



DATA STRUCTURES & ALGORITHMS

CS106.3

Coursework 2023/2024

Name	J M S V JAYAWEERA
Index	29008
Degree	BSc (Hons) Software Engineering

Coursework

Use recursive approach to solve the Binary Search:

1. Write a proper pseudocode
2. Convert in to a program
3. Test the program with [2, 8, 9, 11, 15, 45, 58, 78, 99] array locating 2.

----- [Answer Below] -----

1. The Pseudocode

```
function binarySearch(arr, low, high, target):  
    if low <= high:  
        mid = (low + high) / 2  
  
        if arr[mid] == target:  
            return mid  
        else if arr[mid] < target:  
            return binarySearch(arr, mid + 1, high, target)  
        else:  
            return binarySearch(arr, low, mid - 1, target)  
    else:  
        return -1
```

2. I have written the program in both Java and C to understand and practice the concept.

- Java

```
public class RecursiveBinarySearch {  
  
    public static int binarySearch(int[] arr, int low, int high, int target) {  
        if (low <= high) {  
            int mid = (low + high) / 2;
```

```

        if (arr[mid] == target) {
            return mid;
        } else if (arr[mid] < target) {
            return binarySearch(arr, mid + 1, high, target);
        } else {
            return binarySearch(arr, low, mid - 1, target);
        }
    } else {
        return -1;
    }
}

```

- C

```

#include <stdio.h>
int binarySearch(int arr[], int low, int high, int target) {
    if (low <= high) {
        int mid = (low + high) / 2;

        if (arr[mid] == target) {
            return mid;
        } else if (arr[mid] < target) {
            return binarySearch(arr, mid + 1, high, target);
        } else {
            return binarySearch(arr, low, mid - 1, target);
        }
    } else {
        return -1;
    }
}

```

3. I Have Tried recursive method to Solve Binary Search using the Both Languages C and Java.

- **Java Code**

// This is a code that i wrote to solve Binary search using recursive method

```

public class RecursiveBinarySearch {

    public static int binarySearch(int[] arr, int low, int high, int target) {
        if (low <= high) {
            int mid = (low + high) / 2;

```

```

        if (arr[mid] == target) {
            return mid;
        } else if (arr[mid] < target) {
            return binarySearch(arr, mid + 1, high, target);
        } else {
            return binarySearch(arr, low, mid - 1, target);
        }
    } else {
        return -1;
    }
}

public static void main(String[] args) {
    int[] arr = {2, 8, 9, 11, 15, 45, 58, 78, 99};
    int target = 2;
    int result = binarySearch(arr, 0, arr.length - 1, target);

    if (result != -1) {
        System.out.println("Element " + target + " is found at index " + result +
            ".");
    } else {
        System.out.println("Element " + target + " is not present in the
            array.");
    }
}
}

```

- **C Code**

// This is a code that i wrote to solve Binary search using recursive method
#include <stdio.h>

```

int binarySearch(int arr[], int low, int high, int target) {
    if (low <= high) {
        int mid = (low + high) / 2;

        if (arr[mid] == target) {
            return mid;
        } else if (arr[mid] < target) {
            return binarySearch(arr, mid + 1, high, target);
        } else {
            return binarySearch(arr, low, mid - 1, target);
        }
    }
}

```

```

    } else {
        return -1;
    }
}

int main() {
    int arr[] = {2, 8, 9, 11, 15, 45, 58, 78, 99};
    int n = sizeof(arr) / sizeof(arr[0]);
    int target = 2;
    int result = binarySearch(arr, 0, n - 1, target);

    if (result != -1) {
        printf("Element %d is found at index %d.\n", target, result);
    } else {
        printf("Element %d is not present in the array.\n", target);
    }

    return 0;
}

```

- The Output

The output using C

The screenshot shows a code editor with a file named 'main.c'. The code implements a binary search function and a main function. The output window on the right shows the result of running the program.

```

main.c
1 #include <stdio.h>
2
3 int binarySearch(int arr[], int low, int high, int target) {
4     if (low <= high) {
5         int mid = (low + high) / 2;
6
7         if (arr[mid] == target) {
8             return mid;
9         } else if (arr[mid] < target) {
10            return binarySearch(arr, mid + 1, high, target);
11        } else {
12            return binarySearch(arr, low, mid - 1, target);
13        }
14    } else {
15        return -1;
16    }
17 }
18
19 int main() {
20     int arr[] = {2, 8, 9, 11, 15, 45, 58, 78, 99};
21     int n = sizeof(arr) / sizeof(arr[0]);
22     int target = 2;
23     int result = binarySearch(arr, 0, n - 1, target);
24
25     if (result != -1) {
26         printf("Element %d is found at index %d.\n", target, result);
27     } else {
28         printf("Element %d is not present in the array.\n", target);
29     }
30
31     return 0;

