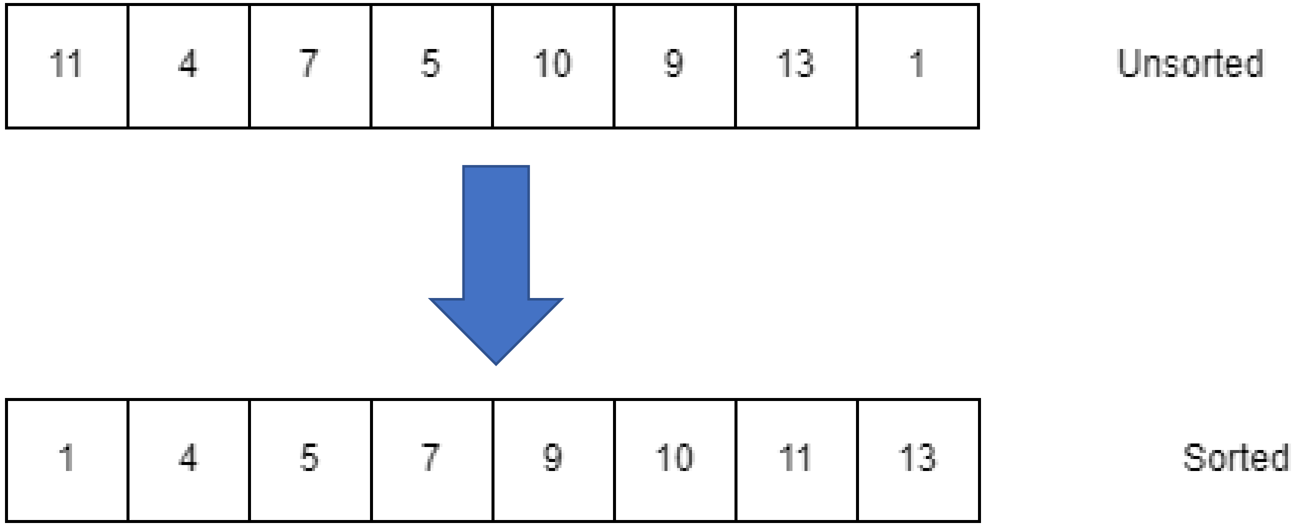
SCHOOL OF
ENGINEERING
BANGKOK UNIVERSITY

Sorting

AIE 311 : Data structure and Algorithm

Sorting

- Sorting => การเรียงลำดับข้อมูล



11	4	7	5	10	9	13	1	Unsorted
1	4	5	7	9	10	11	13	Sorted

Diagram of a Tree Data Structure

Ref: https://medium.com/@verdi/working-with-trees-2083739b8918

Sorting

- **ประเภทของการ Sorting**

1. Comparison sort

   การเรียงลำดับโดยการเปรียบเทียบค่าหรือองค์ประกอบต่าง ๆ ระหว่างข้อมูล

2. Non-comparison sort

   การเรียงลำดับโดยไม่มีการเปรียบเทียบค่าหรือองค์ประกอบต่าง ๆ ระหว่างข้อมูล แต่ใช้คุณสมบัติโดยตรงของข้อมูล เช่น การใช้ค่าความถี่ของข้อมูล

Sorting

- ประเภทของการ Sorting

1. Comparison sort
   - Bubble Sort
   - Selection Sort
   - Insertion Sort
   - Merge Sort
   - Quick Sort
   - Heap Sort

