SCHOOL OF
ENGINEERING
BANGKOK UNIVERSITY

Sorting

AIE 311 : Data structure and Algorithm

Sorting

- **Sorting** => การเรียงลำดับข้อมูล

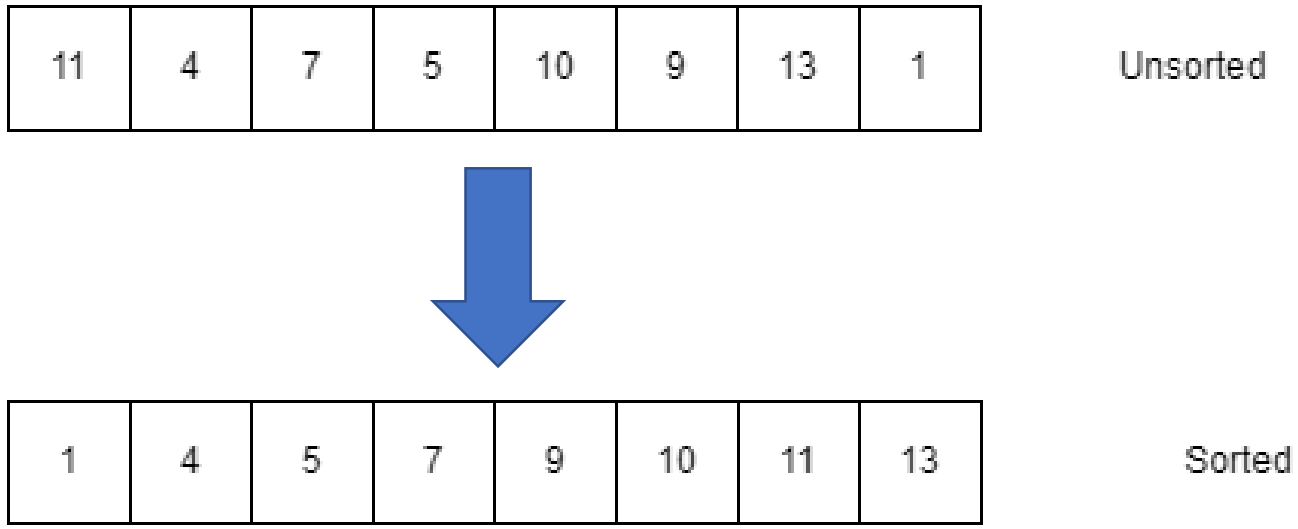


Diagram of a Tree Data Structure

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

- **Bubble Sort**
  - Bubble sort is the sorting method that the selected value will compare and swap location in every pair until reach the last value of array. This meant this method will slow at start pace but faster after sorting.

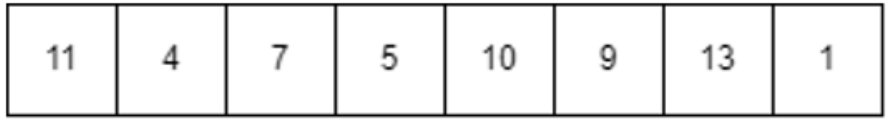


Figure 1: Example array

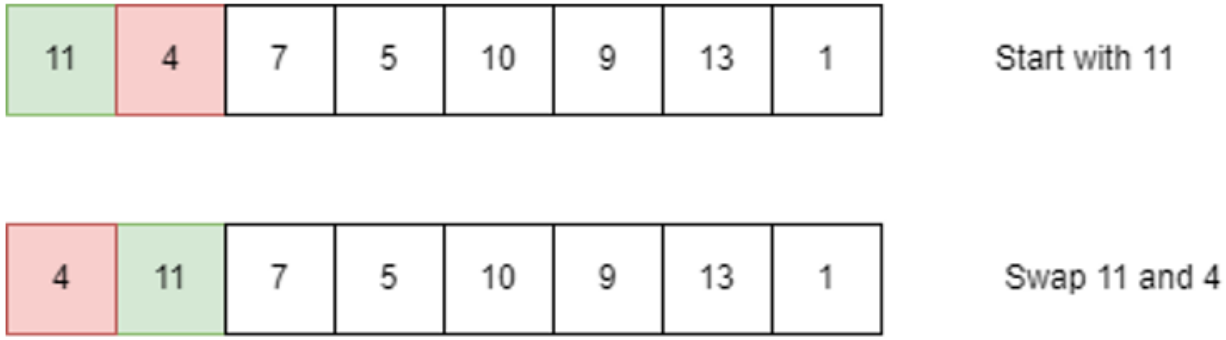


Figure 2: Paring with 1st index and 2nd index then swap

- Bubble Sort (continued)

