Data Structures Lab 04(a)

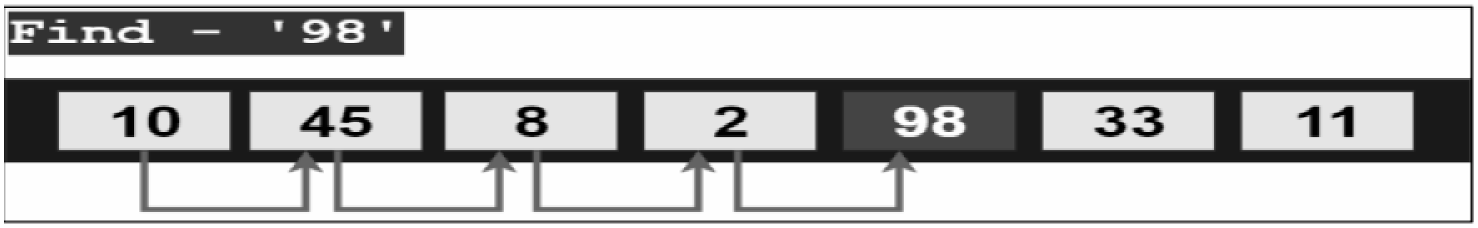
Course: Data Structures (CL2001) **Semester**: Fall 2024
Instructor: Sameer Faisal **T.A**: N/A

Note:
- Maintain discipline during the lab.
- Listen and follow the instructions as they are given.
- 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.

Searching Algorithms:

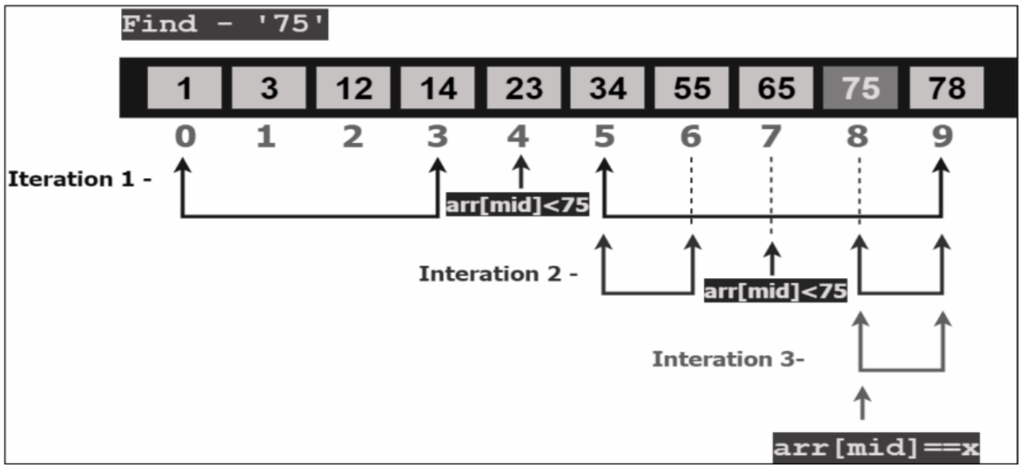
Linear Search

Linear search is a very simple search algorithm. In this type of search, a sequential search is made over all items one by one. Every item is checked and if a match is found then that particular item is returned, otherwise the search continues till the end of the data collection.



Binary Search

Binary search algorithm falls under the category of interval search algorithms. This algorithm is much more efficient compared to linear search algorithm. Binary search only works on sorted data structures. This algorithm repeatedly target the center of the sorted data structure & divide the search space into half till the match is found.