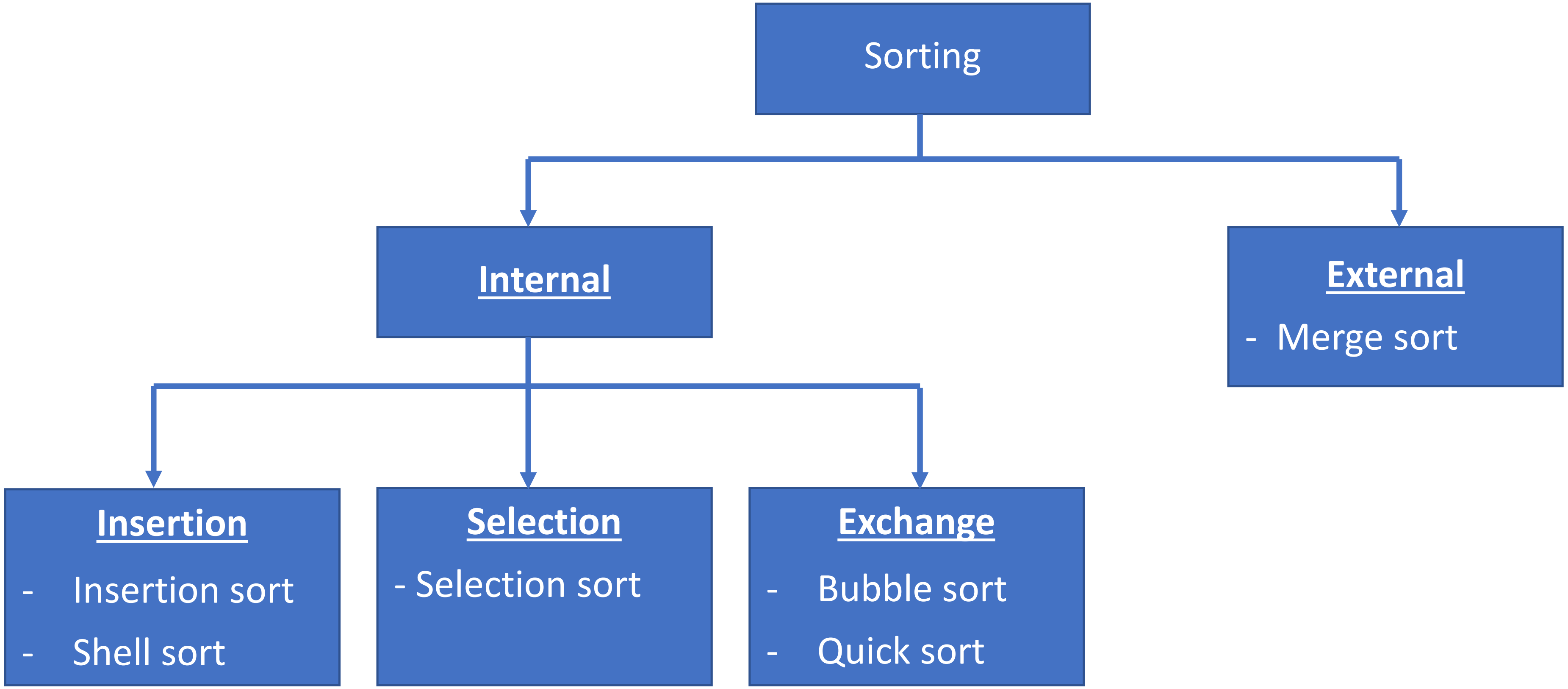
Sorting

- ประเภทของการ Sorting

1. Non-comparison sort
    - Counting Sort
    - Radix Sort
    - Bucket Sort

Sorting

- ประเภทของการ Sorting



Sorting

Internal

External

- Merge sort

Insertion

- Insertion sort
- Shell sort

Selection

- Selection sort

Exchange

- Bubble sort
- Quick sort

Sorting

- Merge sort
  - Merge sort is the sorting method that will divided the unsorted array by half until the array contains only one value in each array. This sorting method is based on recursion programming because of its divided procedure. However, This method has static big-O notation because its steps will the same every time even the array is nearly sorted.

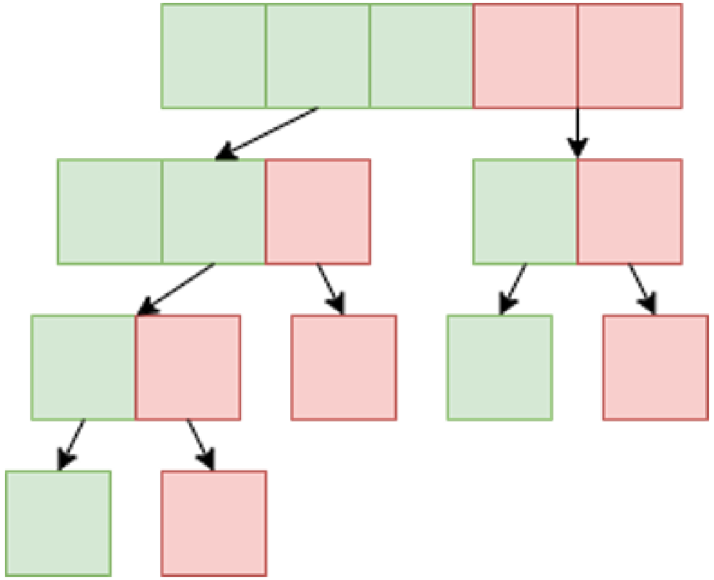
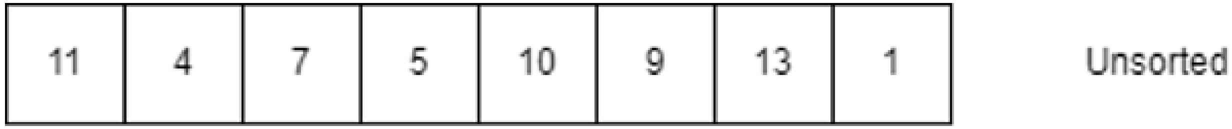


Figure 1: The extraction part of merge sort

Sorting

- Merge sort (continued)
  - If the current extraction part contains odd size of array. The split part will be the **(array length / 2) + 1.**



11	4	7	5	10	9	13	1

Figure 2: Example of unsorted array

Sorting

- Merge sort (continued)
  - If the current extraction part contains odd size of array. The split part will be the **(array length / 2) + 1.**

