Binary Search

zyBook Chap 7.12

Binary Search

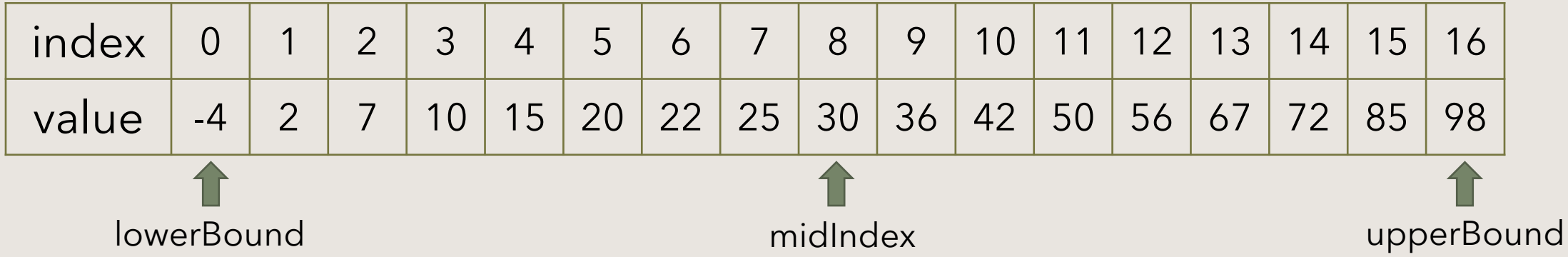
- Locates a **target value** in a **sorted** array by successively **eliminating half** of the array from consideration.

Binary Search (example – search for 42)

index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
value	-4	2	7	10	15	20	22	25	30	36	42	50	56	67	72	85	98

Binary Search (example – search for 42)

- Step 1: Find the indices of 1) lower bound, 2) upper bound, 3) middle (**(lower + upper) / 2**)



Binary Search (example – search for 42)

- Step 1: Find the indices of 1) lower bound, 2) upper bound, 3) middle (**(lower + upper) / 2**)
- Step 2: Compare the target value with the element at the mid index
  - target < arr[midIndex], eliminate the subarray on the right-hand side (including the midIndex)
  - target > arr[midIndex], eliminate the subarray on the left-hand side (including the midIndex)
  - target == arr[midIndex], found!

index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
value	-4	2	7	10	15	20	22	25	30	36	42	50	56	67	72	85	98

Binary Search (example – search for 42)

- Step 1: Find the indices of 1) lower bound, 2) upper bound, 3) middle (**(lower + upper) / 2**)
- Step 2: Compare the target value with the element at the mid index
  - target < arr[midIndex], eliminate the subarray on the right-hand side (including the midIndex)
  - target > arr[midIndex], eliminate the subarray on the left-hand side (including the midIndex)
  - target == arr[midIndex], found!

index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
value	-4	2	7	10	15	20	22	25	30	36	42	50	56	67	72	85	98

lowerBound midIndex upperBound

Binary Search (example – search for 42)

- Step 1: Find the indices of 1) lower bound, 2) upper bound, 3) middle (**(lower + upper) / 2**)
- Step 2: Compare the target value with the element at the mid index
  - target < arr[midIndex], eliminate the subarray on the right-hand side (including the midIndex)
  - **target > arr[midIndex], eliminate the subarray on the left-hand side (including the midIndex)**
  - target == arr[midIndex], found!

index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
value	-4	2	7	10	15	20	22	25	30	36	42	50	56	67	72	85	98

lowerBound midIndex upperBound

Binary Search (example – search for 42)

- Step 1: Find the indices of 1) lower bound, 2) upper bound, 3) middle (**(lower + upper) / 2**)
- Step 2: Compare the target value with the element at the mid index
  - target < arr[midIndex], eliminate the subarray on the right-hand side (including the midIndex)
  - **target > arr[midIndex], eliminate the subarray on the left-hand side (including the midIndex)**
  - target == arr[midIndex], found!