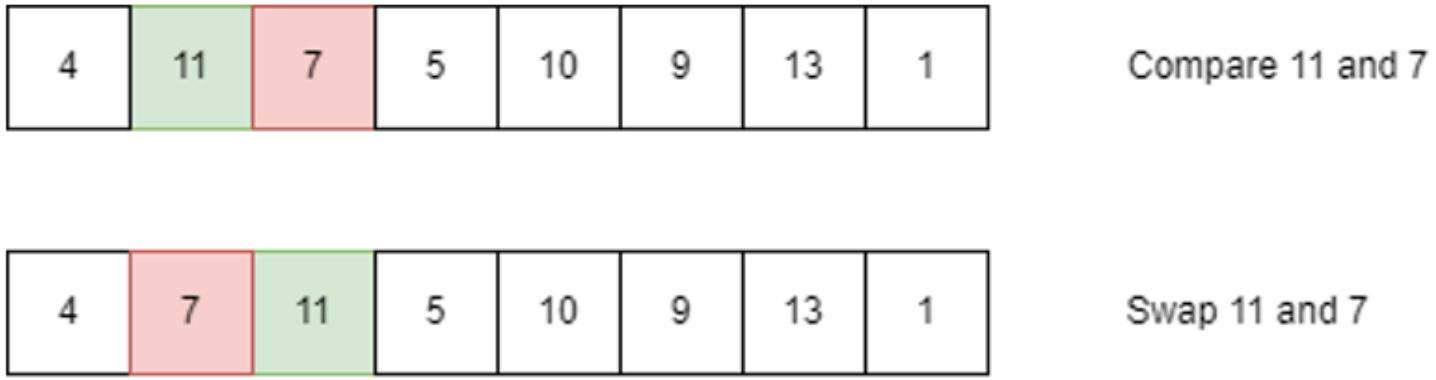


Figure 3: Paring with 2nd index and 3rd index then swap

- Bubble Sort (continued)

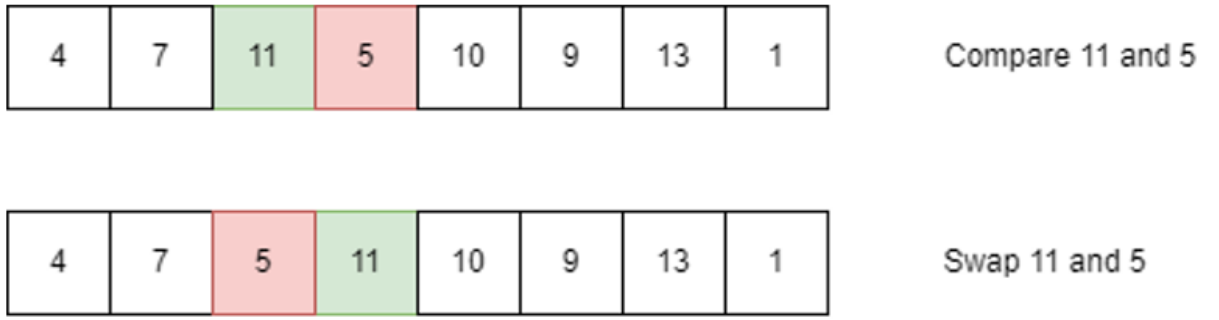
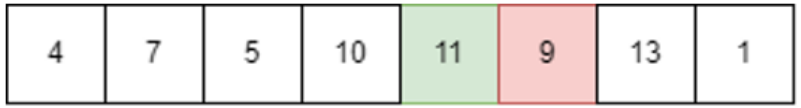


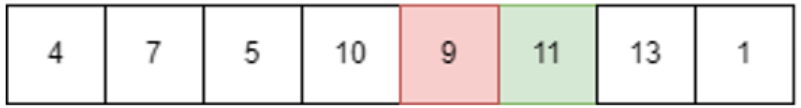
Figure 4: Paring with 3rd index and 4th index then swap

Figure 5: Paring with 4th index and 5th index then swap

- Bubble Sort (continued)

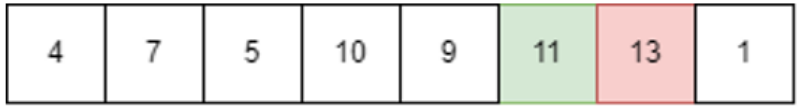


Compare 11 and 9

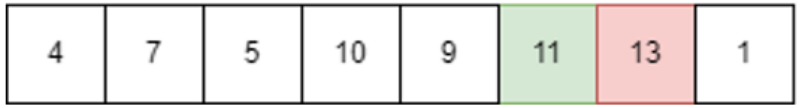


Swap 11 and 9

Figure 6: Paring with 5th index and 6th index then swap



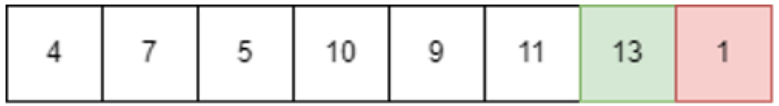
Compare 11 and 13



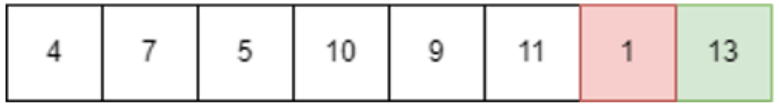
11 is less than 13

Figure 7: Paring with 6th index and 7th index but 6th index value is less than 7th index value

