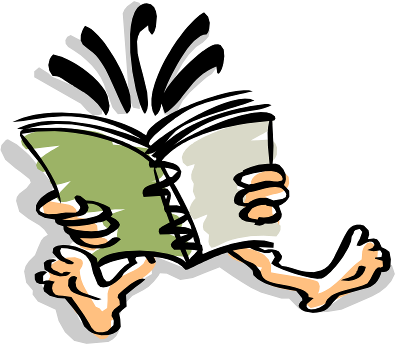
BINARY SEARCH ALGORITHM

Ref Book: Computer Algorithms , By Sartaj Sahni



Nirvruti Patel ,IT Department,UVPCE

Binary Search Algorithm

- Can only be performed on a sorted (non decreasing order) list !!!

- Uses *divide and conquer* technique to search list

Binary Search Algorithm

- Search item is compared with middle element of list
- If search item < middle element of list, search is restricted to first half of the list
- If search item > middle element of list, search second half of the list
- If search item = middle element, search is complete

Binary Search Algorithm

- BinSearch(a,i,l,x)

/ /a[i:l] –sorted array - Whether x is present or not

```
{
        if(l==i){
                 if(x==a[i])
                     return i;
                 else
                     return 0;
        }
        else{
                mid=(i+l)/2;
                if(x==a[mid])
                       return mid;
                else if(x<a[mid])
                       return BinSearch(a,I,mid-1,x);
                else
                       rerun  BinSearch(a,mid+1,l,x);

            }
}
```

Binary Search Algorithm

- Problem P is divided into one sub problem.
- Division takes only O(1) time.
- Answer to subproblem is answer to original problem P ; there is no need for any combining.

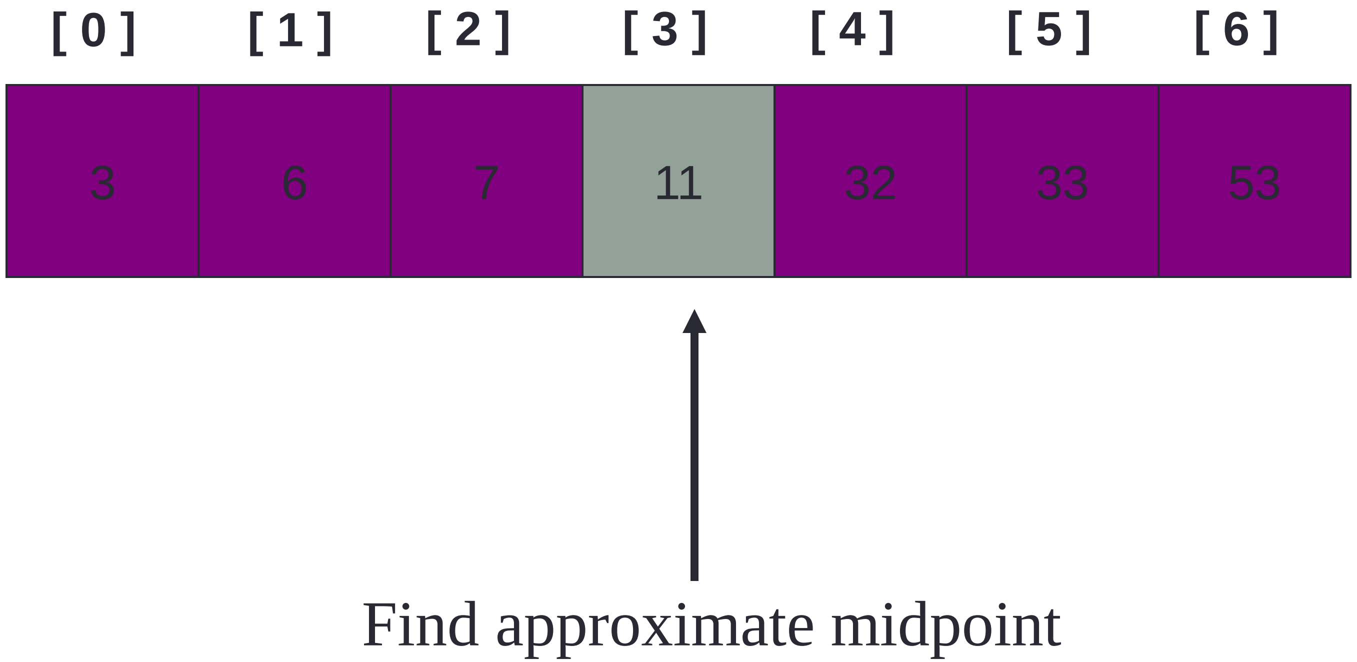
Binary Search

Example: sorted array of integer keys. Target=7.