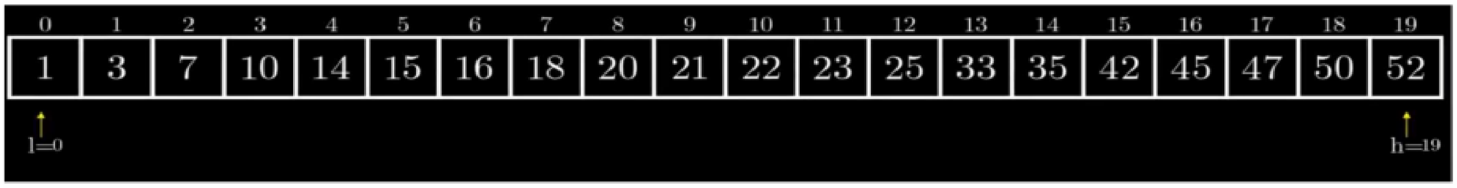
Pseudocode

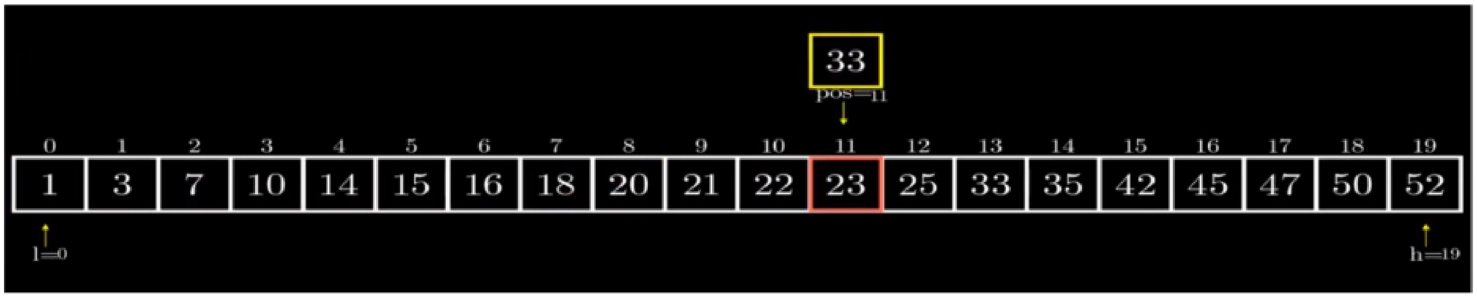
```
• Take input array, left, right & x
• START LOOP – while(left greater than or equal to right)
    – mid = left + (right-left)/2
    – if(arr[mid]==x) then
        • return m
    – else if(arr[mid] less than x) then
        • left = m + 1
    – else
        • right= mid – 1
• END LOOP
• return -1
```

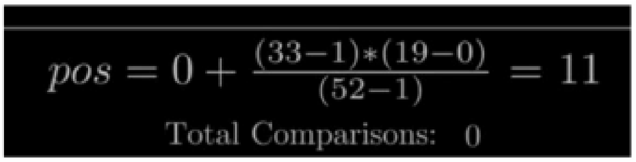
Interpolation Search

The Interpolation Search is an improvement over Binary Search for instances, where the values in a sorted array are uniformly distributed. Interpolation constructs new data points within the range of a discrete set of known data points. Binary Search always checks the value at middle index. But, interpolation search may check at different locations based on the value of element being searched. For interpolation search to work efficiently the array elements/data should be sorted and uniformly distributed.

$$Position = startindex + \frac{(element - Arr[startindex]) * (end\ value\ index - start\ value\ index)}{Arr[endindex] - Arr[startindex]}$$

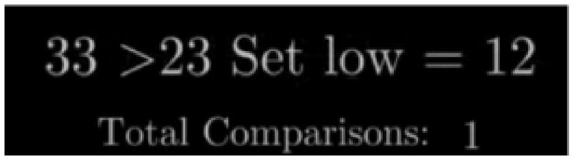






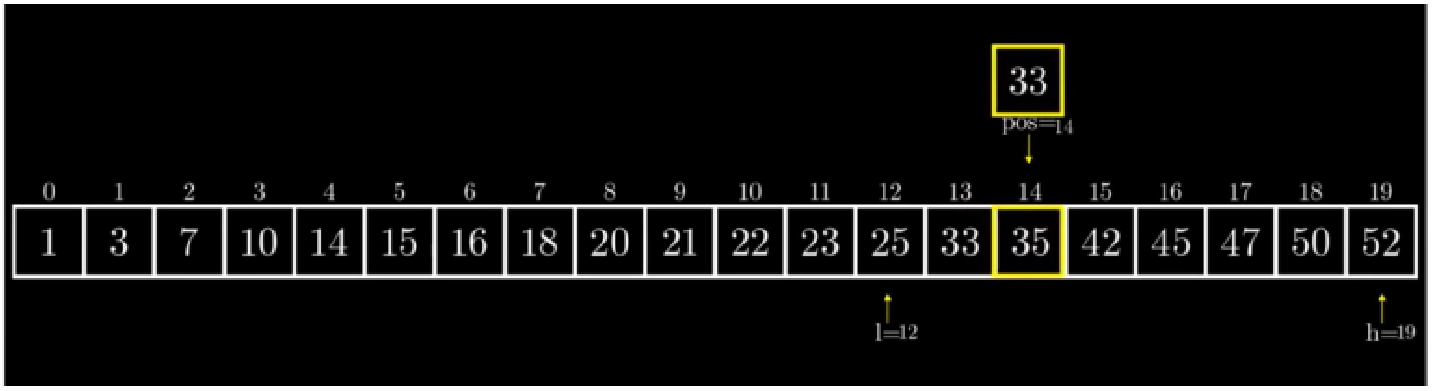
$$pos = 0 + \frac{(33-1)*(19-0)}{(52-1)} = 11$$