- Bubble Sort (continued)

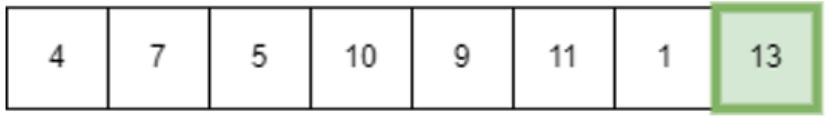


Current value is now 13 and compare to 1



Swap 13 and 1

Figure 8: Current value is now index 7th because 7th index value is more than 6th index value then compare and swap value between 7th index and 8th index



The last index is now locked

Figure 9: The last index is now locked.

This mean in every run, the locked index will greater and the sorting will be faster.

- Bubble Sort (continued)

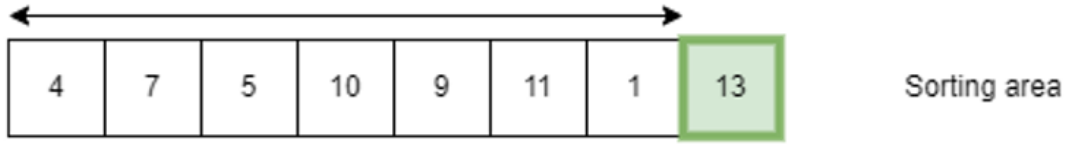


Figure 10: 2nd run with remaining remaing area/index

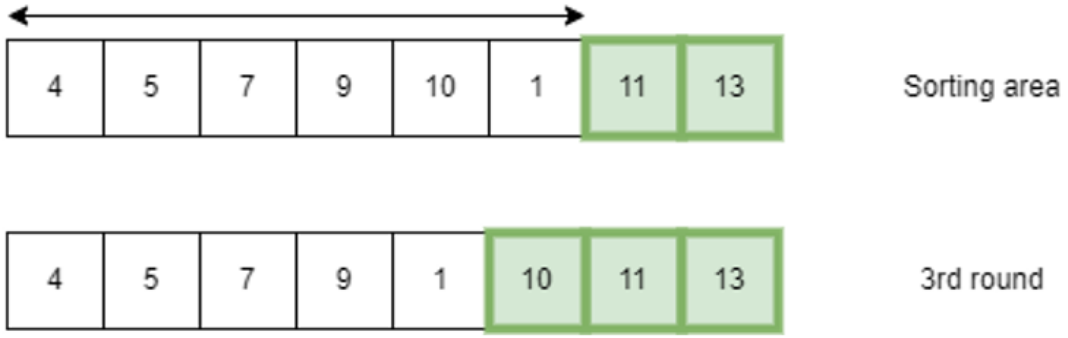


Figure 11: 3rd run with remaining remaing area/index

- Bubble Sort (continued)

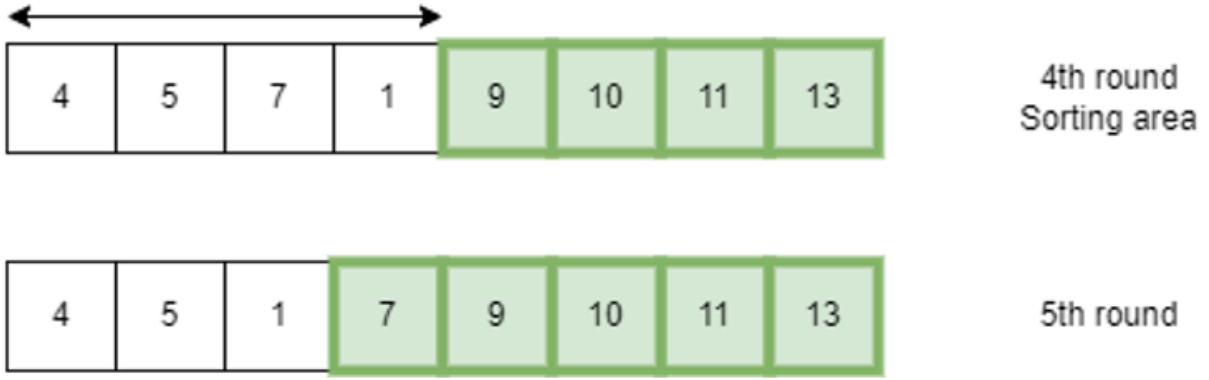


Figure 12: 5th run with remaining remaing area/index

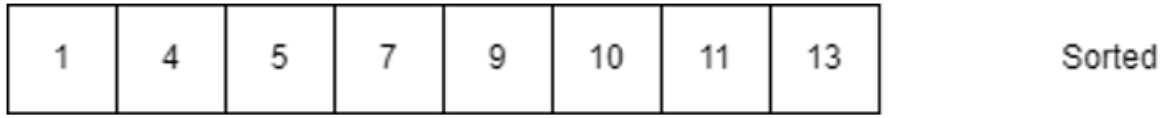


Figure 13: Sorted array

- **Selection Sort**
  - Selection sort is the method to find the lowest value from array then swap to the index of current run. The first run index will equal to 0 and the third run index will equal to 2 which means index will equal running round. - 1.

