Data Structures and Algorithms

Search Techniques

Engr. Bushra Tahir
Department of Electrical Engineering
Iqra National University

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.

Linear Search

Pseudocode

```
procedure linear_search (list, value)
  for each item in the list
    if match item == value
      return the item's location
    end if
  end for
```

Binary Search

Binary search is a fast search algorithm with run-time complexity of O(log n). This search algorithm works on the principle of divide and conquer. For this algorithm to work properly, the data collection should be in the sorted form.

Binary Search

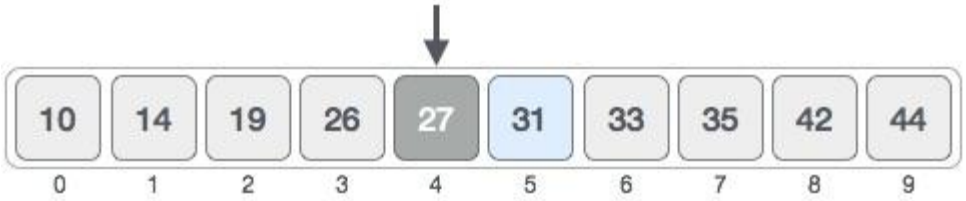
- Binary search looks for a particular item by comparing the middle most item of the collection.
- If a match occurs, then the index of item is returned.
- If the middle item is greater than the item, then the item is searched in the sub-array to the left of the middle item.
- Otherwise, the item is searched for in the sub-array to the right of the middle item. This process continues on the sub-array as well until the size of the subarray reduces to zero.

How Binary Search Works?

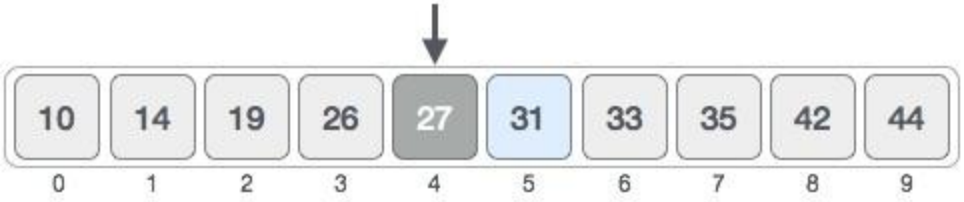
- For a binary search to work, it is mandatory for the target array to be sorted. We shall learn the process of binary search with a pictorial example.
- The following is our sorted array and let us assume that we need to search the location of value 31 using binary search.

How Binary Search Works?

- First, we shall determine half of the array by using this formula −
- mid = low + (high - low) / 2
- Here it is, 0 + (9 - 0) / 2 = 4 (integer value of 4.5). So, 4 is the mid of the array.



How Binary Search Works?



- compare the value stored at location 4, with the value being searched, i.e. 31.
- We find that the value at location 4 is 27, which is not a match. As the value is greater than 27 and we have a sorted array, so we also know that the target value must be in the upper portion of the array.

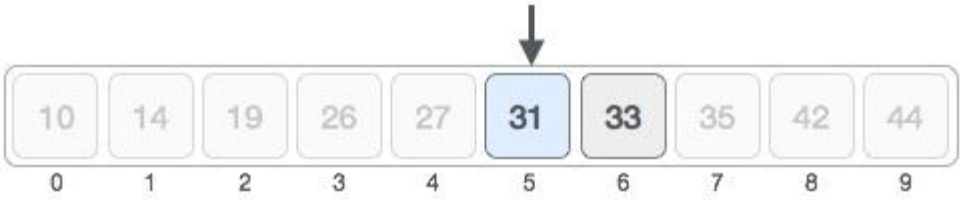
How Binary Search Works?

- We change our low to mid + 1 and find the new mid value again
- low = mid + 1 , mid = low + (high - low) / 2
- Our new mid is 7 now. We compare the value stored at location 7 with our target value 31.

How Binary Search Works?

The value stored at location 7 is not a match, rather it is more than what we are looking for. So, the value must be in the lower part from this location.

Hence, we calculate the mid again. This time it is 5.



How Binary Search Works?

- We compare the value stored at location 5 with our target value. We find that it is a match.

We conclude that the target value 31 is stored at location 5. Binary search halves the searchable items and thus reduces the count of comparisons to be made to very less numbers.

Interpolation Search

- Interpolation search is an improved variant of binary search.
- This search algorithm works on the probing position of the required value.
- For this algorithm to work properly, the data collection should be in a sorted form and equally distributed.

Interpolation Search

- Binary search has a huge advantage of time complexity over linear search.
- Linear search has worst-case complexity of O(n) whereas binary search has O(log n).

Positioning in Binary Search

- In binary search, if the desired data is not found then the rest of the list is divided in two parts, lower and higher. The search is carried out in either of them.

Even when the data is sorted, binary search does not take advantage to probe the position of the desired data.

Position Probing in Interpolation Search

- Interpolation search finds a particular item by computing the probe position.

If a match occurs, then the index of the item is returned

Position Probing in Interpolation Search

- To split the list into two parts, we use the following method –

mid = Lo + ((Hi - Lo) / (A[Hi] - A[Lo])) * (X - A[Lo])

where –

A = list

Lo = Lowest index of the list

Hi = Highest index of the list

A[n] = Value stored at index n in the list

Position Probing in Interpolation Search

- If the middle item is greater than the item, then the probe position is again calculated in the sub-array to the right of the middle item. Otherwise, the item is searched in the subarray to the left of the middle item.
- This process continues on the sub-array as well until the size of subarray reduces to zero.

Runtime Complexity

- Runtime complexity of interpolation search algorithm is **O(log (log n))** as compared to **O(log n)** of BST in favorable situations.