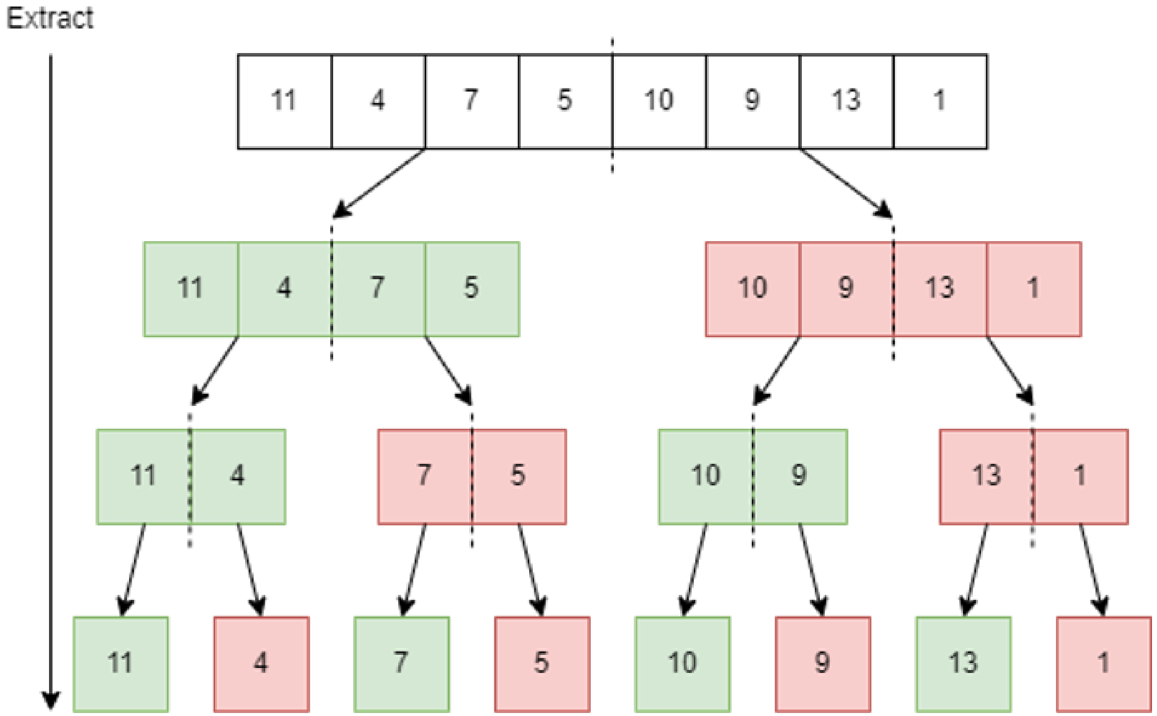


Figure 3: The extraction part of merge sort with the example array

Sorting

- Merge sort (continued)
  - If the current extraction part contains odd size of array. The split part will be the **(array length / 2) + 1.**

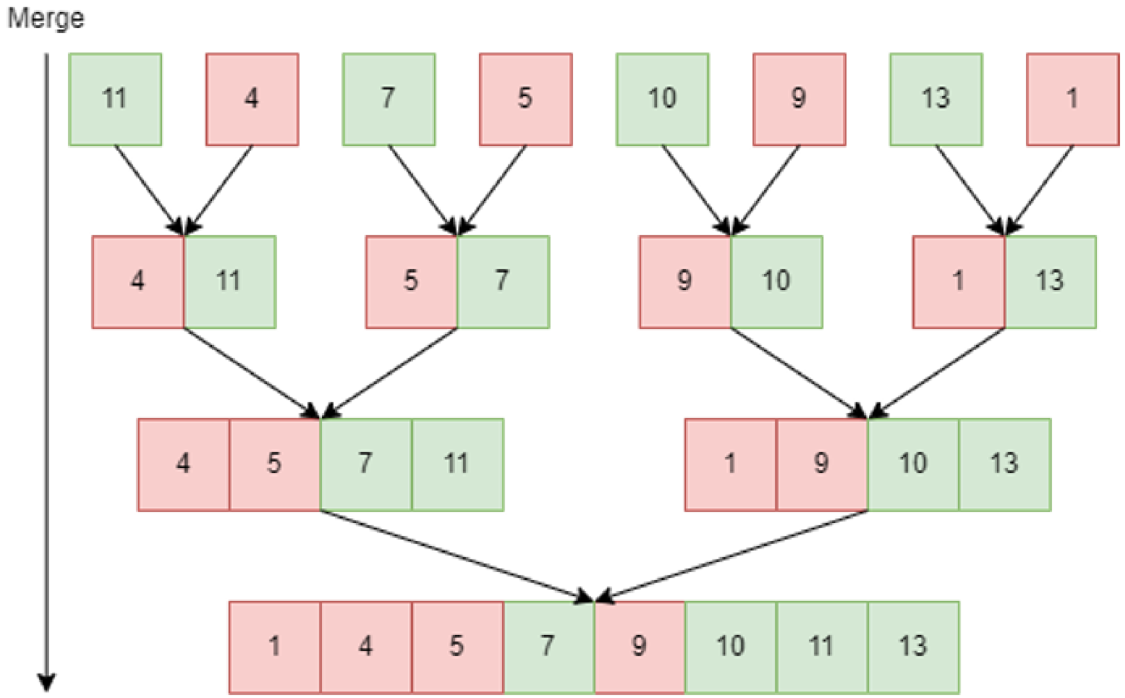


Figure 4: The merge part of merge sort with the example array.

Sorting

- Quick sort
  - The quicksort is the method that is using recursion programming. This sorting method is relying on the pivot to rotate. However this method still rely on luck because of the selecting pivot. If the pivot that selected is better luck, the sorting will be fast unlike unluck selected pivot.

    To selected the started pivot, there are two ways to select.

    1.) **The median or first index.** (Hoare partition scheme)

    2.) **The last index.** (Lomuto partition scheme)

    3.) **Random quicksort** (Random selecting pivot)

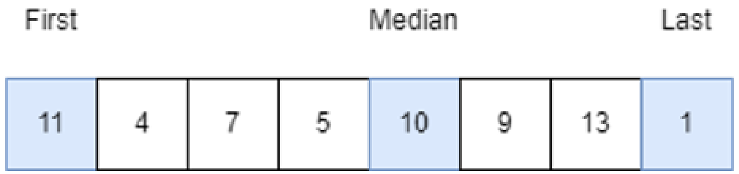
Sorting

Quick sort (continued)

- The quicksort is the method that is using recursion programming. This sorting method is relying on the pivot to rotate. However this method still rely on luck because of the selecting pivot. If the pivot that selected is better luck, the sorting will be fast unlike unluck selected pivot.

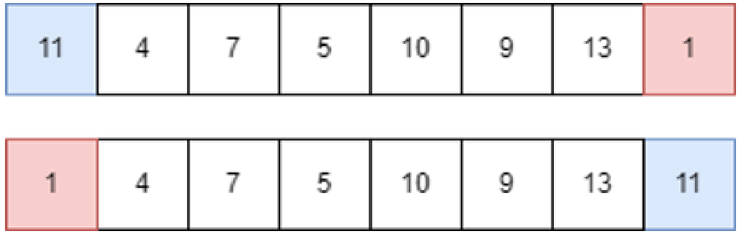
  To selected the started pivot, there are two ways to select.

