L7 - Binary Search

Thursday, August 20, 2020 10:00 AM

Given a sorted list of elements/numbers, find whether a ***key*** occurs in the list or not. If the ***key*** if found, it's index in the list, otherwise -1, is returned.

Algorithm:

1. Find the middle element of the ***current list***.
2. If the middle element is equal to the ***key***, return the key index.
3. If the middle element is greater than the ***key***, make current list equal to the lower half of the ***current list***. Repeat the Algorithm (from step 1.)
4. If the middle element is lesser than the ***key***, make current list equal to the upper half of the ***current list***. Repeat the Algorithm (from step 1.)
5. Otherwise, the ***key*** is not found.

Arr: 1 3 4 6 8 12 34 56 67
Key: 67

No. Iteration	start	end	mid	Arr[mid]
1	1 6	10	5	8
2	6 9	10	8	56
3	9	10	9	67

Arr: 1 3 4 6 8 12 34 56 67
Key: 75

No. Iteration	start	end	mid	Arr[mid]
1	1 6	10	5	8
2	6 9	10	8	56
3	9 10	10	9	67
4	10	10		

Arr: 1 3 4 6 8 12 34 56 67
Key: 34

No. Iteration	start	end	mid	Arr[mid]
1	1 6	10	5	8
2	6	10 8	8	56
3	6	8	7	34

Pseudo-Code
BinarySearch(Arr, key)
// Arr is sorted in the ascending order.

```
1.  let start = 1
2.  let end = Arr.length+1
3.  while start != end {
4.        let mid = start + (end - start)/2       // mid = (start + end)/2
5.            // assume integer division
6.        if (Arr[mid] == key)
7.            return mid
8.        if (Arr[mid] < key)
9.            start = mid + 1
10.       else if (Arr[mid] > key)
11.           end = mid
12. }
13. return -1
```

Worst Case Analysis:

Pseudo-Code

```
BinarySearch(Arr, key)
// Arr is sorted in the ascending order.
                                              Operations   WCF
 1.  let start = 1                                c1        1
 2.  let end = Arr.length+1                       c2        1
 3.  while start != end                           c3        log(n) (+ 1)
 4.       let mid = start + (end - start)/2       c4        log(n)
 5.            // assume integer division
 6.       if (Arr[mid] == key)                    c5        log(n)
 7.            return mid                         c6        1 (0)
 8.       if (Arr[mid] < key)                     c7        log(n)
 9.            start = mid + 1                    c8        log(n)
10.       else if (Arr[mid] > key)
11.            end = mid
12.  return -1                                    c9        1
```

// Either line 7 or line 12 will be executed

Worst case arises when the key does not occur in the list

T(n) = (c3+c4+c5+c7+c8)log(n) + (c1+c2+c3+c6+c9)

= c10*log(n) + c11

= **O(log(n)) Logrithmic**

$T(n) = c1$ if $n=0$

$T(n) = T(n/2) + O(1)$ if $n > 0$

$T(n) = T(n/2) + O(1)$ if $n > 0$