SORT(A, p, q)
        MERGE_SORT(A, q + 1, r)
        MERGE(A, p, q, r)
```

Quicksort *O(N logN)*

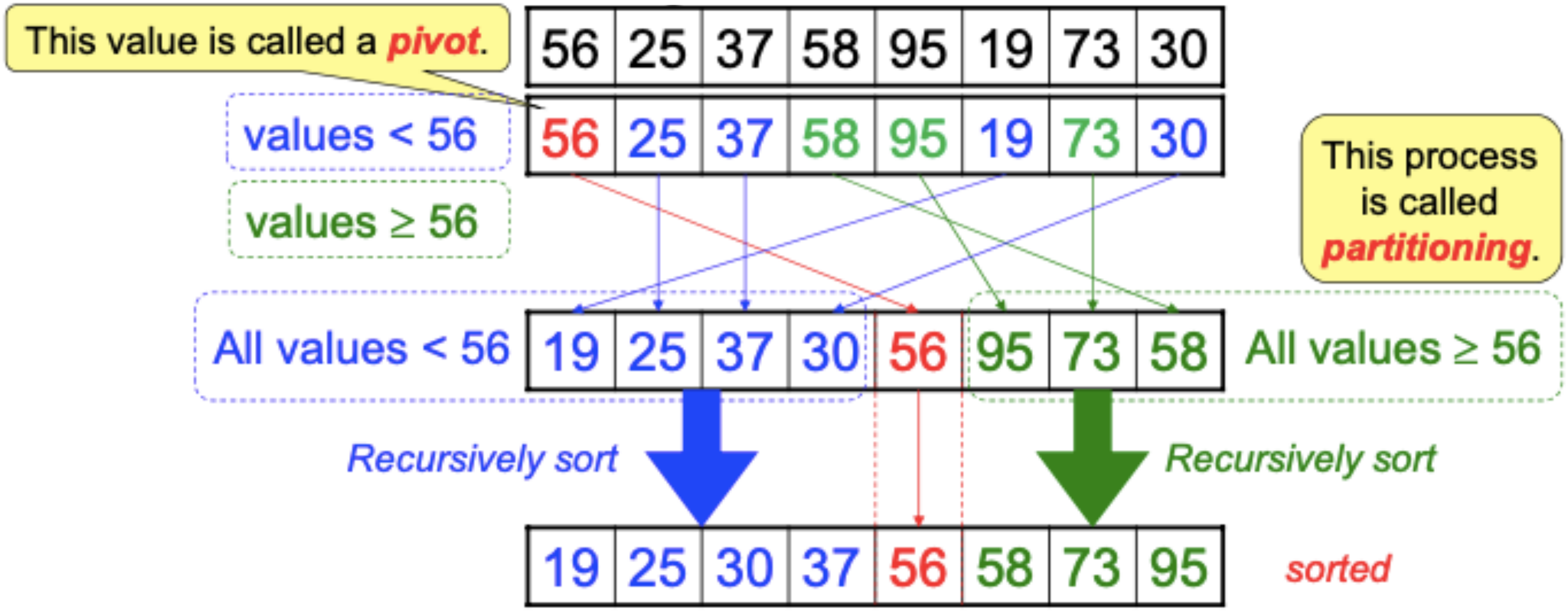
Like Merge Sort, QuickSort is a Divide and Conquer algorithm. It picks an element as pivot and partitions the given array around the picked pivot.

Any array element can be chosen as the pivot.