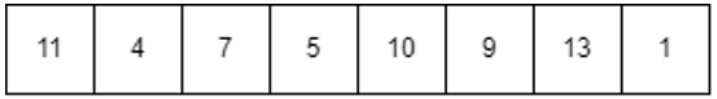


Figure 14: Example array

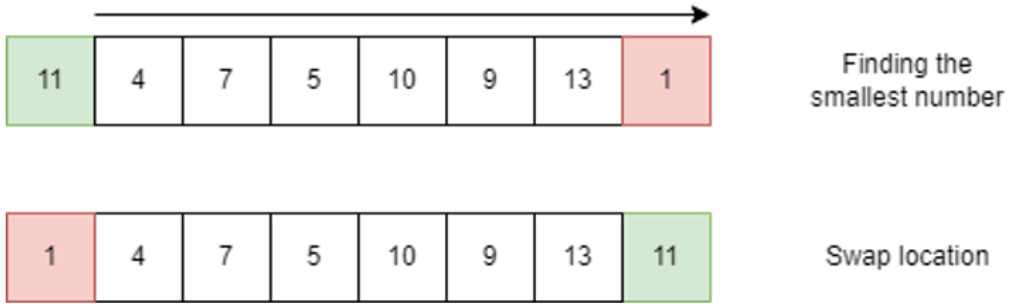


Figure 14: Find the minimum value and swap to the 1st index as the first run.

Figure 15: Lock 1st index after swapped.

- Selection Sort (Continued)

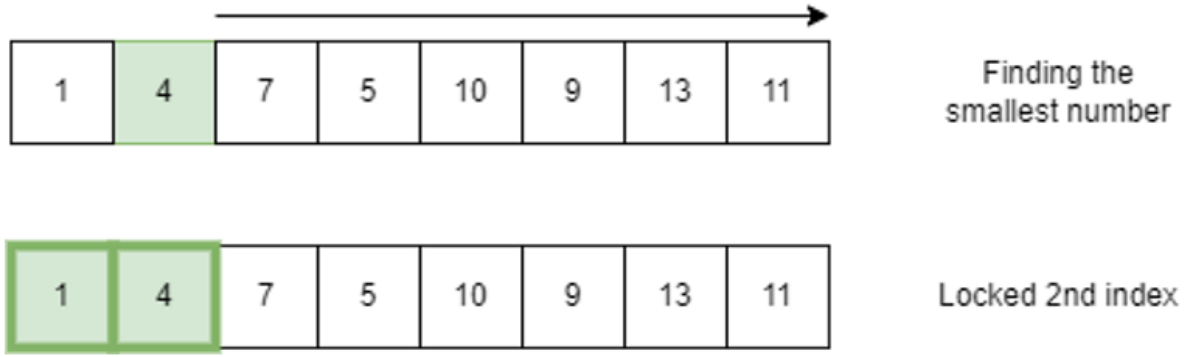


Figure 16: Find the minimum value after 2nd index but its value is the smallest

- Selection Sort (Continued)

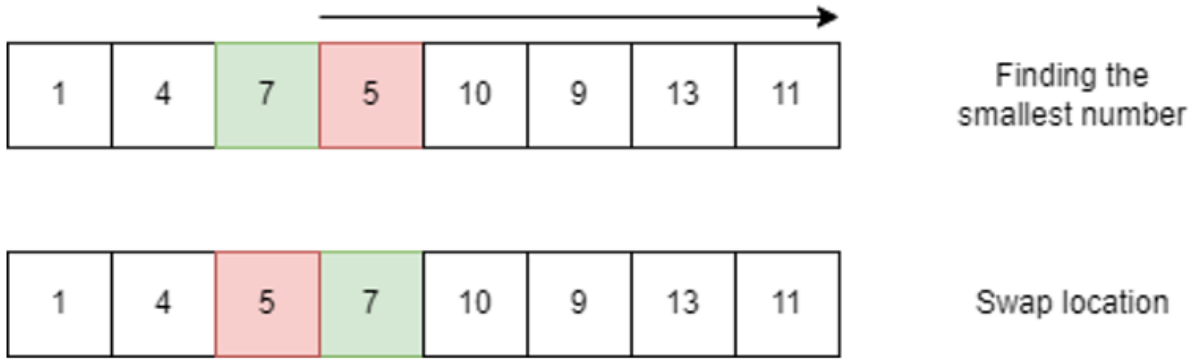


Figure 17: Find the minimum value and swap to the 3rd index as the third run.