[0]	[1]	[2]	[3]	[4]	[5]	[6]
3	6	7	11	32	33	53

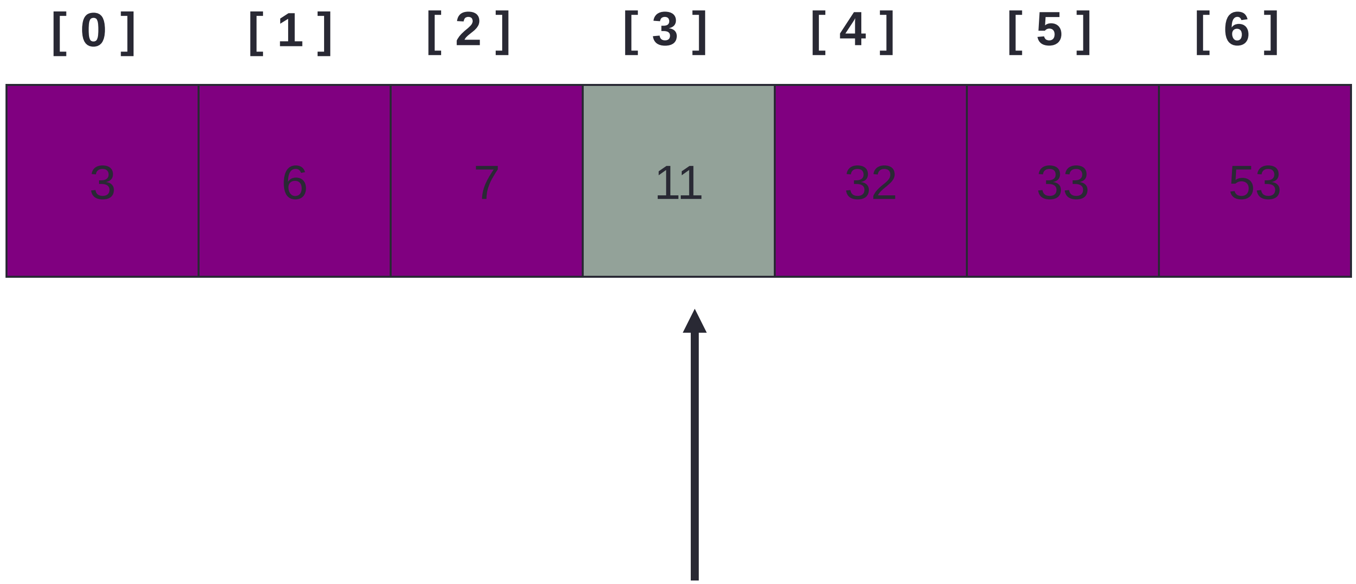
Binary Search

Example: sorted array of integer keys. Target=7.



Binary Search

Example: sorted array of integer keys. Target=7.



Is 7 = midpoint key? NO.

Binary Search

Example: sorted array of integer keys. Target=7.

[0]	[1]	[2]	[3]	[4]	[5]	[6]
3	6	7	11	32	33	53

Is 7 < midpoint key? YES.

Binary Search

Example: sorted array of integer keys. Target=7.

[0]	[1]	[2]	[3]	[4]	[5]	[6]
3	6	7	11	32	33	53

Search for the target in the area before midpoint.

Binary Search

Example: sorted array of integer keys. Target=7.

[0]	[1]	[2]	[3]	[4]	[5]	[6]
3	6	7	11	32	33	53

Find approximate midpoint

Binary Search

Example: sorted array of integer keys. Target=7.

[0]	[1]	[2]	[3]	[4]	[5]	[6]
3	6	7	11	32	33	53

Target = key of midpoint? NO.

