UHS C{}MPUTER SCIENCE

New Execs Version.

BINARY SEARCH YAY :)

Find the index of the number 8

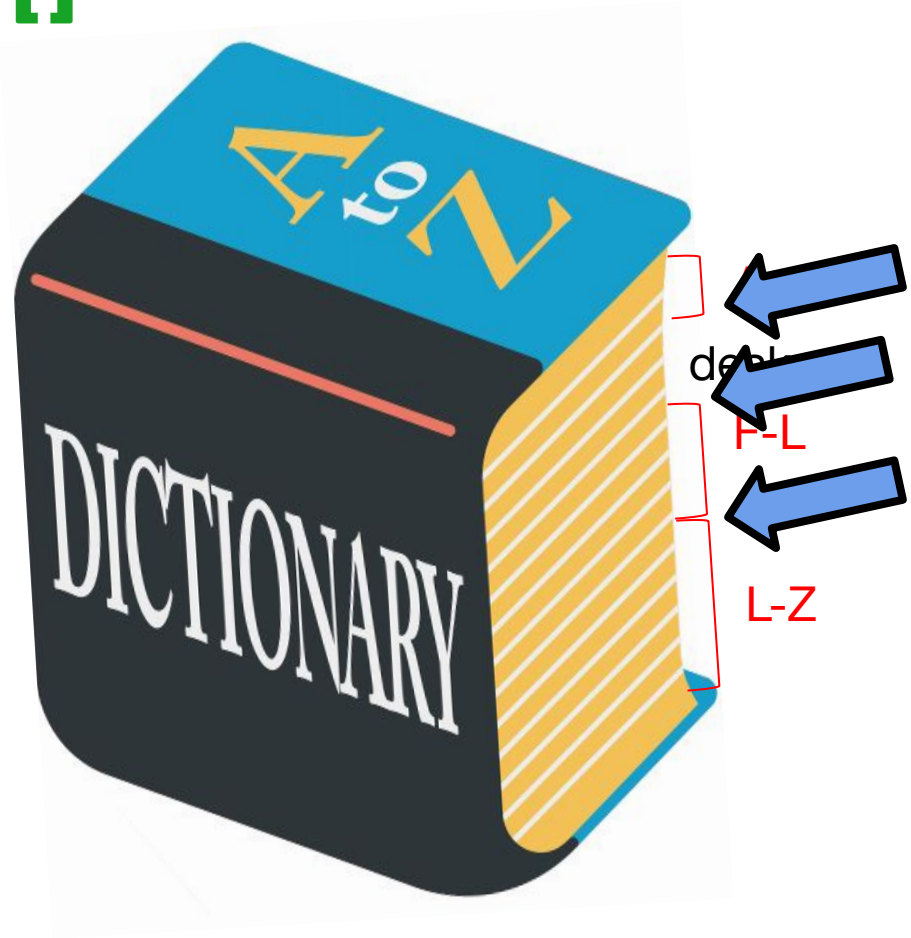
Index:	0	1	2	3	4	5	6	7
Element:	1	2	5	6	6	8	9	9

O(N) <- N operations (where N is size of array)

