Data Structures Lab 3

Course: Data Structures (CL2001) **Semester:** Spring 2024
Instructor: Bushra Sattar **T.A:**

Note:
Lab manual cover following below elementary sorting algorithms
{Bubble, insertion, selection}

- Maintain discipline during the lab.
- Just raise hand if you have any problem.
- Completing all tasks of each lab is compulsory.
- Get your lab checked at the end of the session.

Bubble Sort
Bubble Sort is a simple algorithm which is used to sort a given set of n elements provided in form of an array with n number of elements. Bubble Sort compares all the element one by one and sort them based on their values.

If the given array has to be sorted in ascending order, then bubble sort will start by comparing the first element of the array with the second element, if the first element is greater than the second element, it will swap both the elements, and then move on to compare the second and the third element, and so on.

Task-1:
Given an array of int arr[]. Sort given integers using Bubble Sort and display the sorted array.
Key Points:

1. Bubble Sort, the two successive strings arr[i] and arr[i+1] are exchanged whenever arr[i]> arr[i+1]. The larger values sink to the bottom and hence called sinking sort. At the end of each pass, smaller values gradually "bubble" their way upward to the top and hence called bubble sort.

Implementing Bubble Sort Algorithm

Following are the steps involved in bubble sort (for sorting a given array in ascending order):

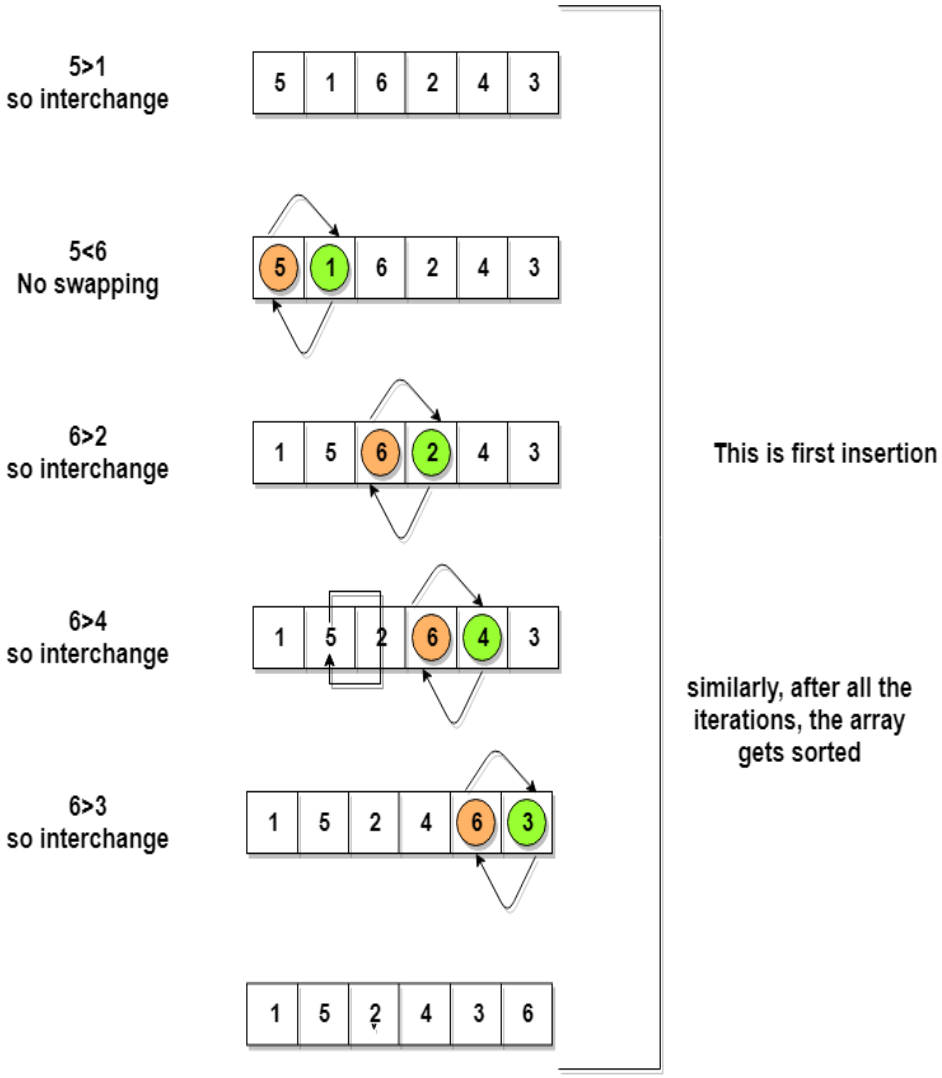
1. Starting with the first element (index = 0), compare the current element with the next element of the array.