Figure 18: Lock 3rd index after swapped.

- Insertion Sort
  - Insertion sort is similar to bubble sort but only compare and will stop when current value is less than the previous value. After insert sorted value the new run will started.

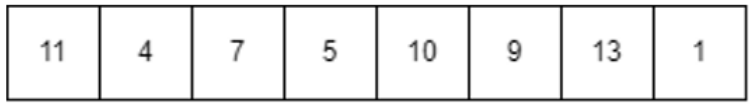


Figure 19: Example array

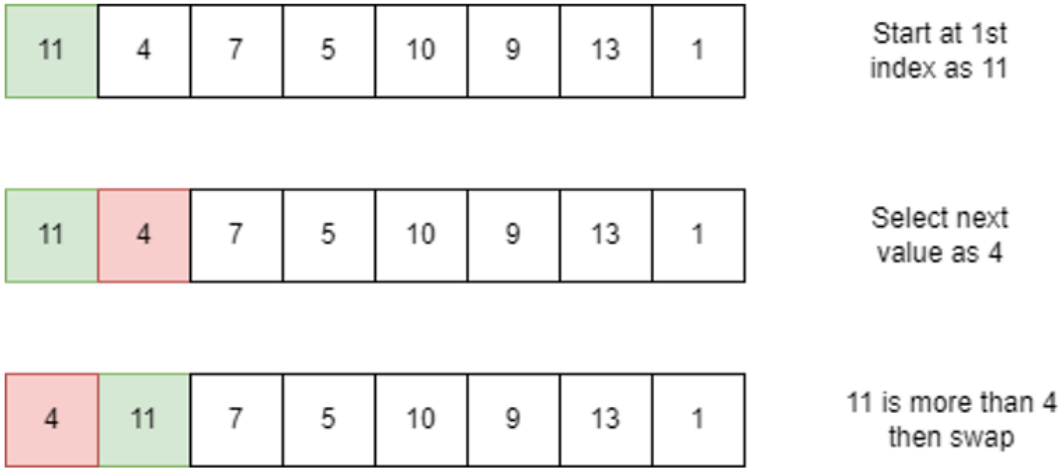


Figure 20: Insertion sort 1st run

- Insertion Sort (Continued)

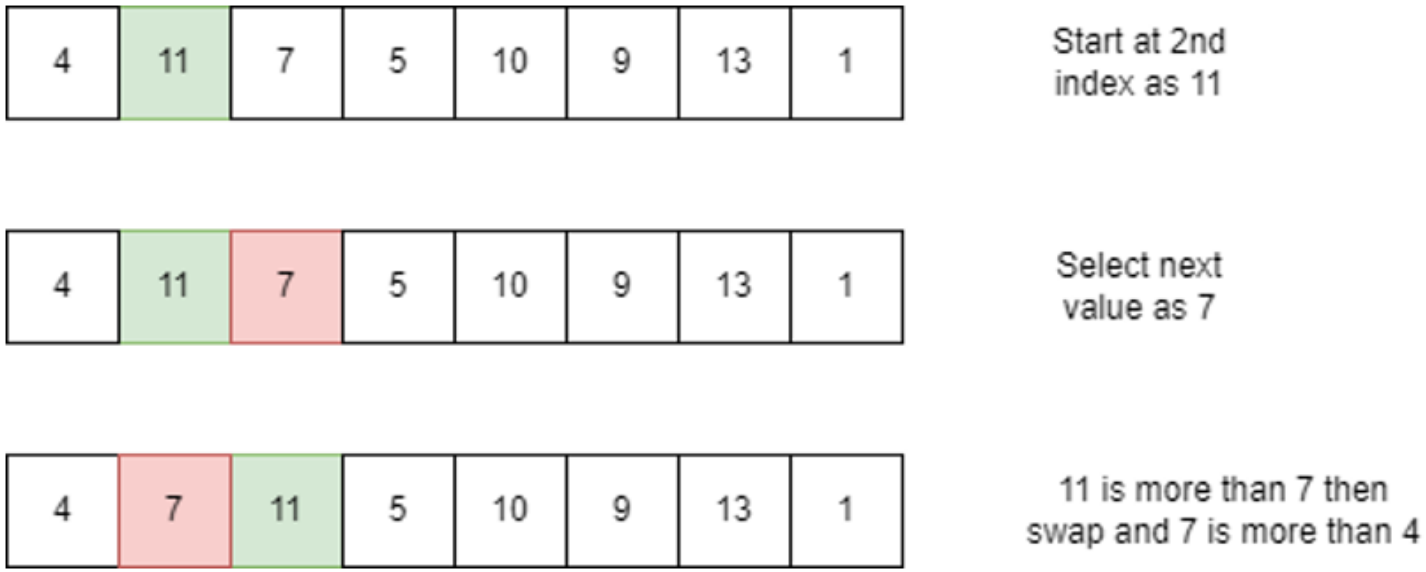


Figure 21: Insertion sort 2nd run

- Insertion Sort (Continued)

