Search in Array

Outline

- Midterm, Oct 16, in class
- Search
- Object-oriented Design

Review: array algorithms

- Copy
- Sum and Avg
  - decimal representation is not precise
- Min
- Max

Warning: cannot assign one array to another using assignment operator =

```
int squares[5] = { 0, 1, 4, 9, 16 };
int lucky_numbers[5];
squares = lucky_numbers;
```

Cannot assign arrays! Compiler will report a syntax error.

Linear Search

- Given an unsorted array of integers and a target, search whether the target is in the array of not.
- If found, return the index of its first occurrence.
- Otherwise return -1.
  - An index must be non-negative, -1 implies that the target is not found.

Linear search in an unordered array

- Linear Search Code
- Key ideas:
- continue to search the array until
  - the target is found. If found, stop at the first match and return the current index. Or
  - All the elements of the array are searched and the target is not found. Return -1.

Linear Search Function: fill in code

```
int linearSearch(int arr[], int size, int target)
{
    for (int i = 0; i < size; i++ )
        if ( arr[i] == target )
            return i;


    return -1;
}
```

Linear Search Function: complete code

```
int linearSearch(int arr[], int size, int target)
{
    for (int i = 0; i < size; i++)
        if (arr[i] == target)
            return i;


    return -1;
}
```

What is wrong with the following code

```
int linearSearch(int arr[], int size, int target)
{
    for (int i = 0; i < size; i++)
        if (arr[i] == target)
            return i;
        else return -1;
}
```

Binary Search in a Sorted Array

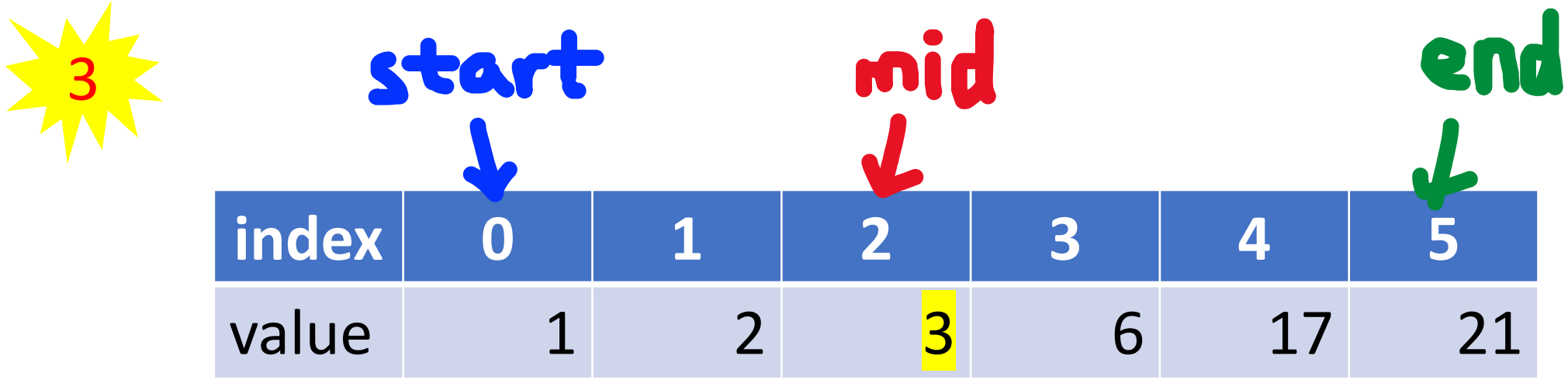
- Given a **non-descending sorted** integer array and a target, find out whether the target is in the array or not.
- If found, return the index of one occurrence.
- Otherwise, return -1.
- For simplicity of drawing a binary search tree later on, we assume that **no duplicate element** in this sorted array.

Key idea of binary search

- Compare the mid with the target,
  - if found, return the index
  - If the target is smaller than the element at mid, search on the smaller half (how?)
  - Otherwise, search on the bigger half (how?)