```
/* low  --> Starting index,  high  --> Ending index */
quickSort(arr[], low, high)
{
    if (low < high)
    {
        /* pi is partitioning index, arr[pi] is now
           at right place */
        pi = partition(arr, low, high);

        quickSort(arr, low, pi - 1);  // Before pi
        quickSort(arr, pi + 1, high); // After pi
    }
}
```

Quicksort   *O(N logN)*



Counting Sort *O(N + K)*

Counting sort is an algorithm for sorting a collection of objects according to keys that are small integers.

Pseudocode:

```
count = array of k+1 zeros
for x in input do
    count[key(x)] += 1

total = 0
for i in 0, 1, ... k do
    count[i], total = total, count[i] + total

output = array of the same length as input
for x in input do
    output[count[key(x)]] = x
    count[key(x)] += 1

return output
```

Radix Sort *O(D(N + K))*

Given N D-digits number in which each digit can take on up to K possible values, radix sort correctly sorts these numbers in O(D(N + K)).

- There are D counting sorts which each takes O(N + K).

Original	Digit 0	Digit 1	Digit 2
329	720	720	329
457	355	329	355
657	436	436	436
839	457	839	457
436	657	355	657
720	329	457	720
355	839	657	839

Exercise 1

Which of the following sorting algorithms in its typical implementation gives best performance when applied on an array which is sorted or almost sorted (maximum 1 or two elements are misplaced).

- A. Quicksort
- B. Merge Sort
- C. Insertion Sort

Insertion sort takes linear time when input array is sorted or almost sorted (maximum 1 or 2 elements are misplaced). All other sorting algorithms mentioned above will take more than lienear time in their typical implementation.

Exercise 2

Suppose we are sorting an array of eight integers using quicksort, and we have just finished the first partitioning with the array looking like this:

2	5	1	7	9	12	11	10

- A. The pivot could be either the 7 or the 9.
- B. The pivot could be the 7, but it is not the 9.
- C. The pivot is not the 7, but it could be the 9.
- D. Neither the 7 nor the 9 is the pivot.

7 and 9 both are at their correct positions (as in a sorted array). Also, all elements on left of 7 and 9 are smaller than 7 and 9 respectively and on right are greater than 7 and 9 respectively.

Exercise 3

Given an array where numbers are in range from 1 to n^6, which sorting algorithm can be used to sort these number in linear time?

- A. Not possible to sort in linear time.
- B. Radix Sort.
- C. Counting Sort.
- D. Quick Sort.

Exercise 4

Randomized quicksort is an extension of quicksort where the pivot is chosen randomly. What is the worst case complexity of sorting n numbers using randomized quicksort?

- A.O(n)
- B. O(n log n)
- C. $O(n^2)$
- D. O(n!)

Randomized quicksort has expected time complexity as O(nLogn), but worst case time complexity remains same. In worst case the randomized function can pick the index of corner element every time.

Exercise 5

Assume that a mergesort algorithm in the worst case takes 30 seconds for an input of size 64. Which of the following most closely approximates the maximum input size of a problem that can be solved in 6 minutes?

- A.256
- B. 512
- C. 1024
- D. 2048

Exercise 5

Assume that a mergesort algori
input of size 64. Which of the fo num
input size of a problem that can

- A.256
- B. 512
- C. 1024
- D. 2048

```
Time complexity of merge sort is Θ(nLogn)

c*64Log64 is 30
c*64*6 is 30
c is 5/64

For time 6 minutes

5/64*nLogn = 6*60

nLogn = 72*64 = 512 * 9

n = 512.
```

Exercise 6

Which is the correct order of the following algorithms with respect to their time Complexity in the best case ?

- A. Merge sort > Quick sort >Insertion sort > selection sort
- B. Insertion sort < Quick sort < Merge sort < selection sort
- C. Merge sort > selection sort > quick sort > insertion sort
- D. Merge sort > Quick sort > selection sort > insertion sort

In best case Quick sort: O (nlogn)
Merge sort: O (nlogn)
Insertion sort: O (n)
Selection sort: O (n^2)

Exercise 7

Which one of the following in place sorting algorithms needs the minimum number of swaps?

- A. Quick sort
- B. Insertion sort
- C. Selection sort

Selection sort takes minimum number of swaps to sort an array. It takes maximum of O(n) comparisons to sort an array with n elements.