

Statement Purpose:

Purpose of this Lab is to familiarize the students with the code of linear search and binary search. Another aim is to teach the students the difference between the running times of linear search and its more efficient counterpart: binary search. The students are given small task related to linear search and binary search which they complete during the lab session under the supervision of the lab instructor. This helps them understand the concepts well which they learn in their lectures.

Activity Outcomes:

The students will learn how to write code for linear search and binary search methods. They will also understand how to find the difference between the running times of these two approaches.

Theory Review (10 Minutes):

Linear Search:

The linear search approach compares the key element **key** sequentially with each element in the array. It continues to do so until the key matches an element in the array or the array is exhausted without a match being found. If a match is made, the linear search returns the **index** of the element in the array that matches the key. If no match is found, the search returns **-1**.

Code for Linear Search Method

```
public static int linearSearch(int[] array, int key) {
    for(int i = 0; i < array.length; i++) {
        if(key == array[i])
            return i; //If found, return index of that element
    }
    return -1; //if not found, return -1
}
```



Binary Search:

Binary search is the other common search approach for a list of values. For binary search to work, the elements in the array must already be ordered. Assume that the array is in ascending order. The binary search first compares the key with the element in the middle of the array.

Consider the following three cases:

- If the key is equal to the middle element, the search ends with a match.
- If the key is less than the middle element, you need to continue to search for the key only in the first half of the array.
- If the key is greater than the middle element, you need to continue to search for the key only in the second half of the array.

Clearly, the binary search method eliminates at least half of the array after each comparison.

Code for Binary Search Method

```
public static int binarySearch(int[] array, int key) {
    int low = 0, high = array.length-1, mid = 0;

    while (low <= high) {
        mid = (low + high)/2;

        if (key == array[mid])
            return mid;
        else if (key < array[mid]) // change high
            high = mid - 1;
        else // change low
            low = mid + 1;
    }
    return -1; // if key not found above
}
```



LAB EXERCISE: (70 Minutes)

1. Write code in the method `linearSearch()` in "SearchMethods" class which finds the element in an array using **linear search** approach and returns its index to the calling main method, otherwise it returns -1.

Sample Run:

```
Output - SearchProject (run)
run:
*****
***** Search Menu *****
*****
| 1. Linear Search          |
| 2. Binary Search         |
| 3. Exit the program      |
-----

> Enter your choice: 1

> Please enter a value to search for (between 0 and 100 million): 1201101

> The value, 1201101, was not found in the array.
> This search took 20985622 nanoseconds to complete (or 20 milliseconds).

*****
***** Search Menu *****
*****
| 1. Linear Search          |
| 2. Binary Search         |
| 3. Exit the program      |
-----

> Enter your choice: 1

> Please enter a value to search for (between 0 and 100 million): 1211123

> The value, 1211123, was found at index 1212648 in the array.
> This search took 2932155 nanoseconds to complete (or 2 milliseconds).
```



2. Write code in the method `binarySearch()` in "SearchMethods" class which finds the element in an array using **binary search** approach and returns its index to the calling main method, otherwise it returns -1.

Sample Run:

```
Output - SearchProject (run) ✖
run:
*****
***** Search Menu *****
*****
| 1. Linear Search          |
| 2. Binary Search         |
| 3. Exit the program      |
-----

> Enter your choice: 2

> Please enter a value to search for (between 0 and 100 million): 43210

> The value, 43210, was found at index 43319 in the array.
> This search took 13999 nanoseconds to complete (or 0 milliseconds).

*****
***** Search Menu *****
*****
| 1. Linear Search          |
| 2. Binary Search         |
| 3. Exit the program      |
-----

> Enter your choice: 2

> Please enter a value to search for (between 0 and 100 million): 12345

> The value, 12345, was not found in the array.
> This search took 10731 nanoseconds to complete (or 0 milliseconds).
```