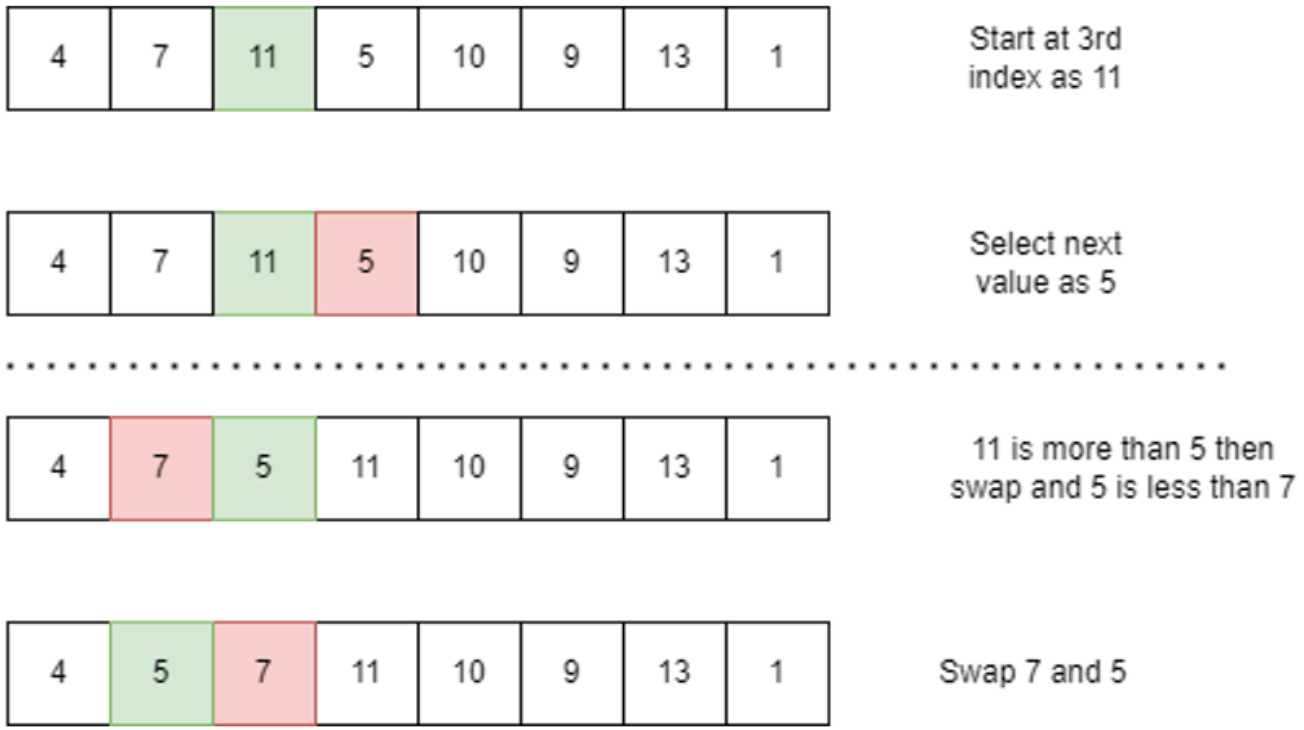


Figure 22: Insertion sort 3rd run

- Insertion Sort (Continued)