2. If the current element is greater than the next element of the array, swap them.

3. If the current element is less than the next element, move to the next element. **Repeat Step 1**.

Let's consider an array with values {5, 1, 6, 2, 4, 3}

Below, we have a pictorial representation of how bubble sort will sort the given array.



So as we can see in the representation above, after the first iteration, 6 is placed at the last index, which is the correct position for it.

Similarly, after the second iteration, 5 will be at the second last index, and so on.

Selection sort

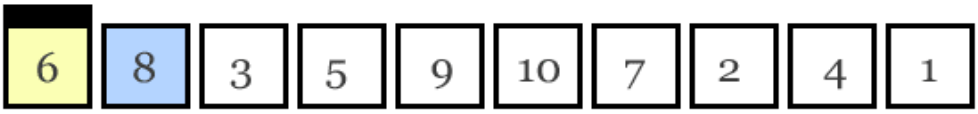
Selection sort is a simple and efficient sorting algorithm that works by repeatedly selecting the smallest (or largest) element from the unsorted portion of the list and moving it to the sorted portion of the list.

The algorithm repeatedly selects the smallest (or largest) element from the unsorted portion of the list and swaps it with the first element of the unsorted part. This process is repeated for the remaining unsorted portion until the entire list is sorted.

Task-2:
Develop C++ solution that takes Marks for ten students in science subjects as user input and perform Sorting using Selection Sort.

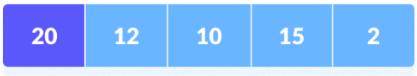
Key Points:

void selectionSort(int [] array, int size) {
Find the smallest element in the array and exchange it with the element in the first position.
Find the second smallest element in the array and exchange it with the element in the second position.
Continue this process until done.
}



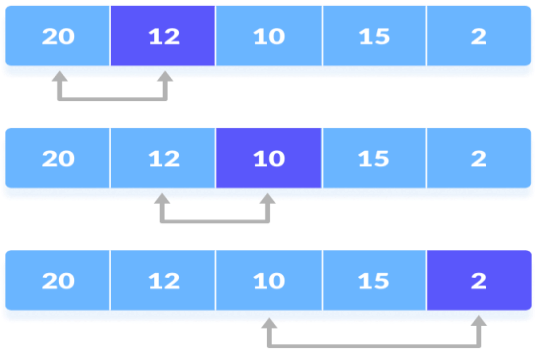
Working of Selection Sort

1. Set the first element as minimum.



2. Compare minimum with the second element. If the second element is smaller than minimum, assign the second element as minimum.

Compare minimum with the third element. Again, if the third element is smaller, then assign minimum to the third element otherwise do nothing. The process goes on until the last element.

