**DHA Suffa University**



**Department of Computer Science**

**CS 201L – Data Structures and Algorithms Lab**

**Fall 2019**

**Lab 01 – Searching Algorithms**

**Objective:**

To learn implementation of Searching Algorithms (Linear Search, Binary Search, Probabilistic Search, Sentinel Search).

Linear Search

Linear search or sequential search is a method for finding a target value within a list. It sequentially checks each element of the list for the target value until a match is found or until all the elements have been searched. Linear search runs in at worst time and makes at most *n* comparisons, where *n* is the length of the list.

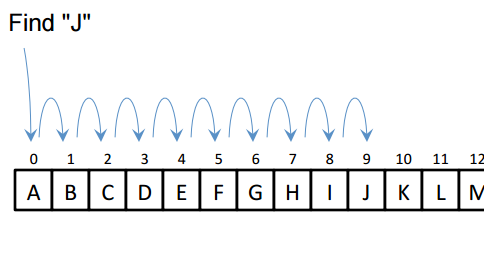
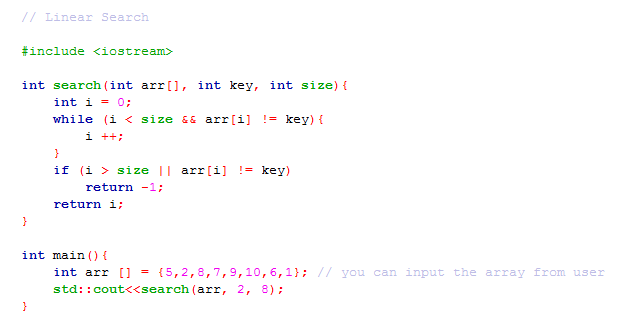


Figure 1.1: Linear Search

**Lab Task 1:**

Implement an abstract class Search with pure virtual function search(), publically inherit LinearSearch, SentinelSearch, ProbabilisticSearch and BinarySearch classes from the Search class.

Following code demonstrates the implementation of Linear Search Algorithm in C++.



Sentinel Search

To reduce overhead of checking the list length, the value to be searched can be appended to the list at the end (or beginning in case of Reverse Search) as a “sentinel value”. A sentinel value is one whose presence guarantees the termination of a loop that processes structured (or sequential) data. Thus on encountering a matching value, its index is returned. The calling function can then determine if the returned index is a valid one or not. Though the optimization resulted in isn’t much, it reduces the overhead of checking if the index is within limit in each step.

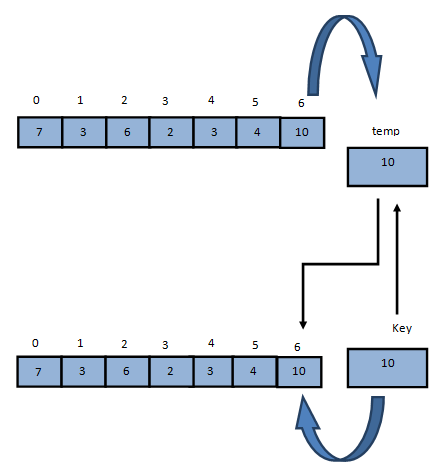
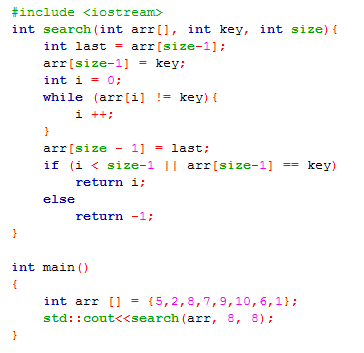
****

Figure 1.2: Sentinel Search

Following code demonstrates the implementation of Sentinel Search Algorithm in C++.



Binary Search

Binary search algorithms searches the element in a sorted array by comparing the given element with the middle element of the search space, if element is smaller than the middle element, the search space is limited to the left half of the middle element. In other case the search space is limited to right half of the middle element. This operation is carried out repeatedly until element is found or search space becomes empty.