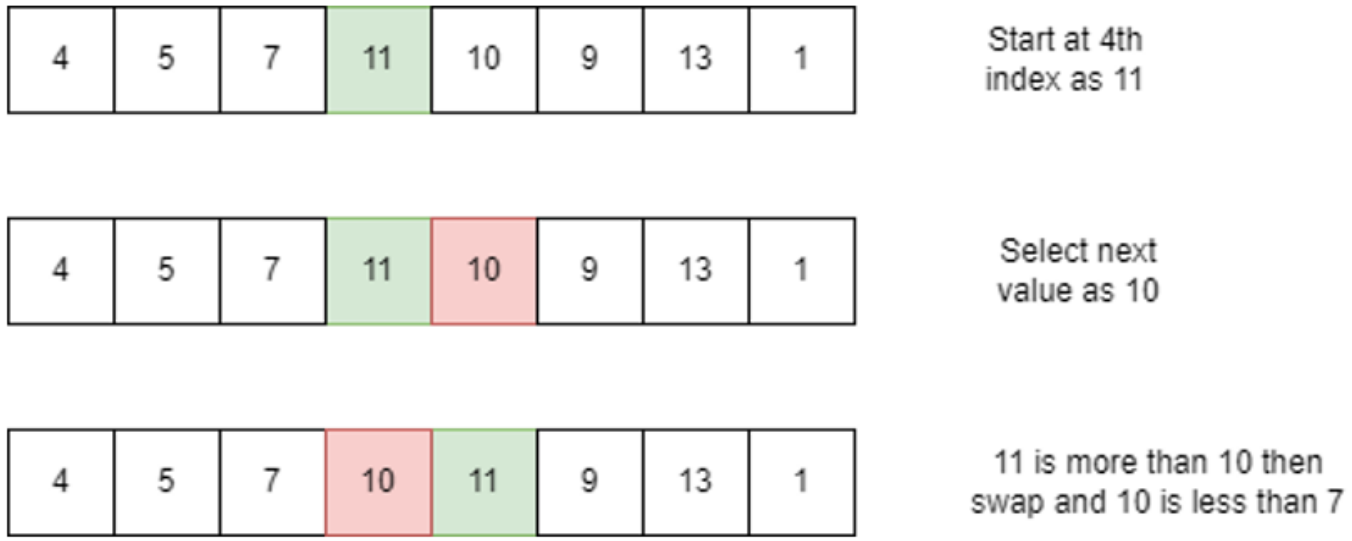


Figure 23: Insertion sort 4th run

- Insertion Sort (Continued)

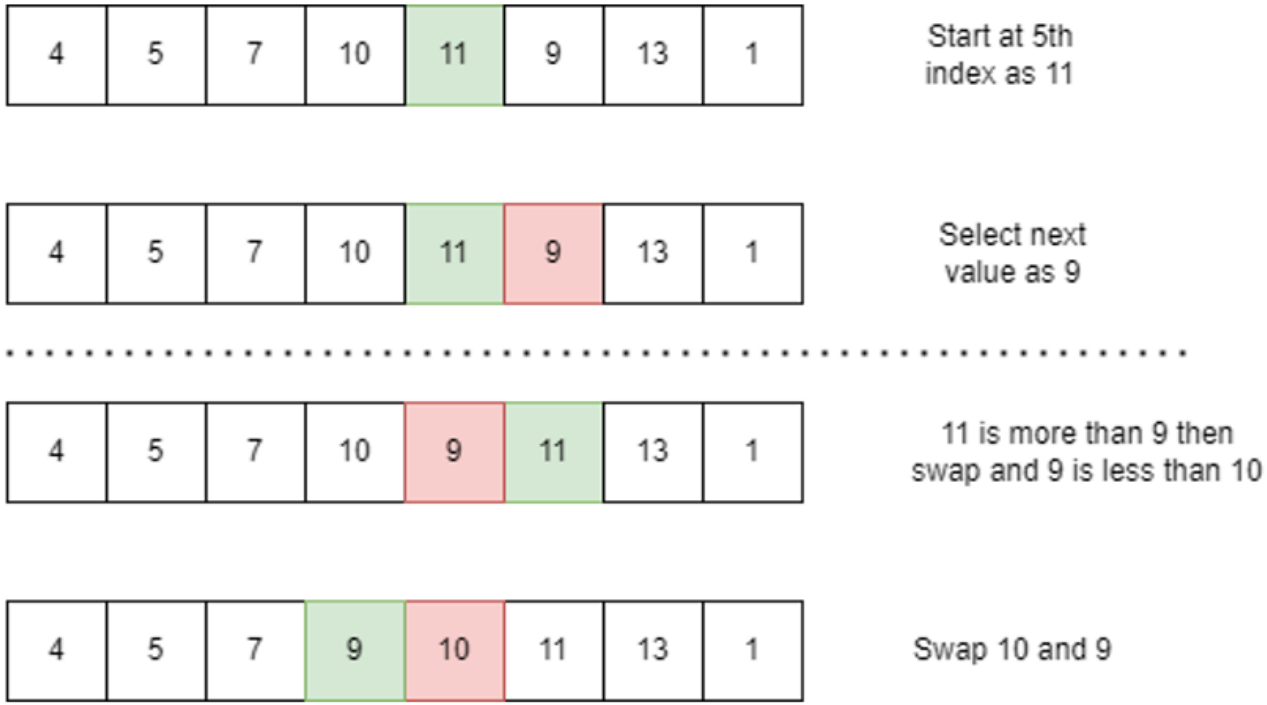


Figure 24: Insertion sort 5th run which similar to the 3rd run

- Insertion Sort (Continued)

Figure 25: Insertion sort 6th run and no need to swap in this run

- Insertion Sort (Continued)

Figure 26: Insertion sort 7th run which similar to the 3rd run (part 1)

- Insertion Sort (Continued)

Figure 27: Insertion sort 7th run which similar to the 3rd run (part 2)

