Data Structures and Algorithms — Lab 2

# Objective

* Simple sorting algorithms — Selection sort
* Simple searching algorithms — Linear search, binary search
* Working with classes

# Selection Sort

Selection sort is a sorting algorithm in which we repeatedly find the next largest (or smallest) element in the array and move it to its final position in the sorted array. Assume that we wish to sort the array in increasing (ascending) order, i.e., the smallest element at the beginning of the array and the largest element at the end. We begin by selecting the smallest element and moving it to the lowest index position. We can do this by swapping the element at the lowest index and the smallest element. We then reduce the effective size of the array by one element and repeat the process on the greater (sub)array. The process stops when the effective size of the array becomes 1 (an array of 1 element is already sorted).

## Pseudocode

Input: An unsorted *array, A of N elements*

Output: Sorted array, A

For I = 0:N-1

SmallSub = I

For J = I+1:N-1

If A[J] < A[SmallSub]

SmallSub = J

End-If

End-For

Swap ( A[I], A[SmallSub] )

End-For

# Linear Search

In computer science, linear search or sequential search is a method for finding a particular value in a list that consists of checking every one of its elements, one at a time and in sequence, until the desired one is found.

## Pseudocode

Input: An unsorted *array,* *A of N elements,* and *value* to be searched

Output: Index of searched element or -1 if not found

For I = 0:N-1

If ( A[I] == Value )

Return [I];

End-if

End-For  
return -1;

# Binary Search

Binary search is another searching algorithm used to search a specific value (or index of value) from the *sorted array*.

In binary search, we first compare the *value to be searched* with the item in the middle position of the array. If there's a match, we can return immediately. If the key is less than the middle key, then the item sought must lie in the lower half of the array; if it's greater than the item sought must lie in the upper half of the array. So we repeat the procedure on the lower (or upper) half of the array.

## Pseudo Code

Input: An *Sorted* *array,* *A of N elements* and *value* to be searched

Output: Index of searched element or -1 if not found

low = 0, high = N-1;

while (low <= high)

mid = (low + high)/2;

If ( A[mid] == Value )

Return mid;

Else-if ( A[mid] < Value )

low = mid + 1;

Else

high = mid – 1;

End-if

End-For  
return -1;

# Tasks

Students are required to complete the following tasks in lab timings.

## Task 1

Implement the Selection Sort Algorithm.

## Task 2

Create a C++ class named *Student*, with the following PRIVATE attributes:

1. regNo: int (consider that registration number is a max 4-digit number)
2. CGPA: float

Your task is to implement 10 objects of students. Initialize all objects (you can use the rand() function to make things easier) and perform the following Search algorithms on them based on CGPA.

1. **Linear Search**
2. **Binary Search**

**These functions MUST be implemented as PUBLIC member functions of the class Student.**

**NOTE:** To perform Binary Search, the array must be sorted.