Eliminate half of the array by resetting the lower or upper bound

index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
value	-4	2	7	10	15	20	22	25	30	36	42	50	56	67	72	85	98

lowerBound midIndex upperBound

Binary Search (example – search for 42)

- Step 1: Find the indices of 1) lower bound, 2) upper bound, 3) middle (**(lower + upper) / 2**)
- Step 2: Compare the target value with the element at the mid index
  - target < arr[midIndex], eliminate the subarray on the right-hand side (including the midIndex)
  - **target > arr[midIndex], eliminate the subarray on the left-hand side (including the midIndex)**
  - target == arr[midIndex], found!

Eliminate half of the array by resetting the lower or upper bound

index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
value	-4	2	7	10	15	20	22	25	30	36	42	50	56	67	72	85	98

midIndex

lowerBound

upperBound

Binary Search (example – search for 42)

- Step 1: Find the indices of 1) lower bound, 2) upper bound, 3) middle (**(lower + upper) / 2**)
- Step 2: Compare the target value with the element at the mid index
  - target < arr[midIndex], eliminate the subarray on the right-hand side (including the midIndex)
  - target > arr[midIndex], eliminate the subarray on the left-hand side (including the midIndex)
  - target == arr[midIndex], found!
- Step 3: Repeat until target is found or the range for consideration becomes empty

index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
value	-4	2	7	10	15	20	22	25	30	36	42	50	56	67	72	85	98

Binary Search (example – search for 42)

- Step 1: Find the indices of 1) lower bound, 2) upper bound, 3) middle (**(lower + upper) / 2**)
- Step 2: Compare the target value with the element at the mid index
  - target < arr[midIndex], eliminate the subarray on the right-hand side (including the midIndex)
  - target > arr[midIndex], eliminate the subarray on the left-hand side (including the midIndex)
  - target == arr[midIndex], found!
- Step 3: Repeat until target is found or the range for consideration becomes empty

index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
value	-4	2	7	10	15	20	22	25	30	36	42	50	56	67	72	85	98

Binary Search (example – search for 42)

- Step 1: Find the indices of 1) lower bound, 2) upper bound, 3) middle (**(lower + upper) / 2**)
- Step 2: Compare the target value with the element at the mid index
  - **target < arr[midIndex], eliminate the subarray on the right-hand side (including the midIndex)**
  - target > arr[midIndex], eliminate the subarray on the left-hand side (including the midIndex)
  - target == arr[midIndex], found!
- Step 3: Repeat until target is found or the range for consideration becomes empty

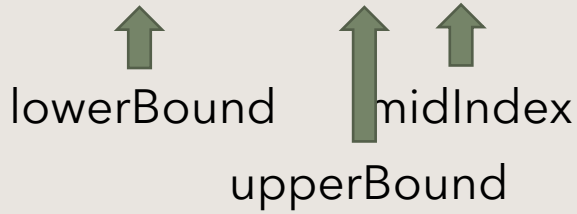
index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
value	-4	2	7	10	15	20	22	25	30	36	42	50	56	67	72	85	98

lowerBound midIndex upperBound

Binary Search (example – search for 42)

- Step 1: Find the indices of 1) lower bound, 2) upper bound, 3) middle (**(lower + upper) / 2**)
- Step 2: Compare the target value with the element at the mid index
  - **target < arr[midIndex], eliminate the subarray on the right-hand side (including the midIndex)**
  - target > arr[midIndex], eliminate the subarray on the left-hand side (including the midIndex)
  - target == arr[midIndex], found!
- Step 3: Repeat until target is found or the range for consideration becomes empty

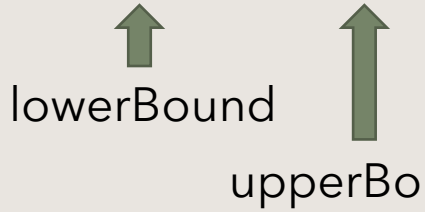
index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
value	-4	2	7	10	15	20	22	25	30	36	42	50	56	67	72	85	98