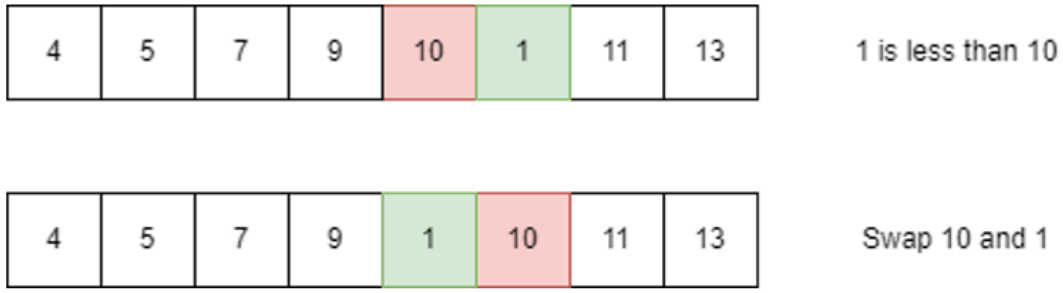


Figure 28: Insertion sort 7th run which similar to the 3rd run (part 3)

- Insertion Sort (Continued)

Figure 29: Insertion sort 7th run which similar to the 3rd run (part 4)

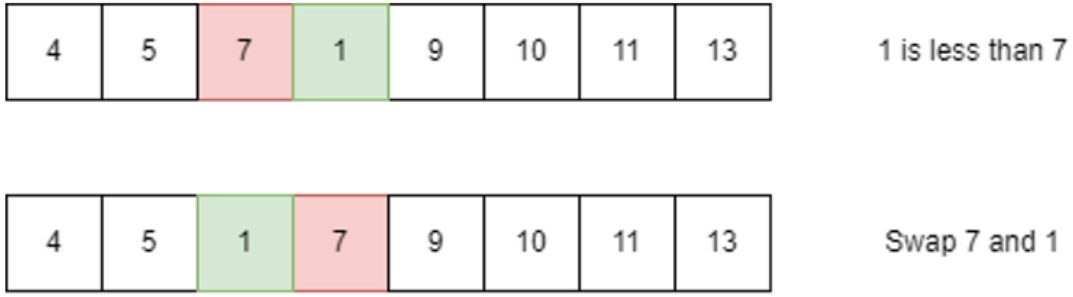


Figure 30: Insertion sort 7th run which similar to the 3rd run (part 5)

- Insertion Sort (Continued)

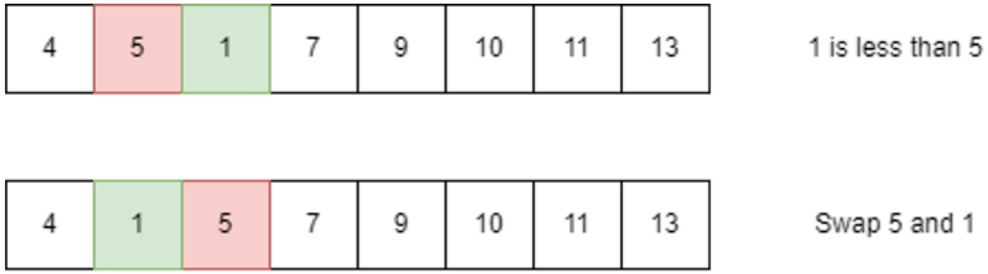


Figure 31: Insertion sort 7th run which similar to the 3rd run (part 6)

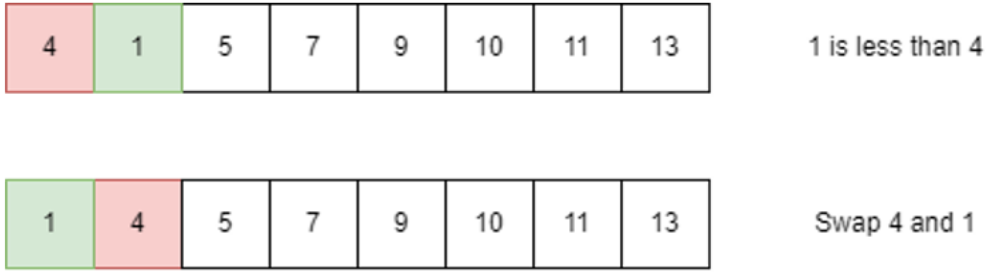


Figure 32: Insertion sort 7th run which similar to the 3rd run (part 7)

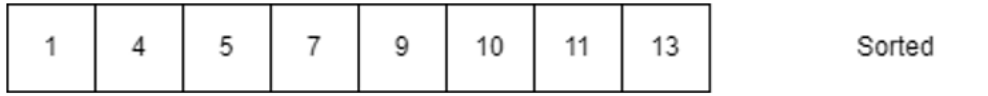


Figure 33: Sorted array