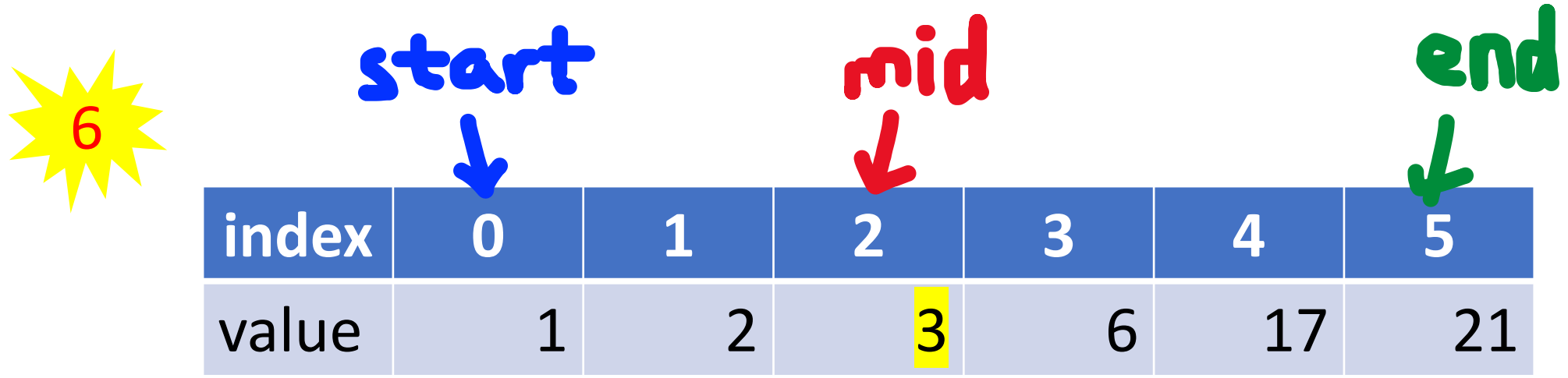
Binary Search in a Sorted Array Example

- Suppose target is 3.



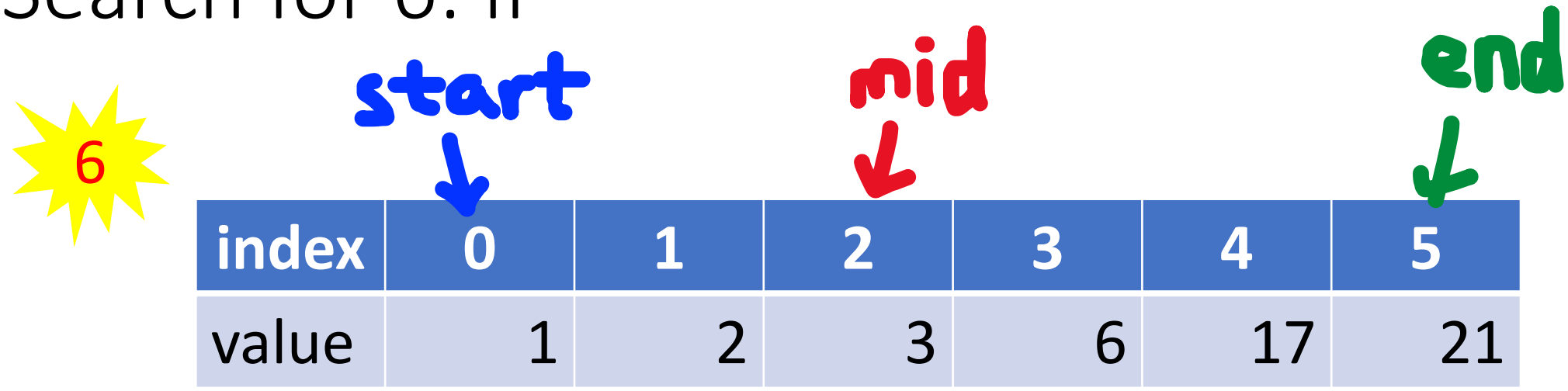
- Step 1: Initialize start- and end- index.
- Step 2: Calculate mid index.
- Step 3: Compare the element at mid index with the target.
- Step 3a: if the element at mid index equals to target, then …