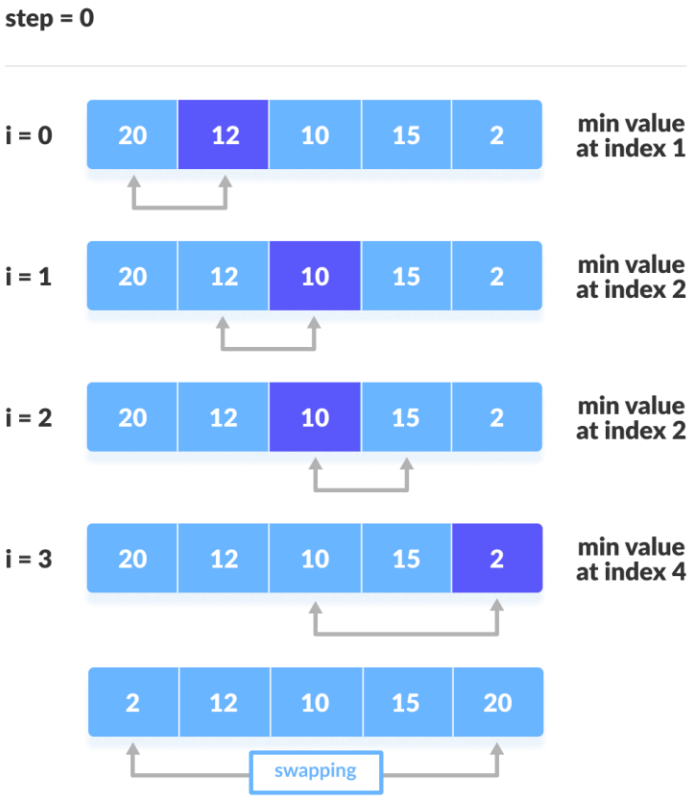


Compare minimum with the remaining elements

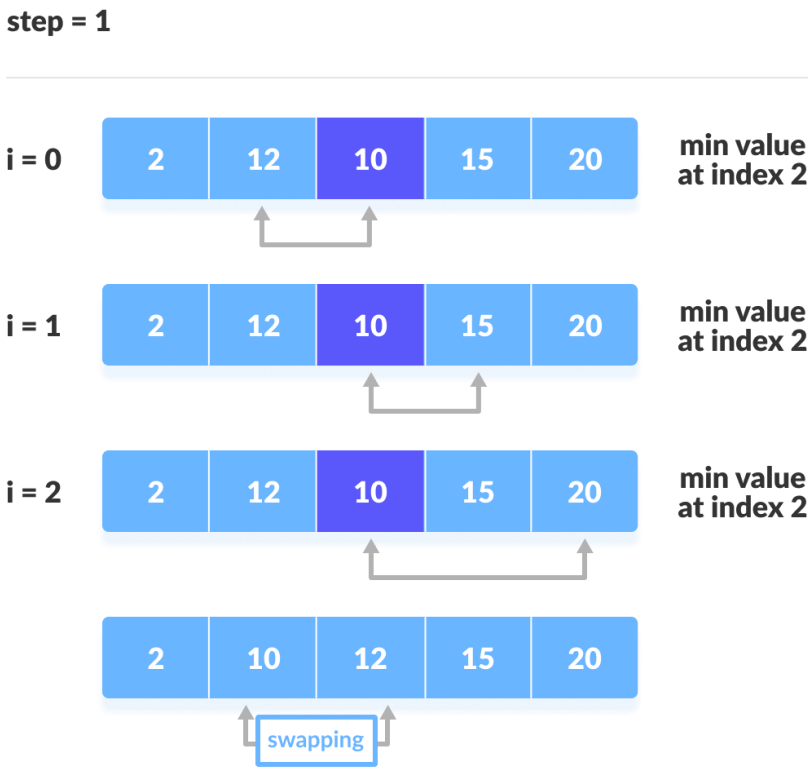
3. After each iteration, minimum is placed in the front of the unsorted list.

Swap the first with minimum

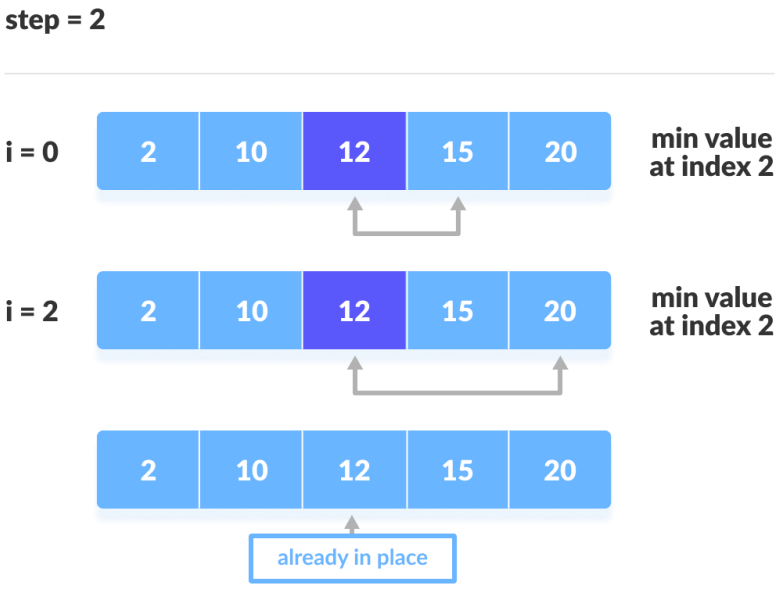
For each iteration, indexing starts from the first unsorted element. Step 1 to 3 are repeated until all the elements are placed at their correct positions.



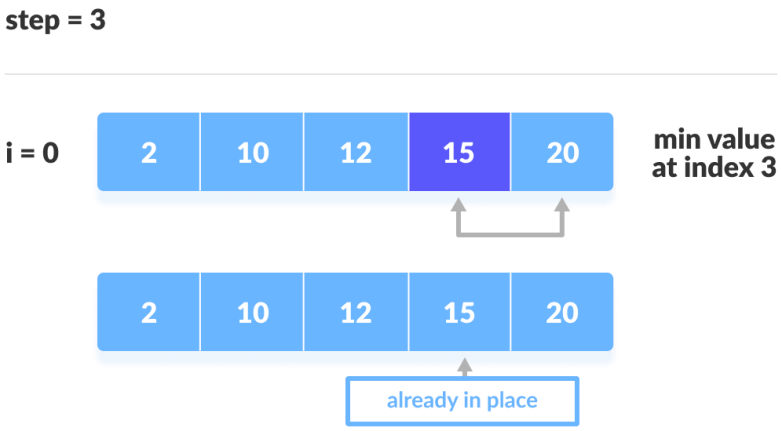
The first iteration



The second iteration



The third iteration



The fourth iteration

Insertion sort

Insertion sort is a simple sorting algorithm that works similar to the way you sort playing cards in your hands. The array is virtually split into a sorted and an unsorted part. Values from the unsorted part are picked and placed at the correct position in the sorted part.