  1.) **The median or first index.** (Hoare partition scheme)



First				Median			Last
11	4	7	5	10	9	13	1

Figure 5: The pivot selection methods.



11	4	7	5	10	9	13	1
1	4	7	5	10	9	13	11

Figure 6: Selecting the first index (blue) then swap with the last index (red)

Sorting

- ## Quick sort (continued)
  - The quicksort is the method that is using recursion programming. This sorting method is relying on the pivot to rotate. However this method still rely on luck because of the selecting pivot. If the pivot that selected is better luck, the sorting will be fast unlike unluck selected pivot.

    To selected the started pivot, there are two ways to select.

    1.) **The median or first index.** (Hoare partition scheme)

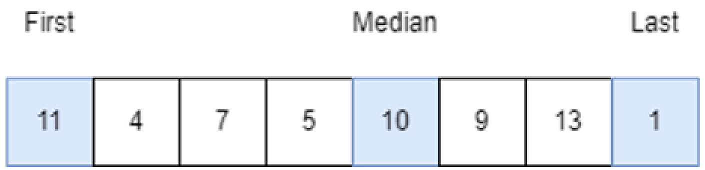


Figure 5: The pivot selection methods.

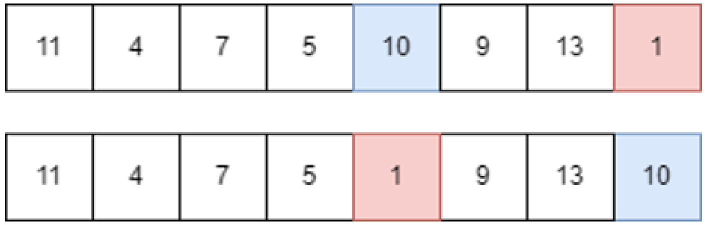


Figure 7: Selecting the median index (blue) then swap with the last index (red)

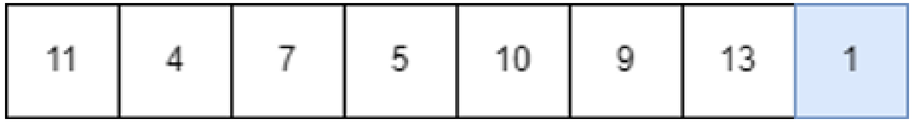
Sorting

- Quick sort (continued)
  - The quicksort is the method that is using recursion programming. This sorting method is relying on the pivot to rotate. However this method still rely on luck because of the selecting pivot. If the pivot that selected is better luck, the sorting will be fast unlike unluck selected pivot.

    To selected the started pivot, there are two ways to select.

    1.) **The median or first index.** (Hoare partition scheme)

    2.) **The last index.** (Lomuto partition scheme)



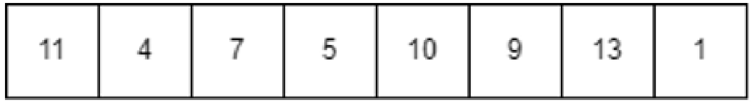
11	4	7	5	10	9	13	1

Figure 8: Selecting the last index (blue)

Sorting

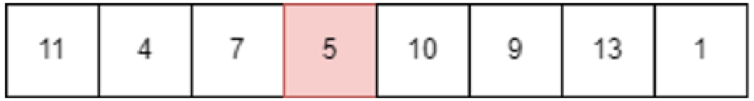
- Quick sort (continued)

  1.) **The median or first index.** (Hoare partition scheme)



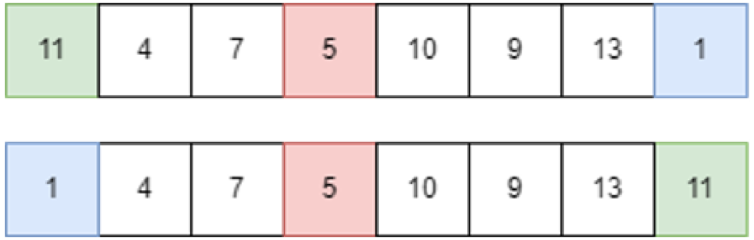
11	4	7	5	10	9	13	1

Example of unsorted array



11	4	7	5	10	9	13	1

Figure 23: Select the middle index as pivot



11	4	7	5	10	9	13	1
1	4	7	5	10	9	13	11

Comparison index green as approaching from left and blue as approaching from right. If left value is more than pivot and right value, right value is less than pivot and less than left value. Swap left and right value.

Sorting

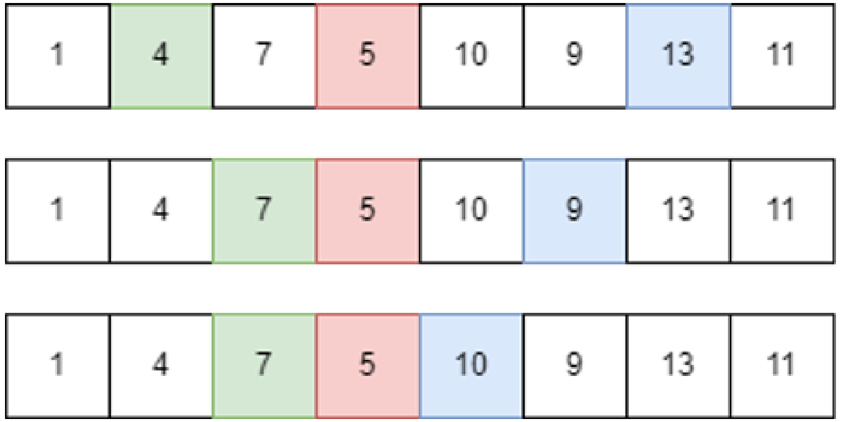
- Quick sort (continued)