Binary Search

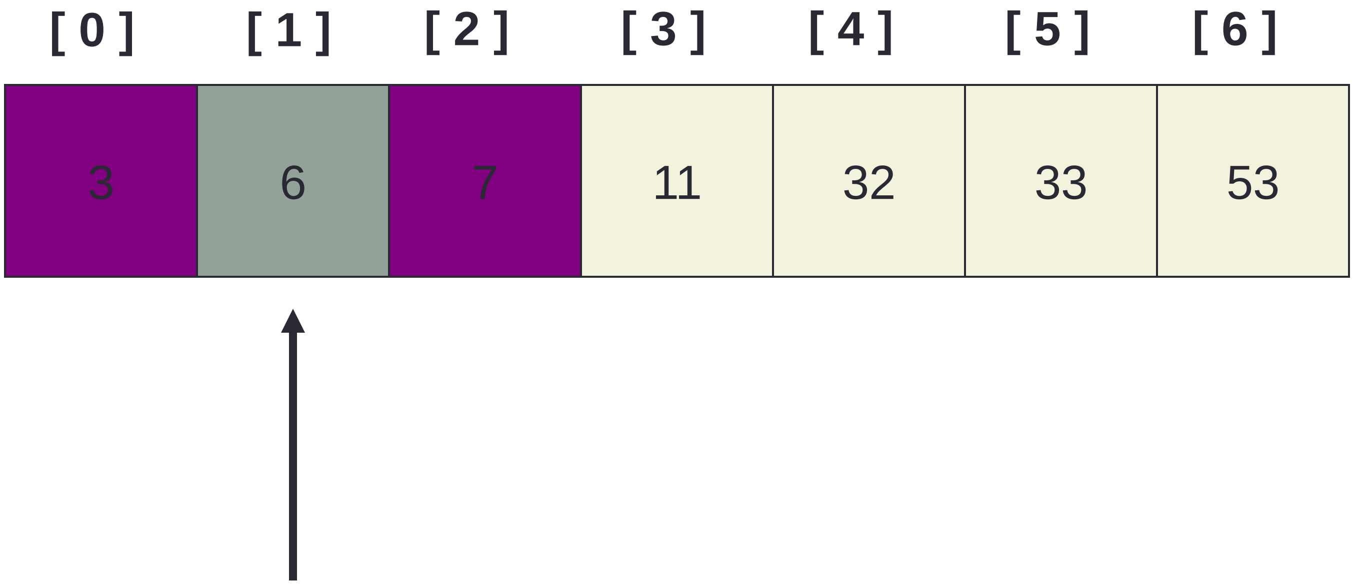
Example: sorted array of integer keys. Target=7.

[0]	[1]	[2]	[3]	[4]	[5]	[6]
3	6	7	11	32	33	53

Target < key of midpoint? NO.

Binary Search

Example: sorted array of integer keys. Target=7.



Target > key of midpoint? YES.

Binary Search

Example: sorted array of integers keys. Target=7.

[0]	[1]	[2]	[3]	[4]	[5]	[6]
3	6	7	11	32	33	53

Search for the target in the area after midpoint.

Binary Search

Example: sorted array of integer keys. Target=7.

[0]	[1]	[2]	[3]	[4]	[5]	[6]
3	6	7	11	32	33	53

Find approximate midpoint.
Is target = midpoint key? YES.

Binary Search

- Element at middle element requires minimum comparison

100	110	125	150	170	190	200	210

- [0] [1] [2] [3] [4] [5] [6] [7]
- Target =210

Low	high	middle	
0	7	3	150
4	7	5	190
6	7	6	200
7	7	7	210 <-Target

Binary Search

- Element at middle element requires minimum comparison

100	110	125	150	170	190	200	210

- [0] [1] [2] [3] [4] [5] [6] [7]
- Target =205

Low	high	middle	
0	7	3	150
4	7	5	190
6	7	6	200
7	7	7	210 <-Target

Binary Search

- Successful Search
  - Best case: $\Theta(1)$
  - Average and worst case = $\Theta(\log_2 n)$
- Unsuccessful Search
  - $\Theta(\log_2 n)$

  - *If n (no of elements) is in the range [*$2^{k-1}, 2^k$*) then binarysearch makes at most K element comparisons for a successfully search and either k-1 or k comparisons for unsuccessfully search.(k=log2n)*

Binary Search

Recurrence Relation for worst and average:

$$T(N) = \begin{cases} \Theta(1) & N == 1 \\ T\left(\frac{N}{2}\right) + 1 & otherwise \end{cases}$$

- Solve this:
- Ans: $\Theta(log_2 N)$

Binary Search

It means !!!

Let's say Tycho-2 star catalog contains information about the brightest ***2,539,913*** *stars in our galaxy. Suppose that you want to search the catalog for a particular star, based on the star's name.*

If the program examined every star in the star catalog in order starting with the first (an algorithm called ***linear search****), the computer might have to examine all* **2,539,913** *stars to find the star you were looking for, in the worst case.*

If the catalog were sorted alphabetically by star names, ***binary search*** *would not have to examine more than* ***22 stars****, even in the worst case.*