14-Dec-2021

14 December 2021 20:44

Linear Search :- $O(n)$

[10, 40, 30, 50, 15, 35, 60] $n = 17$

0 1 2 3 4 5 6

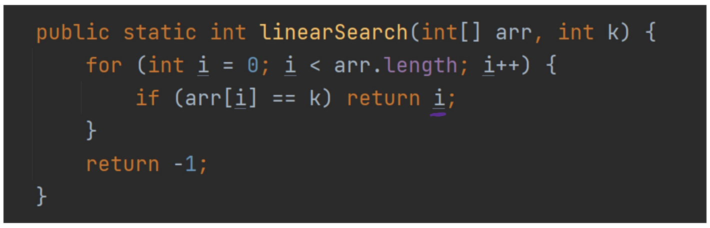
```
for (i=0 to arr.leng-1) {
    if (arr[i] == n) {
        return i;
    }
}
return -1;
}
```

0 1 2 3 4 5

5, 4, 3, 10, 1, 2

i = ~~0 1 2 3 4 5~~ 6

K = 11.



```
public static int linearSearch(int[] arr, int k) {
    for (int i = 0; i < arr.length; i++) {
        if (arr[i] == k) return i;
    }
    return -1;
}
```

Binary Search :-   Binary = 0, 1

- Element has to be in a sorted manner.

0 1 2 3 4 5

1, 3, 5, 10, 15, 20 $n = 17$

l

$\frac{(0+5)}{2} = 2$

$\frac{5+5}{2} = 5$ | 5 < 15, 4, ans | $\frac{3+5}{2} = 4$

[5, 6, 8, 10, 11, 15, 16, 18, 20, 23, 25, 28, 30]

0 1 2 3 4 5 6 7 8 9 10 11 12

$n = 11$

l = ~~0~~ 3

| if (n < arr[mid]) | if (n > arr[mid])

$x=11$

$l = \cancel{0}\ 3$

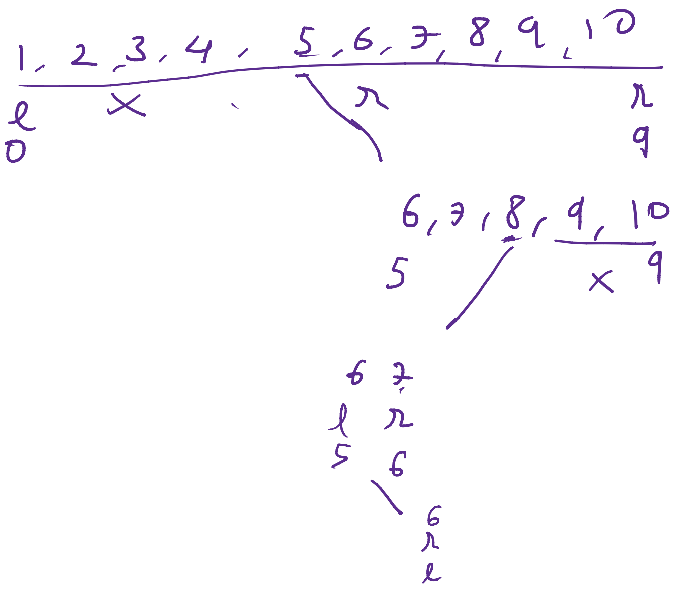
$r = \cancel{12}\ 5$

$mid = \cancel{6}\ \cancel{2}\ 4$

```
if ( x< arr[mid])
r = mid-1
```

```
if (x > arr[mid]
l= mid+1
```

$x = 7$



1000 00

$O(\log n)$

```
binSearch I ( arr, x ) {
  int l = 0, r = arr.length - 1;
  while ( l ≤ r ) { int mid = (l +r)/2;
    if (arr[mid] == x ) return mid;
  else if ( arr[mid] < x ) {
      l = mid+1;
  } else {
      r = mid - 1;
  }
  } return -1;
}
```

Binary Search Recursion :-

[0, 1, 2, 3, 4, 5, 6, 7]

=> we found the element

=> arr[mid] < x

=) arr[mid] > x

l > r return -1

```
bSR (arr, x, l, r) { if (l > r) return -1;
mid = (l + r)/2;
if (arr[mid] == x) return mid;
else if (arr[mid] < x) return bSR(arr, x, mid+1, r);
else return bSR(arr, x, l, mid-1);
}
```

l	r	mid
~~0~~	~~-1~~	
~~0~~	~~0~~	~~0~~
~~0~~	~~2~~	~~1~~
~~0~~	~~7~~	~~3~~

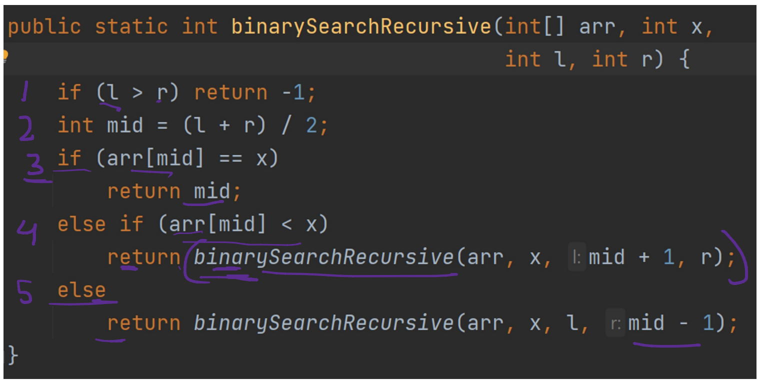
-1

1 ✓

~~1~~ ~~2~~ ~~5~~

~~1~~ ~~2~~ ~~5~~

~~1~~ ~~2~~ ~~5~~



```java
public static int binarySearchRecursive(int[] arr, int x,
                                         int l, int r) {
    if (l > r) return -1;
    int mid = (l + r) / 2;
    if (arr[mid] == x)
        return mid;
    else if (arr[mid] < x)
        return binarySearchRecursive(arr, x, l: mid + 1, r);
    else
        return binarySearchRecursive(arr, x, l, r: mid - 1);
}
```

[1, 2, 3, 4, 5, 6, 7, 8]
 0 1 2 3 4 5 6 7

x = 0

l = 0

r = 7