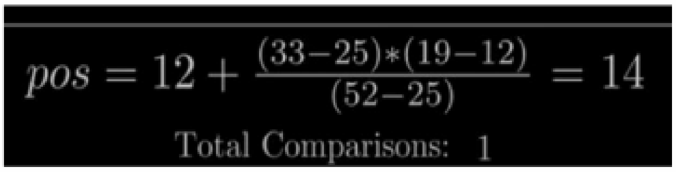
Total Comparisons: 0



33 >23 Set low = 12

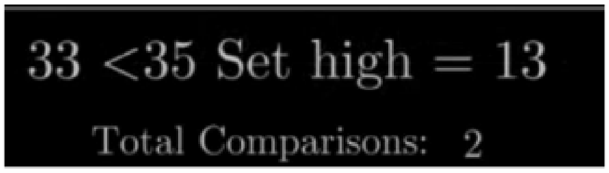
Total Comparisons: 1





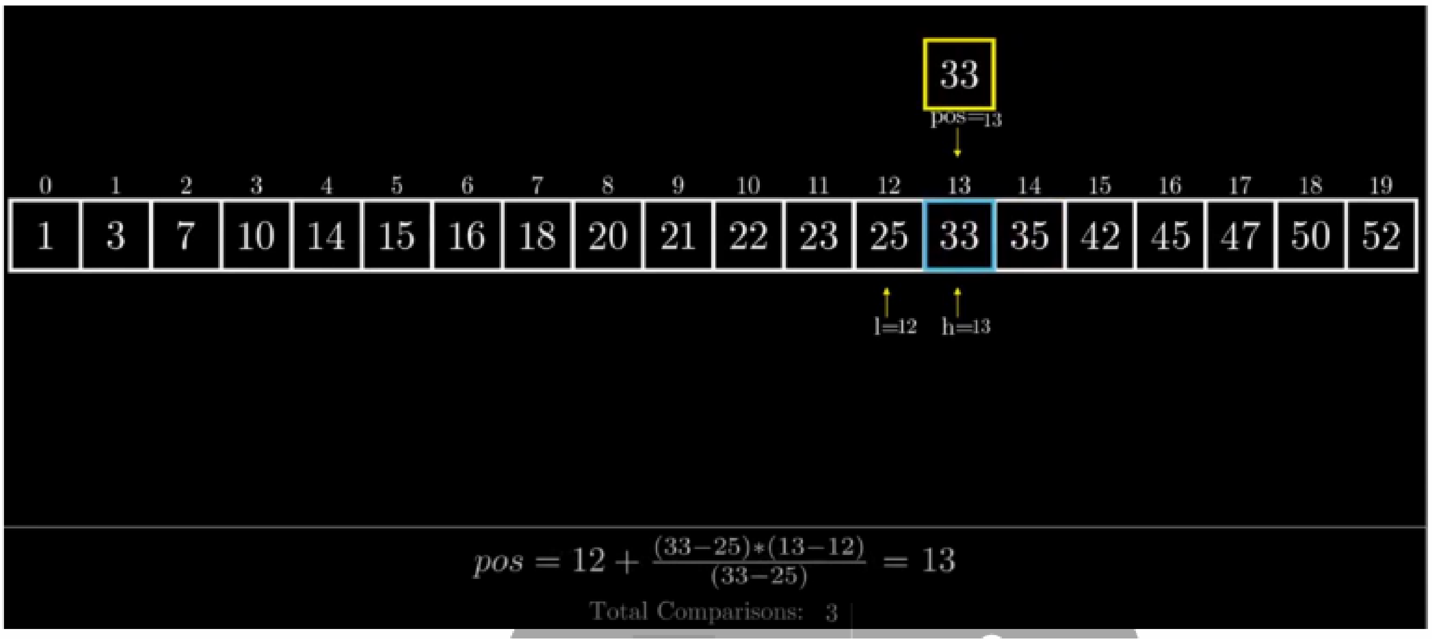
$$pos = 12 + \frac{(33-25)*(19-12)}{(52-25)} = 14$$

Total Comparisons: 1



33 <35 Set high = 13

Total Comparisons: 2



$$pos = 12 + \frac{(33-25)*(13-12)}{(33-25)} = 13$$

Total Comparisons: 3

Pseudocode

```
1. start = 0 & end = n-1
2. calculate position to start searching at using formula provided above.
3. If A [pos] == Element, element found at index pos.
4. Otherwise if element > A[pos] we make start = pos + 1
5. Else if element < A[pos] we make end = pos -1
6. Do steps 2,3, 4, 5, While : start <= end && element >= A[start] && element =< A[end]
```

– Start <= end - that is until we have elements in the sub-array.
– Element >= A[start] - element we are looking for is greater than or equal to the starting element of sub-array we are looking in.
– Element =< A[end] - element we are looking for is less than or equal to the last element of sub-array we are looking in.

Note: An array is considered as uniformly distributed when the difference between the elements is equal or almost same. Example 1: 1,2,3,4,5,6 (Difference is 1).

Lab Exercises:

1. You’ve been given an array of numbers representing employee IDs. Your task is to identify the employee whose ID matches the last two digits of your roll number. If your roll number’s last two digits are not present in the array, insert the missing value in its correct position within the array. You must use binary search to locate the position of that value within the array.

2. Your team has been given a large dataset (input by user) of sorted, uniformly distributed account balances. If the data is not sorted, you have to sort it first. If the data is not uniformly distributed after you apply sorting (if necessary) you can prompt an error. Your manager has asked you to implement Interpolation Search because it estimates the position of the target value based on the data distribution. This will allow the search to "jump" closer to the target in fewer iterations.