Search for 6



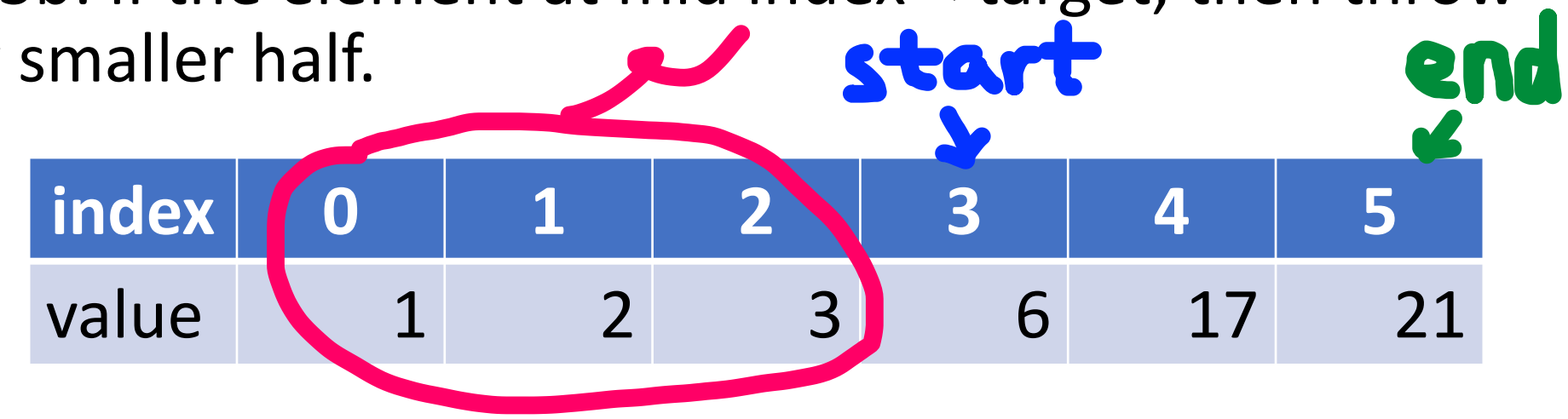
- Step 1: Initialize start- and end- index.
- Step 2: Calculate mid index.
- Step 3: Compare the element at mid index with the target.
- Step 3b: if the element at mid index < target, then …

Search for 6: II



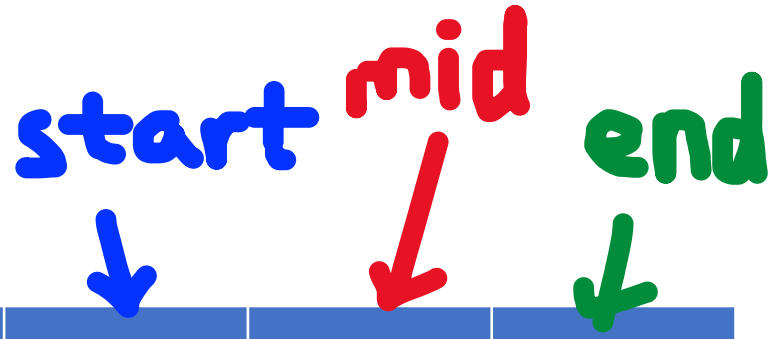
index	0	1	2	3	4	5
value	1	2	3	6	17	21

- Step 3b: if the element at mid index < target, then throw away smaller half.



index	0	1	2	3	4	5
value	1	2	3	6	17	21

Search for 6: III



index	0	1	2	3	4	5
value	1	2	3	6	17	21

- Step 2’: Calculate mid index.
- Step 3’: Compare the element at mid index with the target.
- Step 3’b: if the element at mid index > target, then …

