Seatwork 7.1	
Using Sorting Algorithms	
Course Code: CPE010	**Program:** Computer Engineering
Course Title: Data Structure and Algorithm	**Date Performed:**
Section: CPE21S4	**Date Submitted:**
Name(s): Francis Nikko I. Andoy	**Instructor: JimLord Quejado**

Objectives

Answer the following questions:

1. What is sorting algorithm?

Sorting algorithm is arranging the data in specific order, such as descending and ascending. This can also help us analyze and present the data in a good manner. It is also the one of the fundamentals in data structure and algorithm. There are also other kinds of sorting technique, such as the bubble sort, selection sort, insertion sort, etc.

2.Where can sorting algorithms be used?

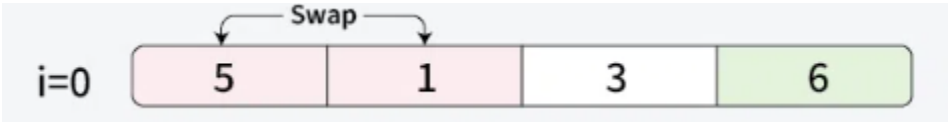
We can use sorting algorithm on some unsorted data, when we want to see the organized data and be able to analyze it. Common sorting algorithms are often essential in various applications, including data processing and management. According to, (Algorithm Examples, 2024) A key application area of sorting algorithms in e-commerce is product recommendations. By analyzing past user behavior and preferences, these algorithms generate a sorted list of products that the customer is likely to be interested in.

3. Explain the different types of sorting algorithms.

The 3 basics of sorting algorithm are bubble sort, selection sort, and insertion sort.

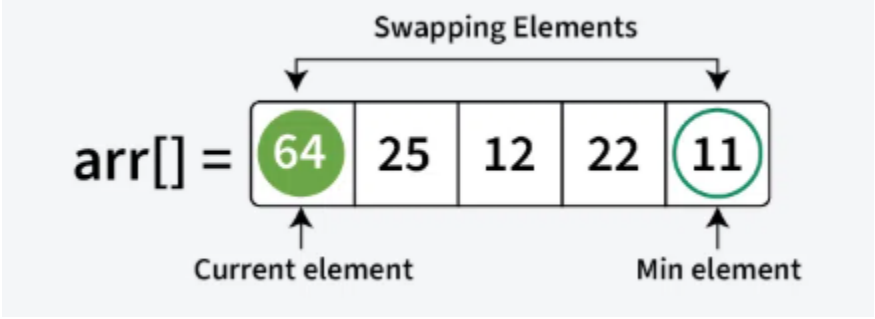
In bubble sorting, we first start at the index then compare it to the next element in the array

according to,



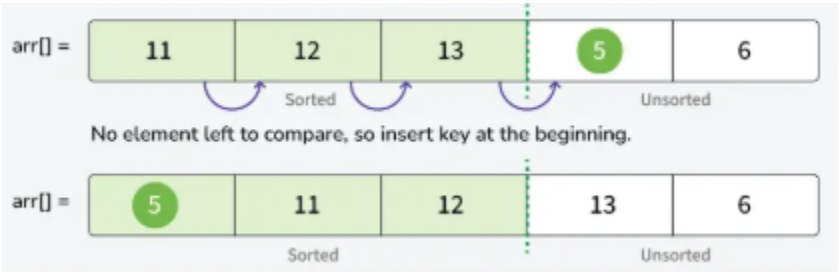
(GeeksforGeeks, 2025c) this is when the first index at the array is greater than the number it will be compared to. Then after that the iteration will just go back to the first element, until the array is sorted.

In selection, this is a sorting algorithm that starts at the first index. it will iterate all the elements at the array to find the smallest value, then swap the first index and the smallest value. According to, (GeeksforGeeks, 2025d)



In the insertion sorting, we will start at the first element then compare it to the next index after just like bubble sort it will compare then the condition. But the difference is that insertion will iterate the whole array just ONE not like bubble sort.

In accordance to,



(GeeksforGeeks, 2025d)

4. Give sample programs in C++ that uses sorting algorithms, specifically selection sort, insertion sort, and bubble sort. Explain how the programs work.

Bubble sort:

```
#include <iostream>

#include <vector>

using namespace std;


void bubbleSort(vector<int>& arr) {

int n = arr.size();

for (int i = 0; i < n - 1; i++) {

for (int j = 0; j < n - i - 1; j++) {

if (arr[j] > arr[j + 1]) {

swap(arr[j], arr[j + 1]);

}
```

```
}

}

}


int main() {

vector<int> arr = {5, 1, 4, 2, 8};

bubbleSort(arr);

for (int num : arr) {

cout << num << " ";

}

return 0;

}
```

In here we are using a bubble sort.

Selection sort:

```
#include <iostream>

using namespace std;


// Function to perform Selection Sort

void selectionSort(int arr[], int n) {

    for (int i = 0; i < n - 1; i++) {

        // Find the minimum element in the unsorted part

        int minIndex = i;

        for (int j = i + 1; j < n; j++) {

            if (arr[j] < arr[minIndex]) {

                minIndex = j;

            }

        }
```

```
        // Swap the found minimum element with the first element
        swap(arr[minIndex], arr[i]);
    }
}

// Function to print the array
void printArray(int arr[], int n) {
    for (int i = 0; i < n; i++) {
        cout << arr[i] << " ";
    }
    cout << endl;
}

int main() {
    int arr[] = {64, 25, 12, 22, 11};
    int n = sizeof(arr) / sizeof(arr[0]);

    cout << "Original array: ";
    printArray(arr, n);

    selectionSort(arr, n);

    cout << "Sorted array: ";
    printArray(arr, n);

    return 0;
}
```

insertion sort

```
#include <iostream>
using namespace std;

// Function to perform Insertion Sort
void insertionSort(int arr[], int n) {
for (int i = 1; i < n; i++) {
int key = arr[i]; // Current element to be placed
int j = i - 1;

// Shift elements of the sorted part to make space for the key
while (j >= 0 && arr[j] > key) {
arr[j + 1] = arr[j];
j--;
}
arr[j + 1] = key; // Place the key in its correct position
}
}

// Function to print the array
void printArray(int arr[], int n) {
for (int i = 0; i < n; i++) {
cout << arr[i] << " ";
}
cout << endl;
}
```

```
int main() {

int arr[] = {12, 11, 13, 5, 6};

int n = sizeof(arr) / sizeof(arr[0]);



cout << "Original Array: ";

printArray(arr, n);



insertionSort(arr, n);



cout << "Sorted Array: ";

printArray(arr, n);



return 0;

}
```