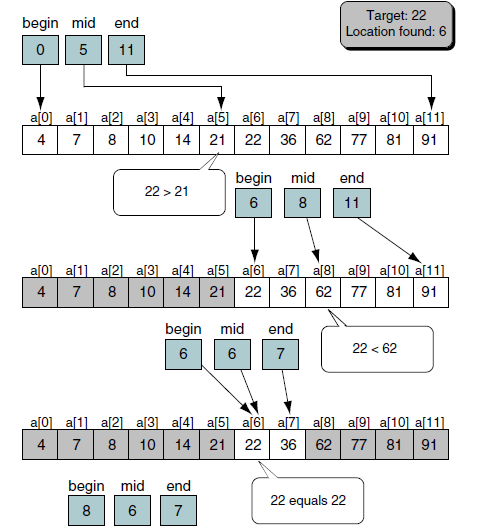
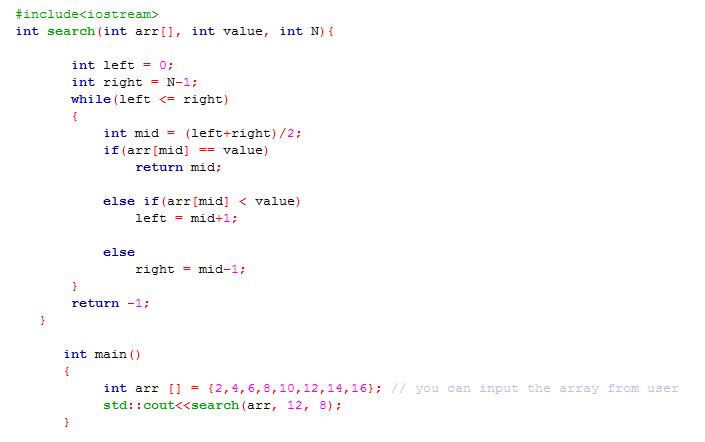


Figure 1.3: Binary Search

Following code demonstrates the implementation of Binary Search Algorithm in C++.



Probabilistic Search

Data is arranged with most probable items at the beginning and the least probable items at the end & useful when relatively few elements are the frequently searched. To ensure that the probability ordering is correct over time, in each search, exchange the located element with the element immediately before it in the array**.**

**Search Key: 7**

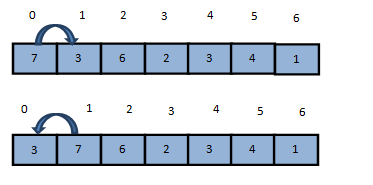
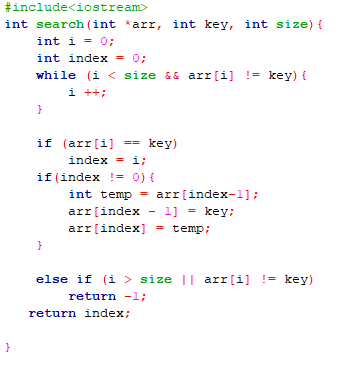
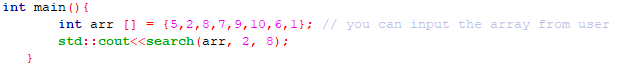
****

Figure 1.4: Probabilistic Search

Following code demonstrates the implementation of Probabilistic Search Algorithm in C++.





**Lab Task 2**:

Using the searching algorithms discussed in lab, write a program that will search the given name of employee from a predefined array of names of size N (you can assume any size).

### 

**Assignment 01:**

Given an input file, write a program that will read the file in a dynamic array of employee. Find a record of employee using the Employee Name column, from the defined multidimensional dynamic array using **each search algorithm** discussed in the lab. Your program should identify whether a record is present in the defined dynamic array or not.

# **Submission Guidelines**

* **Write C++ code , separate function for each operation.**
* **Place your file in a folder named with your rollNo (cs172xxx) where xxx is your 3 digit rollno.**
* **Upload it on LMS.**

**Note:**

**Remember -100 policy for plagiarism.**