The Naive Way

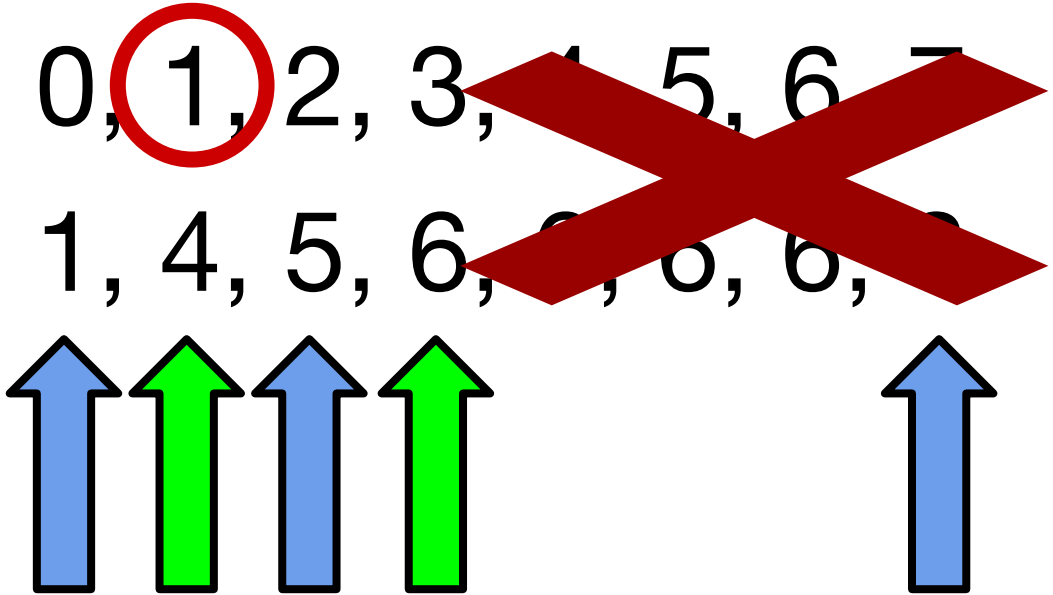
There's got to be a better method!!!

BINARY SEARCH
Joins the Battle!



Find “desk”

