Binary Search

Sunday, March 29, 2020 8:14 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.

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
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. return -1
```

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

Worst Case Analysis:

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

		cost	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		
10.	else if (Arr[mid] > key)		
11.	end = mid		
12.	return -1	c8	1

// Either line 7 or line 12 will be executed

Worst case arises when either the key does not occur in the list or first/last element of the list is the key.

T(n) = (c3+c4+c5+c7)log(n) + (c1+c2+c3+c6+c8)
= c10log(n) + c11
= **O(log(n)) Logrithmic**

```
1. for I = 1 to n
2. let j = 1
3. let a = 1
4. let b = 1
5.         while j <= n
6.                 a = 10+5
7.                 b = a * 3
8.                 j = j * 2
```

		cost	Freq
1.	for I = 1 to n	c1	n+1
2.	let j = 1	c2	n
3.	let a = 1	c3	n
4.	let b = 1	c4	n
5.	while j <= n	c5	n(log(n)+1)
6.	a = 10+5	c6	nlog(n)
7.	b = a * 3	c7	nlog(n)
8.	j = j * 2	c8	nlog(n)