```

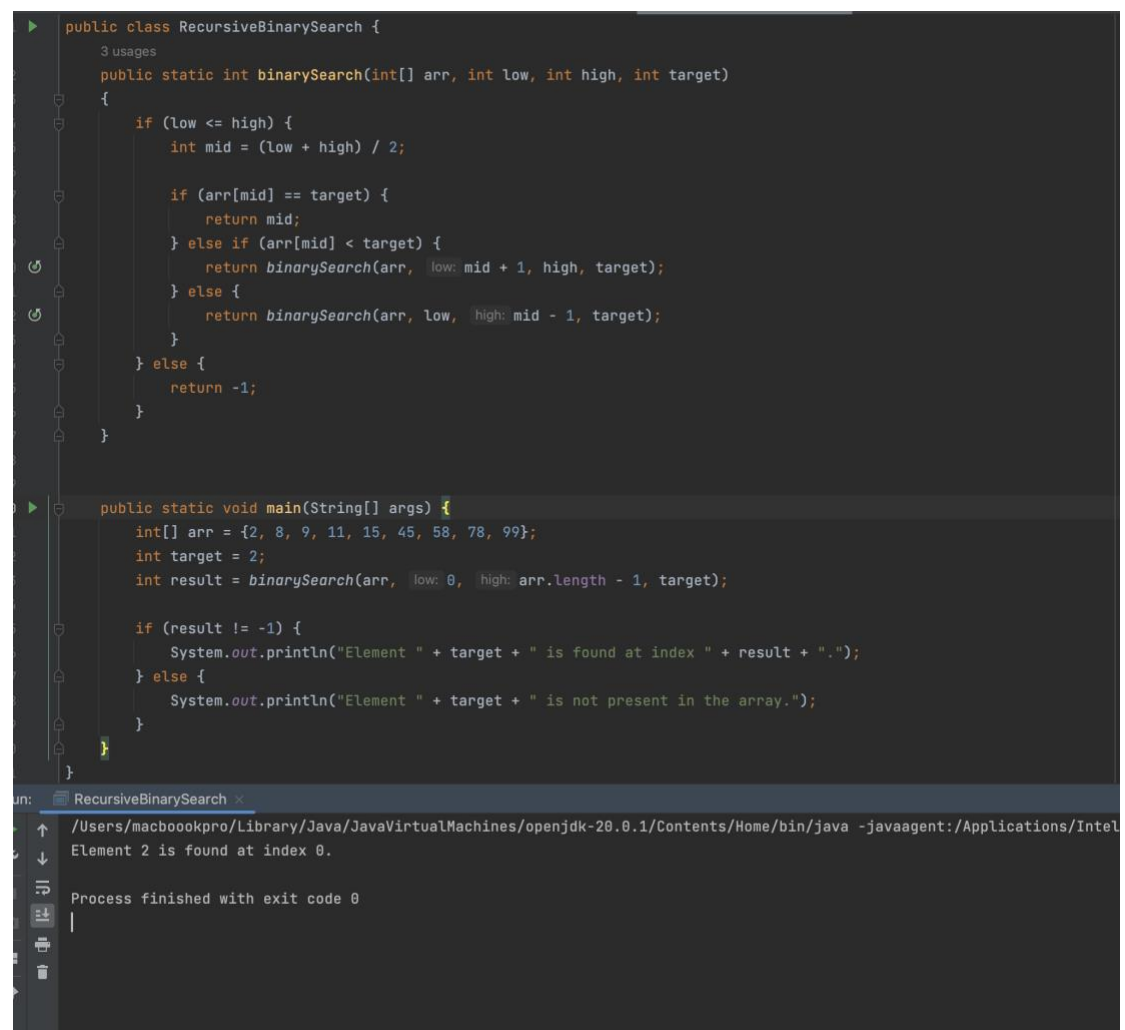
Output

```

/tmp/61rGL59oXL.o
Element 2 is found at index 0.

```

The Output Using Java



```
public class RecursiveBinarySearch {
    3 usages
    public static int binarySearch(int[] arr, int low, int high, int target)
    {
        if (low <= high) {
            int mid = (low + high) / 2;

            if (arr[mid] == target) {
                return mid;
            } else if (arr[mid] < target) {
                return binarySearch(arr, low: mid + 1, high, target);
            } else {
                return binarySearch(arr, low, high: mid - 1, target);
            }
        } else {
            return -1;
        }
    }
}

public static void main(String[] args) {
    int[] arr = {2, 8, 9, 11, 15, 45, 58, 78, 99};
    int target = 2;
    int result = binarySearch(arr, low: 0, high: arr.length - 1, target);

    if (result != -1) {
        System.out.println("Element " + target + " is found at index " + result + ".");
    } else {
        System.out.println("Element " + target + " is not present in the array.");
    }
}
}
```

Run: RecursiveBinarySearch x

/Users/macbookpro/Library/Java/JavaVirtualMachines/openjdk-20.0.1/Contents/Home/bin/java -javaagent:/Applications/Intel

Element 2 is found at index 0.

Process finished with exit code 0