Binary Search

Algorithm, Best case, worst case and average case

Divide and conquer techniques

- Divide and conquer is an algorithm design technique.

- This type of algorithm breaks down a problem into two or more sub - problems of the same type, until the problem is simple enough to be solved directly.

- We basically ignore half of the elements just after one comparison in binary search.

Divide and conquer techniques

- 1. Compare x with the middle element.
- 2. If x matches with the middle element, we return the middle index.
- 3. Else If x is greater than the middle element, then x can only lie in the right half of the subarray after the middle element. So we choose the right subarray
- 4. Else (x is smaller) we choose the left subarray.

10	**20**	**30**	**40**	**50**	**60**	**70**	**80**	**90**	**100**	**110**	**120**	**130**	**140**	**150**	**160**
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Low =0; high=15; keyvalue=10;
// more than one element, apply divided and conquer techniques
Mid = (low + high)/2 = (0+15)/2 = 7
Since , 10 < 80, we need to find only the left sub array, i.e, mid-1

Low=0; high = mid-1 = 7-1=6; keyvalue=10;
// more than one element, apply divided and conquer techniques
Mid = (0+6)/2 = 3;
Again, 10 < 40, we need to find only the left most sub arrayi.e, mid-1;

10	20	30	40	50	60	70	80	90	100	110	120	130	140	150	160
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Low= 0; high=mid-1; =3-1=2;keyvalue=10;
// more than one element, apply divided and conquer techniques
Mid = (0+2)/2 = 1;
Still again, 10 < 20, we need to find the left most subarray i.e., mid-1;

Low=0; high=mid-1 = 1-1 =0;keyvalue=10;
No need to apply divide and conquer techniques, directly apply first part of the algorithm

```
Algorithm BinarySearch(a, low, high, keyvalue)
{
 if(low==high)
{
   if(a[low]==keyvalue)
    return low;
  else
     return -1;
}
else
{
   mid = (low + high)/2;
   if(a[mid]== keyvalue)
       return mid;
  elseif(keyvalue> a[mid])
           return BinarySearch(a, mid+1, high, keyvalue);
 else
     return BinarySearch(a, low, mid-1, keyvalue);
}
}// End of the algorithm
```

Law of trichotomy

- A > B
- A < B
- A == B

We know that
$n = 2^k$; $16 = 2^4$; $4 = \log_2 16$
$n/2 = 2^{k-1}$
$8 = 2^3$
$k = \log_2 n$

<u>Time complexity of the binary search algorithm</u>

if(n==1), t(1) =1 // if the array is having only one element
(i.e.,only one comparison)
otherwise
$t(n) = t(n/2) + 1$
This relation is called recurrence relation
$t(n) = t(n/2) + 1$;
$t(2^k) = t(2^{k-1}) + 1$; ………. (1)

put k=k-1 ; in equation (1)

$t(2^{k-1}) = t(2^{k-2}) + 1$;……..(2)

Substtitute the value of $t(2^{k-1})$ in equation (1)

i.e., $t(2^k) = t(2^{k-2}) + 1 + 1$ ……(3)
keep on substitute the value of k =1, 2, …

$t(2^k) = t(2^{k-3}) + 1 + 1 + 1$;
i.e., $t(2^k) = t(2^{k-3}) + 3(1)$;

…

$t(2^k) = t(2^{k-k}) + k(1)$

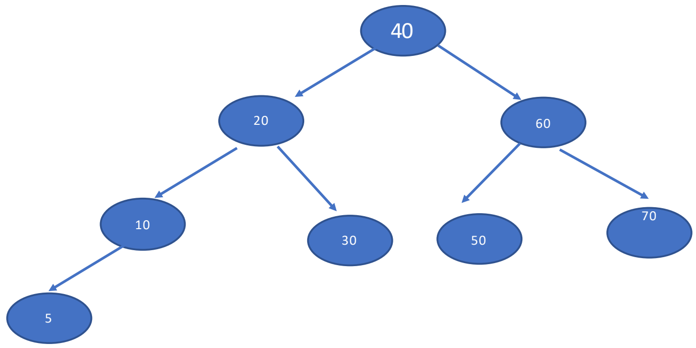
i.e., $t(2^k) = t(2^{k-k}) + k$ ….. (4)
since $t(2^{k-k}) = t(2^0) = t(1) = 1$
Equation (4) becomes
$t(2^k) = 1 + k$
ie., $t(n) = 1 + k = 1 + \log_2 n$

Average case of binary search algorithm

Ceil and floor functions

Ceil(2.78)=ceil(2.01) = 3;Floor(2.99)=floor(2.01) = 2

10, 20, 30 ,40, 50, 60 ,70



Average case = Total number of comparisons of all the elements / n

=(1 + 2 + 2 + 3 + 3 + 3 + 3 + 4)/7 = 21/8 = ceil(2.625) = 3 = logn