Insertion Sort Algorithm

To sort an array of size N in ascending order iterate over the array and compare the current element (key) to its predecessor, if the key element is smaller than its predecessor, compare it to the elements before. Move the greater elements one position up to make space for the swapped element.

Task-3:
Develop an implementation of insertion sort that moves larger items to the right one position rather than doing full exchanges.

Key Points:

void insertionSort (int [] array, int size) {
Choose the second element in the array and place it in order with respect to the first element.
Choose the third element in the array and place it in order with respect to the first two elements.
Continue this process until done.
Insertion of an element among those previously considered consists of moving larger elements one position to the right and then inserting the element into the vacated position
}

Working of Insertion Sort algorithm

Consider an example: arr[]: **{12, 11, 13, 5, 6}**

12	**11**	**13**	**5**	**6**

First Pass:

Initially, the first two elements of the array are compared in insertion sort.

12	**11**	13	5	6

Here, 12 is greater than 11 hence they are not in the ascending order and 12 is not at its correct position. Thus, swap 11 and 12.

So, for now 11 is stored in a sorted sub-array.

11	**12**	13	5	6

Second Pass:

Now, move to the next two elements and compare them

11	**12**	**13**	5	6

Here, 13 is greater than 12, thus both elements seems to be in ascending order, hence, no swapping will occur. 12 also stored in a sorted sub-array along with 11

Third Pass:

Now, two elements are present in the sorted sub-array which are **11** and **12**

Moving forward to the next two elements which are 13 and 5

11	12	**13**	**5**	6

Both 5 and 13 are not present at their correct place so swap them

11	12	**5**	**13**	6

After swapping, elements 12 and 5 are not sorted, thus swap again

11	**5**	**12**	13	6

Here, again 11 and 5 are not sorted, hence swap again

5	**11**	12	13	6

Here, 5 is at its correct position

Fourth Pass:

Now, the elements which are present in the sorted sub-array are **5, 11** and **12**

Moving to the next two elements 13 and 6

5	11	12	**13**	**6**

Clearly, they are not sorted, thus perform swap between both

5	11	12	**6**	**13**

Now, 6 is smaller than 12, hence, swap again

5	11	**6**	**12**	13

Here, also swapping makes 11 and 6 unsorted hence, swap again

5	**6**	**11**	12	13

Finally, the array is completely sorted.

Task 4:
Given an array arr[ ] of length N consisting cost of N toys and an integer K the amount with you. The task is to find maximum number of toys you can buy with K amount.
Example 1:
Input:
N = 7
K = 50
arr[] = {1, 12, 5, 111, 200, 1000, 10}
Output: 4
Explanation: The costs of the toys
you can buy are 1, 12, 5 and 10.
Example 2:
Input:
N = 3
K = 100

arr[] = {20, 30, 50}
Output: 3
Explanation: You can buy all toys

Task 5:
The store manager at a shipping company needs to organize a set of large containers in the warehouse in order of the time they are to be shipped out. The warehouse is almost full, allowing space for only one additional container at a time. The cost of exchanging (moving) container is significantly higher than the cost of comparing them.
Implement suitable algorithm to efficiently arrange the containers. Use a user-generated array representing the initial order of container shipment times. After sorting, display the final order of the containers.

Task 6:
Consider the following array of score of a competition of multiple players
Scores = {197,98,101,189,90,202}
Sort the scores in descending order using
(A) Bubble Sort
(B) Insertion Sort
(C) Selection Sort

Task 7:
Given an array of int as arr[]. Sort given int array using Modified Bubble Sort and display the sorted array. Now apply selection sort in same array and discuss which sorting algorithm is best.
Array = {3,89,9056,76,45,86,109,203,87,43,23,67}

Task 8:
Create an array of 50 random integers, and apply the bubble sort algorithm. Do the following activities: Note: You can generate random numbers using this link https://numbergenerator.org/randomnumberlist-1-100 1.

Display the sorted array 2. Display the number of comparisons performed by the inner array 3. Now change the size of the array to 100 random numbers and repeat point numbers 1 and 2. 4. Take the completely sorted array of 50 numbers and repeat point numbers 1 and 2. 5. Can we reduce the number of comparisons for the sorted array? How? Write a code for it. and repeat point numbers 1 and 2.