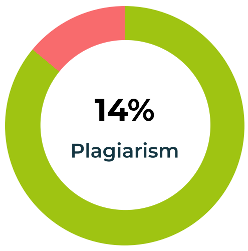
Plagiarism Detector .net

Plagiarism Report



🟩	Unique	86%
🟥	Exact Match	14%
🟦	Partial Match	0%

Primary Sources

1 [https://stackoverflow.com/qu...] 14%

... for (int idx = 0; idx < length; idx++) { // do something } }. results in: public class gti { public gti(); Code: 0: aload_0 1: invokespecial #1 // Method�...

Excluded URL (s)

01 None

Content

```
package LAB1.CODE;
import java.util.*;

public class BinarySearchCode {

// Core Binary Search Mechanism
public static int locateValue(int[] dataset, int keyValue) {
int lowerrBoundArray = 0;
int highrBoundArray = dataset.length - 1;

while (lowerrBoundArray <= highrBoundArray) {
int centerIndex = lowerrBoundArray + (highrBoundArray -
lowerrBoundArray) / 2;

if (dataset[centerIndex] == keyValue) {
return centerIndex;
}
if (dataset[centerIndex] < keyValue) {
lowerrBoundArray = centerIndex + 1;
} else {
highrBoundArray = centerIndex - 1;
}
}
return -1;
}

// Calculate time taken for a single search
public static double trackSearchTime(int[] numbers, int searchTerm) {
long initNano = System.nanoTime();
```

```
locateValue(numbers, searchTerm);
long finalNano = System.nanoTime();
return (finalNano - initNano) / 1_000_000.0; // Convert to milliseconds
}

// Generate a sorted random integer array
public static int[] buildSortedRandomArray(int length, int range) {
Random generator = new Random();
int[] randomizedArray = new int[length];
for (int idx = 0; idx < length; idx++) {
randomizedArray[idx] = generator.nextInt(range * 2) - range; // negatives +
positives
}
Arrays.sort(randomizedArray);
return randomizedArray;
}

public static void main(String[] args) {
// Edge-case arrays
int[] nullSizeArray = {};
int[] oneElementArray = {77};
int[] negativesArray = {-22, -11, 0, 11, 22, 33};
int[] repeatedValues = {9, 9, 9, 9, 9};
int[] oddLengthArray = {1, 3, 5, 7, 9};
int[] evenLengthArray = {2, 4, 6, 8, 10, 12};
int[] targetAtStart = {5, 10, 15, 20};
int[] targetAtEnd = {2, 4, 6, 8, 10};
int[] mixedNegPos = {-50, -20, 0, 20, 40, 60};

// Large dataset
int[] masterData = buildSortedRandomArray(1000, 500);

System.out.println("Binary Search Execution Times (ms):\n");

// ===== BEST CASES =====
int[] best1 = masterData;
System.out.println("Best #1: " + trackSearchTime(best1, best1[best1.length /
2]));

int[] best2 = buildSortedRandomArray(500, 250);
System.out.println("Best #2: " + trackSearchTime(best2, best2[best2.length /
2]));

int[] best3 = buildSortedRandomArray(200, 100);
System.out.println("Best #3: " + trackSearchTime(best3, best3[best3.length /
2]));

int[] best4 = buildSortedRandomArray(50, 25);
System.out.println("Best #4: " + trackSearchTime(best4, best4[best4.length /
2]));

int[] best5 = buildSortedRandomArray(10, 5);
System.out.println("Best #5: " + trackSearchTime(best5, best5[best5.length /
2]));

// ===== WORST CASES =====
System.out.println("\n---- WORST CASES ----");
System.out.println("Worst #1: " + trackSearchTime(masterData, 999999));
System.out.println("Worst #2: " +
trackSearchTime(buildSortedRandomArray(500, 250), -888));
System.out.println("Worst #3: " +
trackSearchTime(buildSortedRandomArray(200, 100), 7777));
System.out.println("Worst #4: " +
trackSearchTime(buildSortedRandomArray(50, 25), -4444));
System.out.println("Worst #5: " +
trackSearchTime(buildSortedRandomArray(10, 5), 321));

// ===== AVERAGE CASES =====
System.out.println("\n---- AVERAGE CASES ----");
Random rand = new Random();

int[] avg1 = masterData;
System.out.println("Average #1: " + trackSearchTime(avg1,
avg1[rand.nextInt(avg1.length - 2) + 1]));

int[] avg2 = buildSortedRandomArray(500, 250);
System.out.println("Average #2: " + trackSearchTime(avg2,
avg2[rand.nextInt(avg2.length - 2) + 1]));

int[] avg3 = buildSortedRandomArray(200, 100);
System.out.println("Average #3: " + trackSearchTime(avg3,
avg3[rand.nextInt(avg3.length - 2) + 1]));

int[] avg4 = buildSortedRandomArray(50, 25);
System.out.println("Average #4: " + trackSearchTime(avg4,
avg4[rand.nextInt(avg4.length - 2) + 1]));

int[] avg5 = buildSortedRandomArray(10, 5);
System.out.println("Average #5: " + trackSearchTime(avg5,
avg5[rand.nextInt(avg5.length - 2) + 1]));

// ===== EXTRA EDGE CASES =====
System.out.println("\n---- EDGE CASES ----");
System.out.println("Empty Array: " + trackSearchTime(nullSizeArray, 10));
System.out.println("Single Found: " + trackSearchTime(oneElementArray,
77));
System.out.println("Single Not Found: " +
trackSearchTime(oneElementArray, 88));
System.out.println("Negatives: " + trackSearchTime(negativesArray, -11));
System.out.println("Duplicates: " + trackSearchTime(repeatedValues, 9));
System.out.println("Odd Length: " + trackSearchTime(oddLengthArray, 5));
System.out.println("Even Length: " + trackSearchTime(evenLengthArray, 6));
System.out.println("Target At Start: " + trackSearchTime(targetAtStart, 5));
System.out.println("Target At End: " + trackSearchTime(targetAtEnd, 10));
System.out.println("Mixed Negatives & Positives: " +
trackSearchTime(mixedNegPos, 0));
}
}
```

References