Search for 6: IV

start mid end

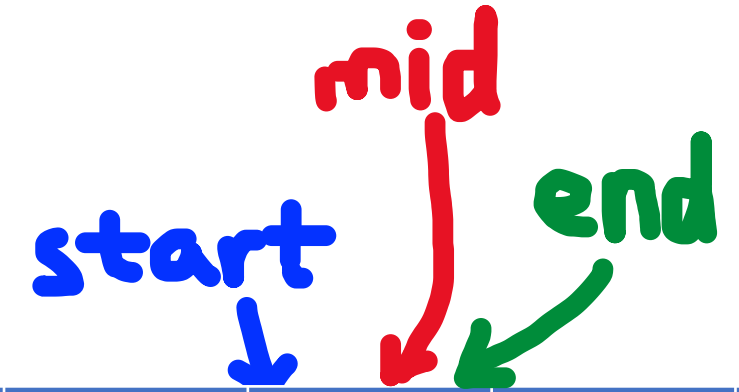
index	0	1	2	3	4	5
value	1	2	3	6	17	21

- Step 3'b: if the element at mid index > target, then throw away bigger half.

start end

index	0	1	2	3	4	5
value	1	2	3	6	17	21

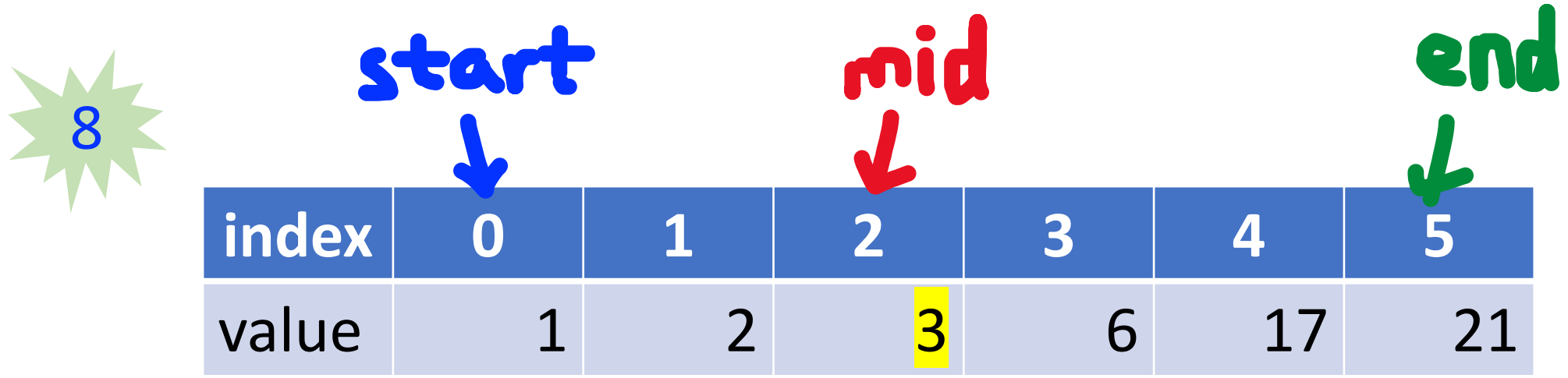
Search for 6: V



index	0	1	2	3	4	5
value	1	2	3	6	17	21

- Step 2’’: Calculate mid index.
- Step 3’’: Compare the element at mid index with the target.
- Step 3’’a: if the element at mid index == target, then …

Search for 8



- Step 1: Initialize start- and end- index.
- Step 2: Calculate mid index.
- Step 3: Compare the element at mid index with the target.
- Step 3b: if the element at mid index < target, then …

Search for 8: II

8

start

mid

end

index	0	1	2	3	4	5
value	1	2	3	6	17	21

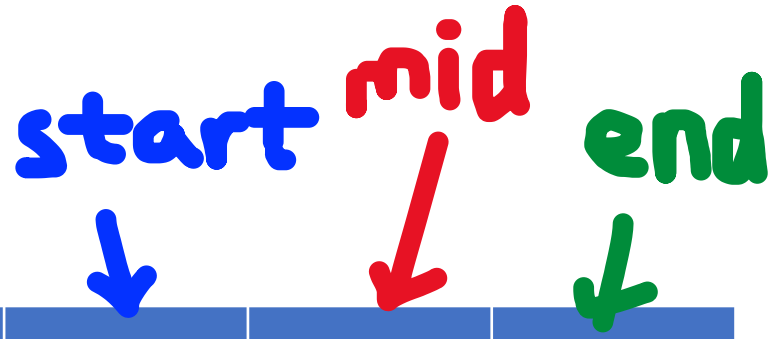
- Step 3b: if the element at mid index < target, then throw away smaller half.