Binary Search (example – search for 42)

- Step 1: Find the indices of 1) lower bound, 2) upper bound, 3) middle (**(lower + upper) / 2**)
- Step 2: Compare the target value with the element at the mid index
  - target < arr[midIndex], eliminate the subarray on the right-hand side (including the midIndex)
  - target > arr[midIndex], eliminate the subarray on the left-hand side (including the midIndex)
  - target == arr[midIndex], found!
- Step 3: Repeat until target is found or the range for consideration becomes empty

index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
value	-4	2	7	10	15	20	22	25	30	36	42	50	56	67	72	85	98



Binary Search (example – search for 42)

- Step 1: Find the indices of 1) lower bound, 2) upper bound, 3) middle (**(lower + upper) / 2**)
- Step 2: Compare the target value with the element at the mid index
  - target < arr[midIndex], eliminate the subarray on the right-hand side (including the midIndex)
  - target > arr[midIndex], eliminate the subarray on the left-hand side (including the midIndex)
  - target == arr[midIndex], found!
- Step 3: Repeat until target is found or the range for consideration becomes empty

index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
value	-4	2	7	10	15	20	22	25	30	36	42	50	56	67	72	85	98

Binary Search (example – search for 42)

- Step 1: Find the indices of 1) lower bound, 2) upper bound, 3) middle (**(lower + upper) / 2**)
- Step 2: Compare the target value with the element at the mid index
  - target < arr[midIndex], eliminate the subarray on the right-hand side (including the midIndex)
  - target > arr[midIndex], eliminate the subarray on the left-hand side (including the midIndex)
  - **target == arr[midIndex], found!**
- Step 3: Repeat until target is found or the range for consideration becomes empty

index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
value	-4	2	7	10	15	20	22	25	30	36	42	50	56	67	72	85	98

Return the midIndex

```java
// Precondition: Elements in the array are in sorted order
public static int binarySearch(int[] arr, int target) {
    // The boundaries of the array indices
    int lowerBound = 0;
    int upperBound = arr.length - 1;

    // While this is not an empty array ( lowerBound == upperBound suggests a size 1 array )
    while (lowerBound <= upperBound) {
        // Set the midIndex given the lowerBound and upperBound
        int midIndex = (lowerBound + upperBound) / 2;
        // If the element at the midIndex is less than the target, that is,
        // the target is somewhere in the sub-array on the right-hand side
        if (arr[midIndex] < target) {
            // Set the lowerBound to be the first element of the RHS sub-array
            lowerBound = midIndex + 1;
        }
        // If the element at the midIndex is greater than the target, that is,
        // the target is somewhere in the sub-array on the left-hand side
        else if (arr[midIndex] > target) {
            // Set the upperBound to be the last element of the LHS sub-array
            upperBound = midIndex - 1;
        }
        // If the element at the midIndex equals the target, that is,
        // the target is found!
        else {
            return midIndex;
        }
    }

    return -1; // If the target is not found
}
```