  1.) **The median or first index.** (Hoare partition scheme)

The Hoare quicksort swap conditions

Left > Pivot > Right

Left > Right



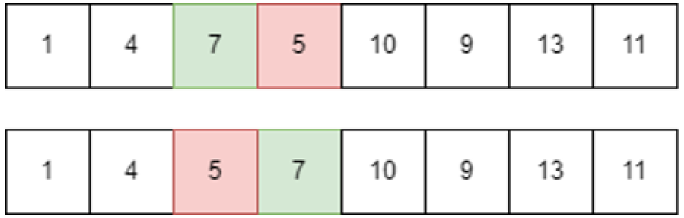
1	4	7	5	10	9	13	11
1	4	7	5	10	9	13	11
1	4	7	5	10	9	13	11

Which value is not entering condition, the pointer will moved

Sorting

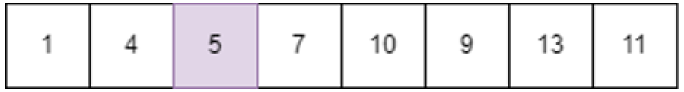
- Quick sort (continued)

  1.) **The median or first index.** (Hoare partition scheme)



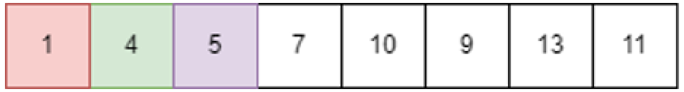
1	4	7	5	10	9	13	11
1	4	5	7	10	9	13	11

Right side is now correct but the left side is not 7 still more than 5 and need to swap



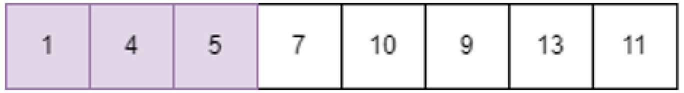
1	4	5	7	10	9	13	11

Figure 27: 5 is now in correct position



1	4	5	7	10	9	13	11

Sorting the right side but they all sorted



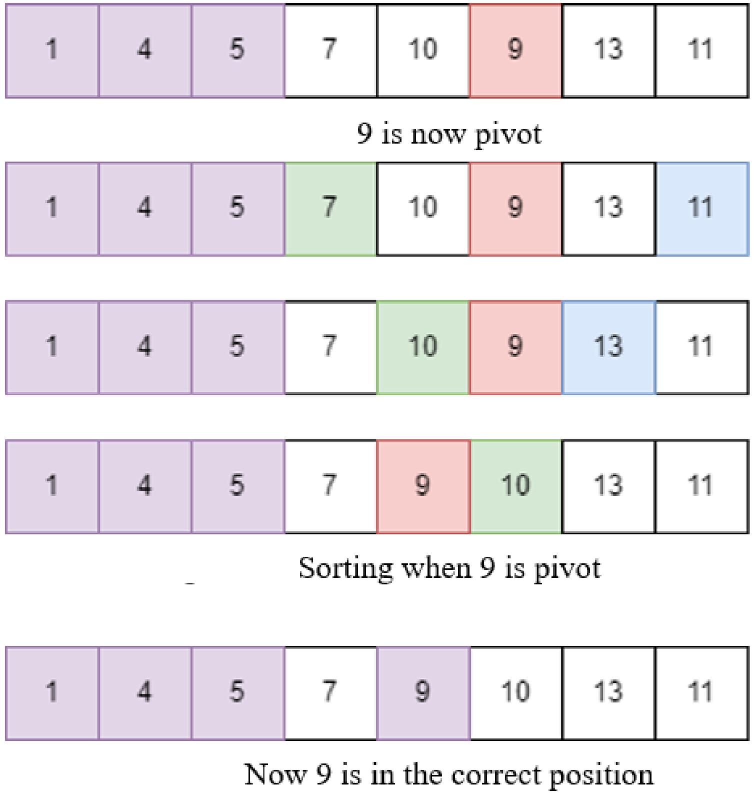
1	4	5	7	10	9	13	11

Half sorted

Sorting

- Quick sort (continued)

  1.) **The median or first index.** (Hoare partition scheme)



1	4	5	7	10	9	13	11

9 is now pivot

1	4	5	7	10	9	13	11

1	4	5	7	10	9	13	11

1	4	5	7	9	10	13	11

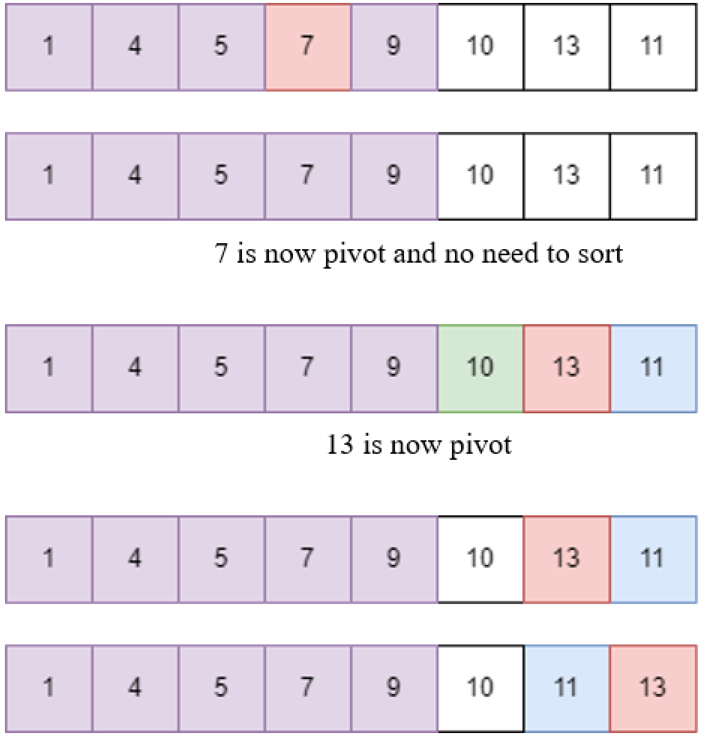
Sorting when 9 is pivot

1	4	5	7	9	10	13	11

Now 9 is in the correct position

- Quick sort (continued)