start

end

index	0	1	2	3	4	5
value	1	2	3	6	17	21

Search for 8: III



index	0	1	2	3	4	5
value	1	2	3	6	17	21

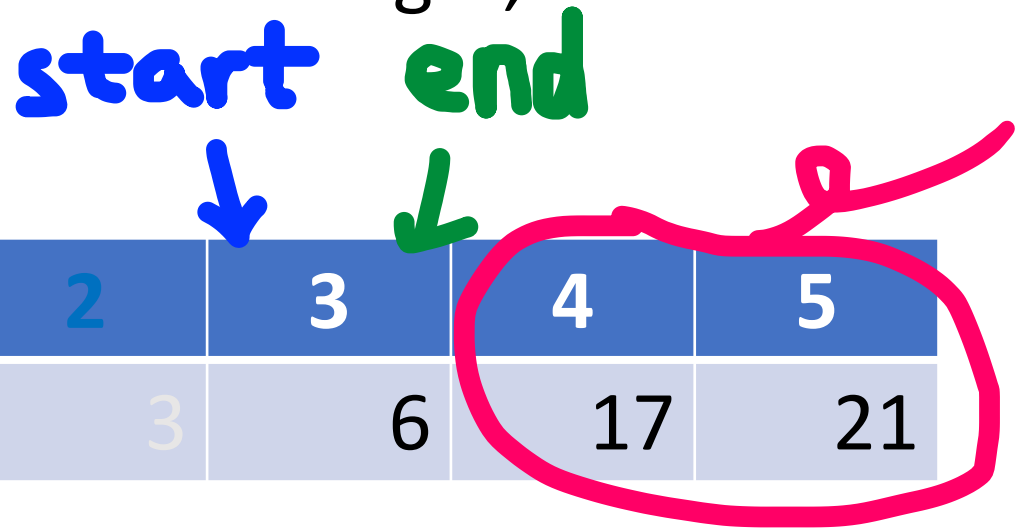
- Step 2’: Calculate mid index.
- Step 3’: Compare the element at mid index with the target.
- Step 3’b: if the element at mid index > target, then …

Search for 8: IV

8

index	0	1	2	3	4	5
value	1	2	3	6	17	21

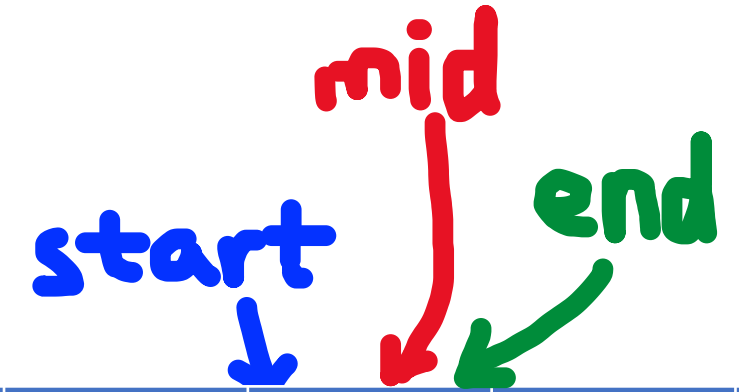
- Step 3'b: if the element at mid index > target, then throw away bigger half.



index	0	1	2	3	4	5
value	1	2	3	6	17	21

Search for 8: V

8

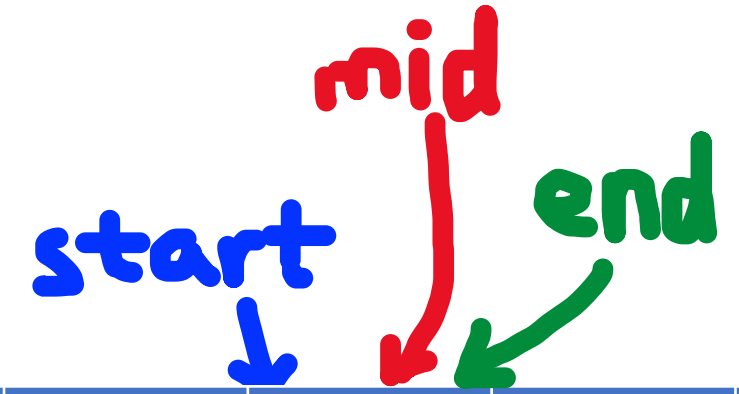


index	0	1	2	3	4	5
value	1	2	3	6	17	21

- Step 2’’: Calculate mid index.
- Step 3’’: Compare the element at mid index with the target.
- Step 3’’a: if the element at mid index < target, then …