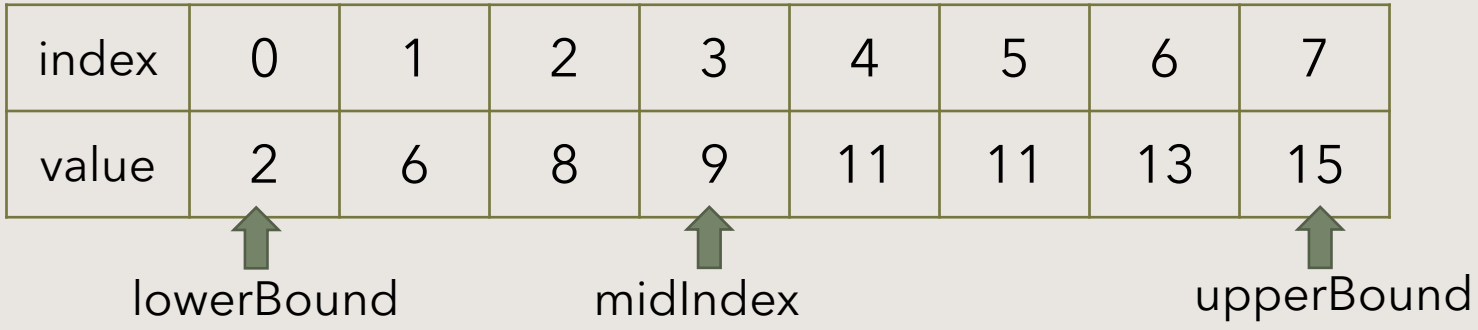
Binary Search Implementation

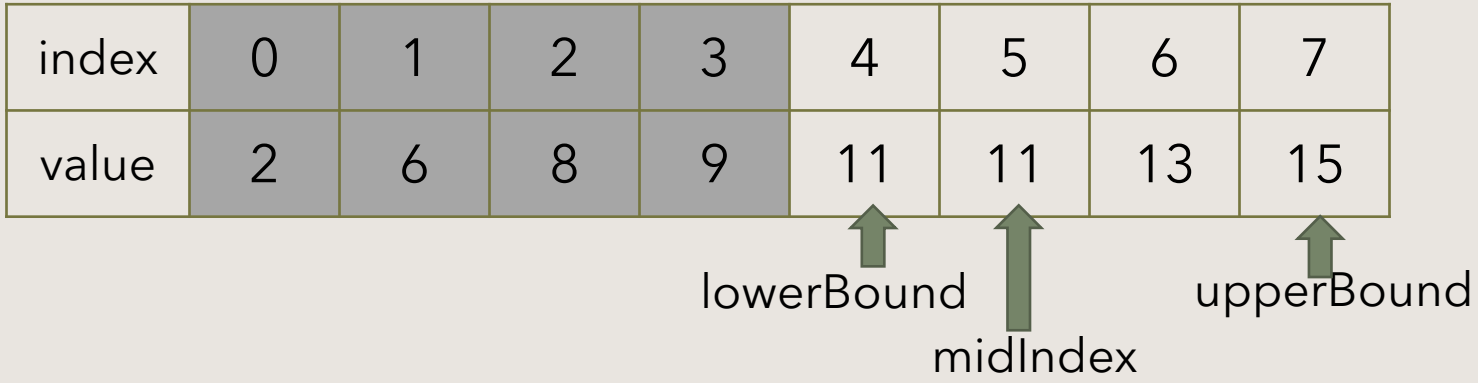
Binary Search Implementation

```java
import java.util.Scanner;
import java.util.Arrays;

public class BinarySearch {
    public static void main(String[] args) {

        int[] arr = {2, 6, 8, 9, 11, 11, 13, 15};

        Scanner input = new Scanner(System.in);
        System.out.print("Enter an integer: ");

        int target = input.nextInt();

        int foundAt = binarySearch(arr, target);

        if (foundAt != -1) {
            System.out.println("Number " + target + " is found at index " + foundAt
                               + " in the array " + Arrays.toString(arr) + ".");
        } else {
            System.out.println("Number " + target + " cannot be found in the array "
                               + Arrays.toString(arr) + ".");
        }
    }

    public static int binarySearch(int[] arr, int target) {
        // See previous slide...
    }
}
```

```
$ javac BinarySearch.java
$ java BinarySearch
Enter an integer: 12
Number 12 cannot be found in the array [2, 6, 8, 9, 11, 11, 13, 15].
$ java BinarySearch
Enter an integer: 11
Number 11 is found at index 5 in the array [2, 6, 8, 9, 11, 11, 13, 15].
```

Q: Determine if the number 11 exists in the following array with Binary Search. If yes, which index will this algorithm return? Visualize each iteration.





Found at index 5