1.) **The median or first index.** (Hoare partition scheme)



1	4	5	7	9	10	13	11

1	4	5	7	9	10	13	11

7 is now pivot and no need to sort

1	4	5	7	9	10	13	11

13 is now pivot

1	4	5	7	9	10	13	11

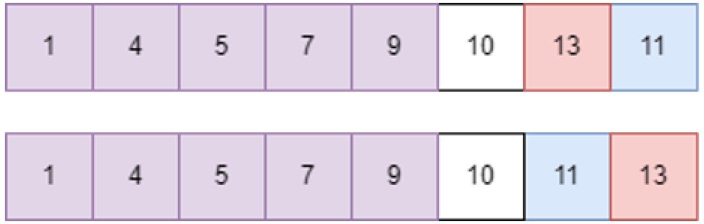
1	4	5	7	9	10	11	13

No remaining value from left side but 13 (pivot) is more than 11 (right)

Sorting

- Quick sort (continued)

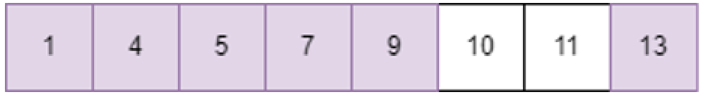
1.) **The median or first index.** (Hoare partition scheme)



1	4	5	7	9	10	13	11

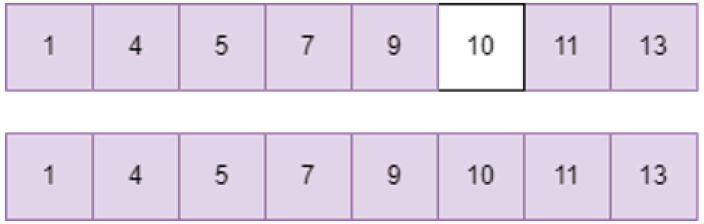
1	4	5	7	9	10	11	13

No remaining value from left side but 13 (pivot) is more than 11 (right)



1	4	5	7	9	10	11	13

Sorted value



1	4	5	7	9	10	11	13

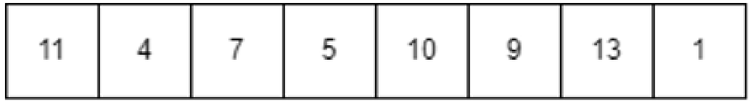
1	4	5	7	9	10	11	13

Sorted value

Sorting

- Quick sort (continued)

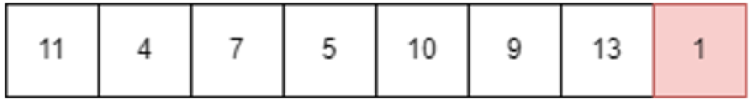
  2.) **The last index.** (Lomuto partition scheme)



11	4	7	5	10	9	13	1

Unsorted

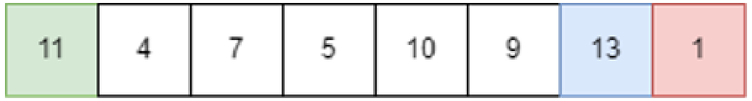
Example of unsorted array



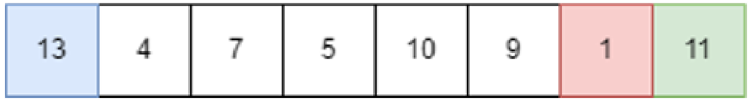
11	4	7	5	10	9	13	🟥 1

Select a pivot

Select the last index as pivot



🟩 11	4	7	5	10	9	🟦 13	🟥 1



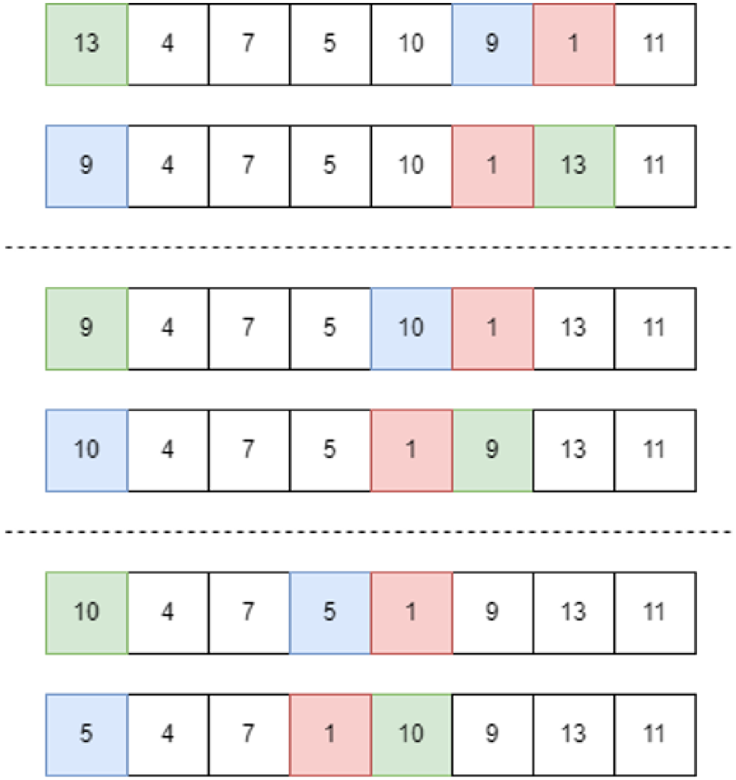
🟦 13	4	7	5	10	9	🟥 1	🟩 11

Compare between pivot (red) and the comparison index (green) then decrease the pivot index by 1, move the comparison index (green) to the former pivot index and move the previous index of pivot (blue) to the first index

