Searching Algorithms

Introduction to Searching Algorithm

Searching is an operation or a technique that helps find the place of a given element or value in the list.

Any Search is said to be successful or unsuccessful depending upon Whether the element that is being Searched is found or not. Some of the standard Searching technique that is being followed in data structure is listed below:-

1. Linear Search

2. Binary Search

→ Linear Search

is a very basic and simple search Algorithm In Linear search, We search an element or value in a given array by traversing the array from the starting till the desired element or value is found.

Linear search is applied or unsorted or unordered lists, When there are fewer elements in a list

Eg:-

10	14	19	26	27	31	33	35	42	44

=

33

Algorithm

```
Linear Search (Array A, Value x)
Setep 1: Set i to 1
Step 2: if i>n then go to Step 7
Step 3: if A[i]=x then go to step 6
Step 4: Set i to i+1
Step 5: go to step 2
Step 6: Print Element x Found at index i and
        go to step 8
Step 7: Print element not found
Step 8: Print element not found
Step 9: Exit
```

Program:-

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

* Binary Search
- is used to With Sorted array or list.

In Binary Search We pollow the pollowing steps:-

1. We start by comparing the element to be searched with the element in the middle of the list/array
2. If we get a match, We return the index of the middle element
3. If We do not get a match, We check Whether the element to be searched is less or Greater than in value than the middle Element
4. If the element/number to be searched is greater in value than the middle no then We pick the elements on the right side of the middle elements and start again from the step1
5. If the element/no to be searched is lesser in value than the middle no, then We pick the elements on the left side of the middle element and start again from the step 1.

* How Binary search Works?
The following is our sorted array and let us assume that We need to search the location of value 31 using binary search

10	14	19	26	27	31	33	35	42	44
0	1	2	3	4	5	6	7	8	9

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

10	14	19	26	$\boxed{27}$	31	33	35	42	44
0	1	2	3	4	5	6	7	8	9

Now we 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 sorted array, So We also know that the target value must be in the upper portion of the array

10	14	19	26	27	31	33	35	42	44
0	1	2	3	4	5	6	7	8	9

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 sorted at location 7 With our target value 31

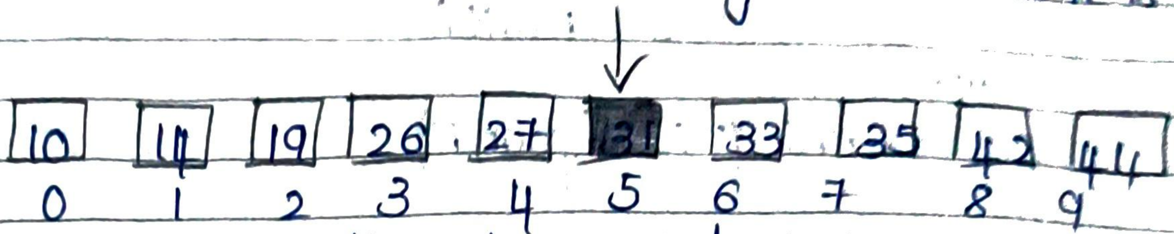
$\downarrow$

10	14	19	26	27	31	33	35	42	44
0	1	2	3	4	5	6	7	8	9

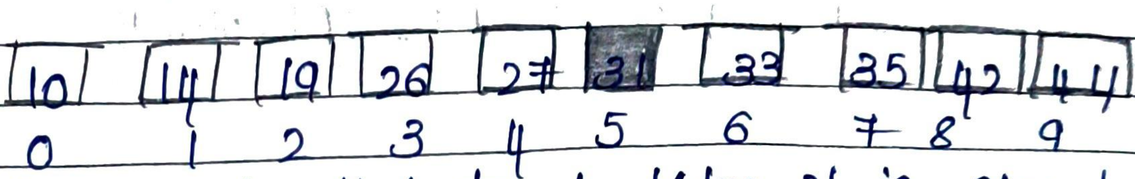
The value stored at location 7 is not a match, rather it is more than What We are looking for so the value must be in the lower part from this location

10	14	19	26	27	31	33	35	42	44
0	1	2	3	4	5	6	7	8	9

Hence, We calculate the mid Again The time it is 5



We compare the Value sorted at location 5 With our target Value We Find that it is a match



We conclude that target Value 31 is stored at location 5

* Program on Binary search

```
def binarysearch(array, x, low, high):
  While low <= high:
    mid = low + (high -low)
    if x == array [mid]:
        return mid
    elif x > array [mid]:
        low = mid + 1
  else:
        high = mid - 1.
   return -1
array = [3, 4, 5, 6, 7, 8, 9 ]
x = 4
result = binarysearch(array, x, 0, len(array) -1)
if result != -1:
    print ("Element is present at index" +
                    str(result))
else:
    Print ("Not found")
```