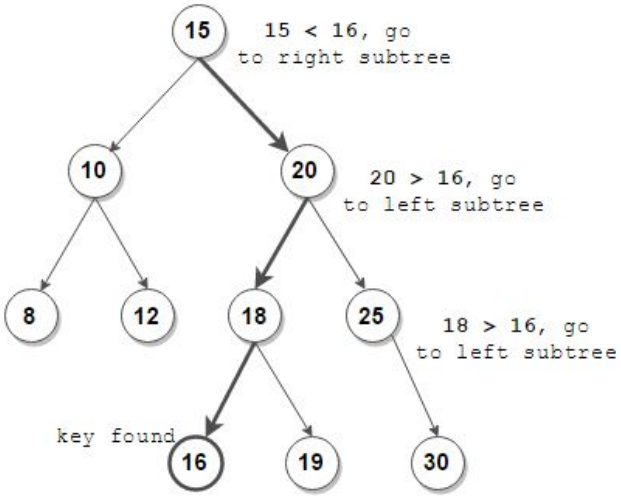
Binary Search for 4



O(log_2N) <- log2(N) operations

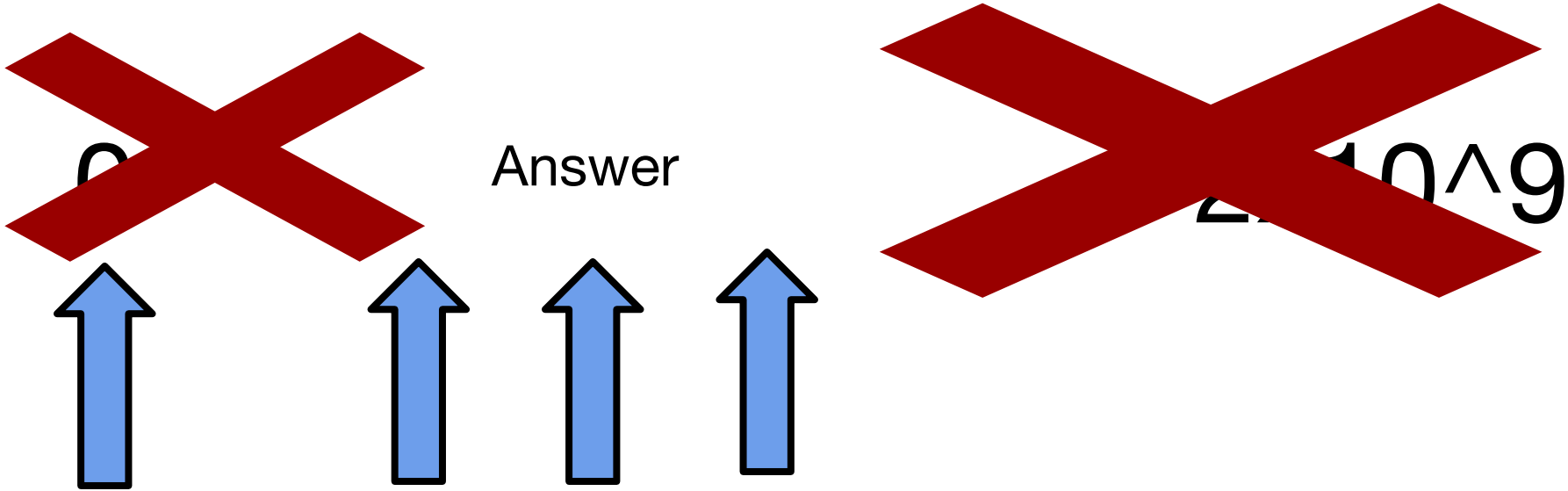
Binary Search Key Points

- Array must be sorted
- Starts from the centre of the array
- Size reduced by half until target found



Search (root, 16)

Why Use Binary Search?


Answer
10^9

Binary Search Functions

C++
Lower_bound <-Binary search for first element >= x
Upper_bound <- Binary search for first element > x

Python
Bisect_left <-Binary search for first element >= x
Bisect_right <- Binary search for first element > x

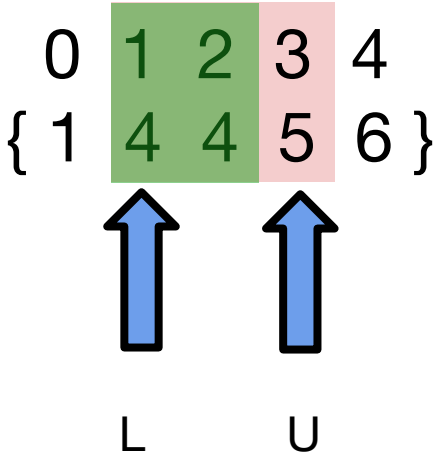
Implementation (C++)

```
#include <bits/stdc++.h>

using namespace std;

Int main(){
        int arr[5] = {1, 4, 4, 5, 6};
        //find index of first element >= 4
        int foursL = lower_bound(arr, arr + 5, 4) - arr;
        //find index of first element > 4
        int foursU = upper_bound(arr, arr+5, 4)-arr;
        //find num of elements = 4
        int ans = foursU-foursL;
}
```

Note for vectors:
lower_bound(v.begin(), v.end(), 4) - v.begin();



Implementation (Python)

```
From bisect import bisect_left, bisect_right
#     0  1  2  3  4  5   6
A = [1, 3, 5, 5, 5, 8, 10]
//find index to first element >= 5
print(bisect_left(A, 5))
//find index to first element > 5
print(bisect_right(A, 5))
```

Base Binary Search Code

```
import java.util.Arrays;


int [] arr = new arr[]{1, 2, 3, 4};
System.out.println(Arrays.binarySearch(arr, 3));
```

Implementation (Java)

```
import Arrays.binarySearch;
//indices                0  1  2  3  4  5  6  7
int [] a = new int[] {1, 3, 4, 6, 6, 6, 8, 9};
//find index to first element >= 6
int index = Math.abs(Arrays.binarySearch(a, 6));
//find number of elements = 6
int indexR = Math.abs(Arrays.binarySearch(a, 7));
int indexL = Math.abs(Arrays.binarySearch(a, 6));
int ans = indexR-indexL;
```

Note: If using ArrayList use Collections.binarySearch();

Some Problems With Java

Problem 1: Does not work as expected when binary searching for a value that has duplicates in the array.
Problem 2: If element does not exist in the array, java returns a negative number. This number is the negative index where the number should be placed -1.

Index:	0	1	2	3	4
Example:	{1,	3,	3,	5,	6}

Binary searching for 4 will return -4.

Practice In Order of Difficulty

https://dmoj.ca/problem/gfsscc21p4 <- easy
https://dmoj.ca/problem/tss17b
https://dmoj.ca/problem/dmopc16c4p2
https://dmoj.ca/problem/aac2p3
https://dmoj.ca/problem/ccc10s3 <- hard