Sorting

- Quick sort (continued)

2.) **The last index.** (Lomuto partition scheme)



13	4	7	5	10	9	1	11
9	4	7	5	10	1	13	11

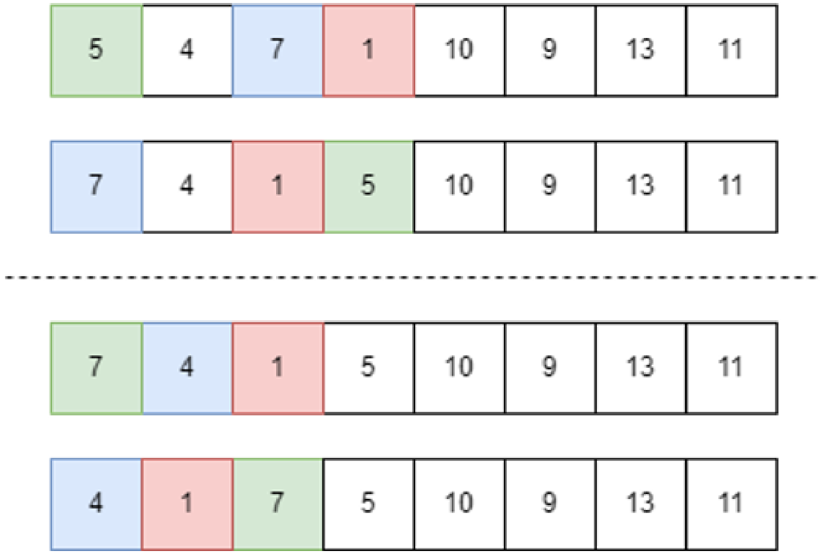
9	4	7	5	10	1	13	11
10	4	7	5	1	9	13	11

10	4	7	5	1	9	13	11
5	4	7	1	10	9	13	11

Sorting

- Quick sort (continued)

2.) **The last index.** (Lomuto partition scheme)



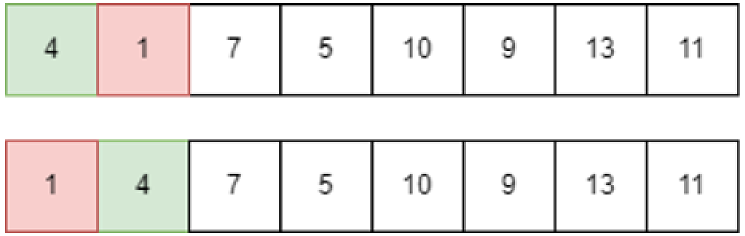
5	4	7	1	10	9	13	11

7	4	1	5	10	9	13	11

7	4	1	5	10	9	13	11

4	1	7	5	10	9	13	11

Sorting when 1 is a pivot



4	1	7	5	10	9	13	11

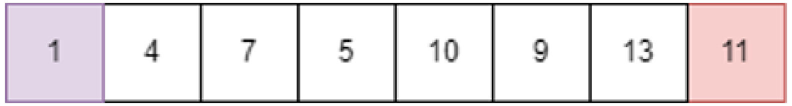
1	4	7	5	10	9	13	11

The finished part of 1 is pivot

Sorting

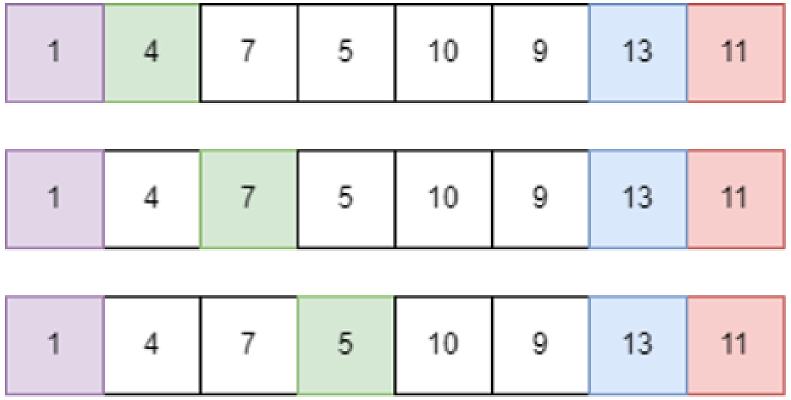
- Quick sort (continued)

2.) **The last index.** (Lomuto partition scheme)