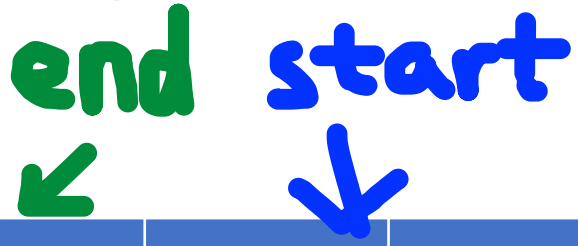
Search for 8: VI

8



index	0	1	2	3	4	5
value	1	2	3	6	17	21

- Step 3’’’a: if the element at mid index < target, then …



index	0	1	2	3	4	5
value	1	2	3	6	17	21

Pseudocode for binary search: II

```
int binarySearch( int* arr, int size, int target ) {
    initialize start and end index
    declare mid
    //to be continued…
```

Pseudocode for binary search

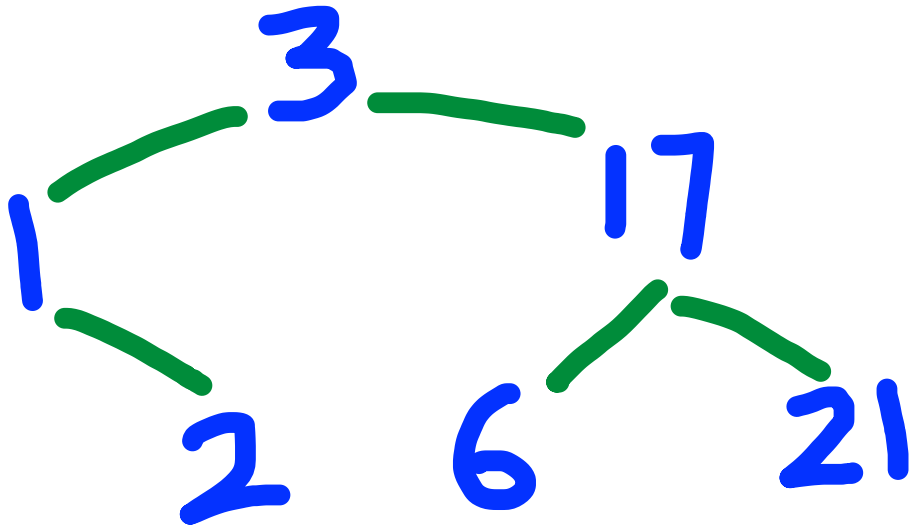
```
while (…) {
        calculate mid indexed, mid of start and end index
        if (arr[mid] == target)
           …
        else if (arr[mid] > target)
                 …
        else …
   }
   ….
} //end of binarySearch function
```

Binary Search in a Sorted Array

- Binary search an element in a sorted array
- Test your submission in leetcode: binary search, choose C language
- Key ideas:
- Compare the mid with the target, if the target matches the element at mid index, return mid, otherwise, concentrate on only half of the original array.
- That is, after a unsuccessful match, the array is reduced by half.

Binary search

- Suppose there are 8 elements in the array, then after each comparison, the array size is reduced to 4, 2, 1.
- In a binary search tree,
  - node in left branch < the root
  - node in right branch > the root
- Number of comparisons to search
  - 3
  - 6
  - 8



linear search vs. binary search

- Suppose an array has n elements
- Linear search works for an unsorted array.
- In the worst case of linear search, need to compare with each elements and not found.
- Binary search needs a sorted array.
- In the worst case of binary search, need $\log_2 n$ comparisons.

Compare n with $\log_2$ n

- Image is from https://dev.to/christinamcmahon/runtime-analysis-big-o-notation-906