1	4	7	5	10	9	13	11

Now 1 is the correct positon and 11 is now pivot

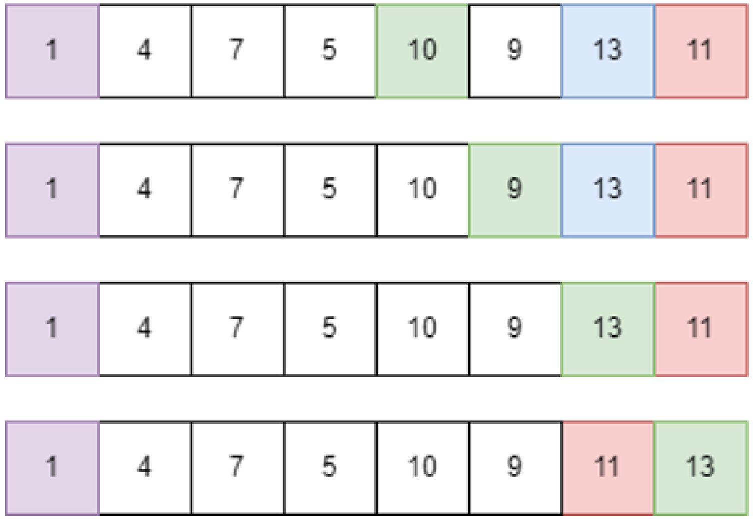


1	4	7	5	10	9	13	11
1	4	7	5	10	9	13	11
1	4	7	5	10	9	13	11

Sorting

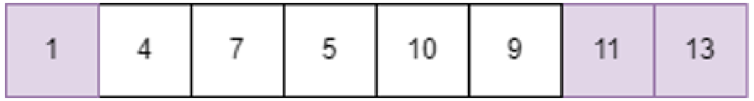
- Quick sort (continued)

2.) **The last index.** (Lomuto partition scheme)



1	4	7	5	10	9	13	11
1	4	7	5	10	9	13	11
1	4	7	5	10	9	13	11
1	4	7	5	10	9	11	13

The 11 is pivot and comparison from the front of the array. If pivot value is more than the comparison value, the comparison value index will increase by 1 until pivot index.



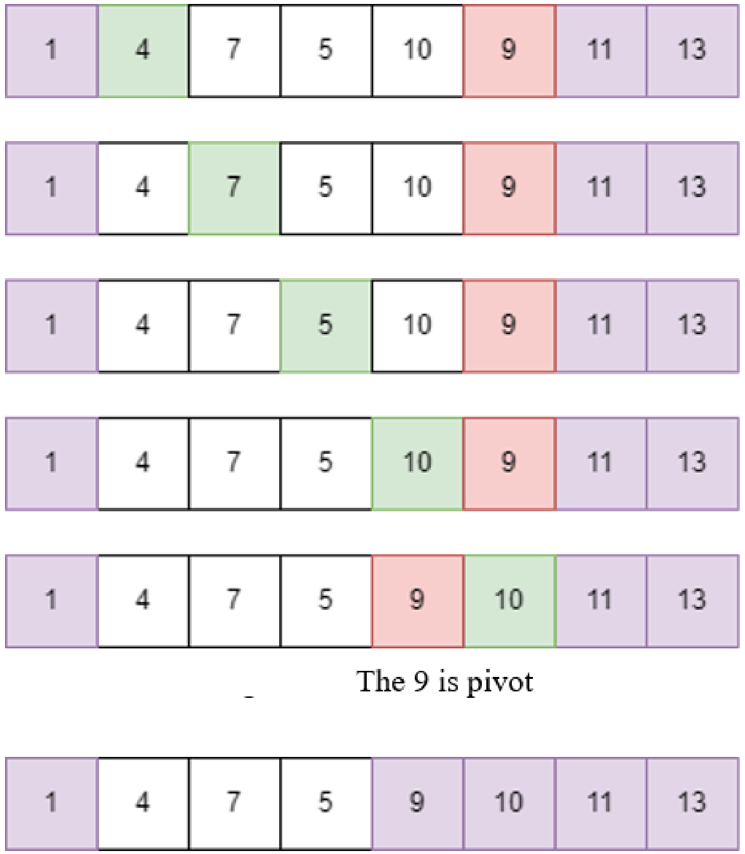
1	4	7	5	10	9	11	13

The correctly index

Sorting

- Quick sort (continued)

2.) **The last index.** (Lomuto partition scheme)



1	4	7	5	10	9	11	13
1	4	7	5	10	9	11	13
1	4	7	5	10	9	11	13
1	4	7	5	10	9	11	13
1	4	7	5	9	10	11	13

The 9 is pivot

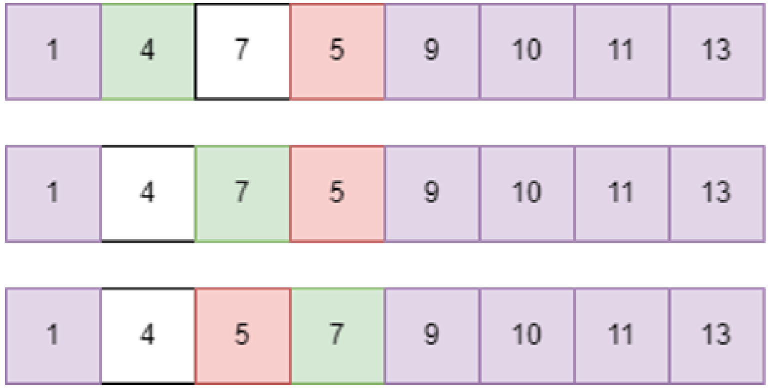
1	4	7	5	9	10	11	13

The correctly index

Sorting

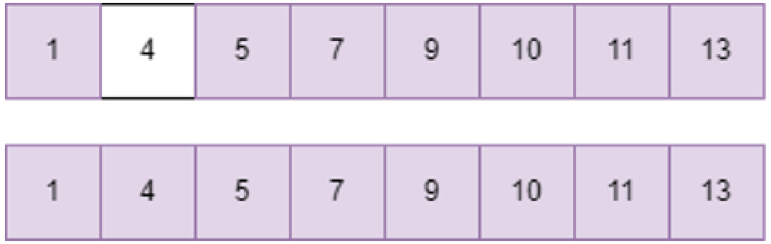
- Quick sort (continued)

2.) **The last index.** (Lomuto partition scheme)



1	4	7	5	9	10	11	13
1	4	7	5	9	10	11	13
1	4	5	7	9	10	11	13

The 5 is pivot



1	4	5	7	9	10	11	13
1	4	5	7	9	10	11	13

5 and 7 are in the correct place and 4 has no sort left