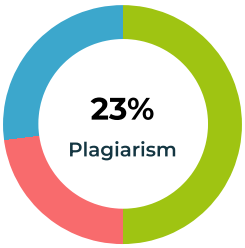
Plagiarism Detector .net

Plagiarism Report



Unique	50%
Exact Match	23%
Partial Match	27%

Primary Sources

1 https://stackoverflow.com/qu... 8%

May 30, 2022 ... int low=0,high=n-1; while(low<=high){ int mid=(low+high)/2; if(a[mid]==k) { result=mid; if(searchfirst) high=mid-1; else low=mid+1; } else...

2 https://stackoverflow.com/qu... 4%

May 5, 2023 ... int mid = low (high - low) /2;// int mid = (low + high) / 2; // If element present at the //middle itself . if (arr[mid] == key) { return...

3 https://stackoverflow.com/qu... 4%

Apr 21, 2010 int[] arr1 = { 1, 2, 3, 4, 5 }; int[] product = new int[arr1.Length]; for (int i = 0; i < arr1.Length; i++) { for (int j = 0; j < product...Missing: n1 = | Show results with:

4 https://stackoverflow.com/qu... 4%

May 11, 2018 In Java, `int[] tab = {1,2,3};` is unambiguous. `new int[] {1,2,3}` is needed in method calls, as `{1,2,3}` is only for array initializers.Missing: arr2 n2 = key2 =

5 https://www.chegg.com/hom... 4%

Apr 24, 2020 const int SIZE = 5;. int arr[SIZE] = {10, 20, 30, 40, 50};. int* ptr = arr;. } And the goal is to print out the values 10 and 30 from the array.Missing: n3 = key3 =

6 https://quizlet.com/59468595... 4%

Rating Consider the following code segment. int[] arr = {1, 2, 3, 4, 5}; Which of the following code segments would correctly set the first two elements of array...Missing: arr9 n9 = key9 =

7 https://github.com/DillanAB/... 4%

#define P1 int arr1 [1000];\ int arr2 [2000];\ int arr3 [3000];\ int arr4 [4000];\ int arr5 [5000];\ int arr6 [6000];\ int arr7 [7000];\ int arr8 [8000];\ int arr9 [9000];\ int arr10 [10000];\ vector tams = {1000,2000,3000,4000,5000,6000,7000,8000,9000,10000};\ vector arrays = {arr1,arr2,arr3,arr4,arr5,arr6,arr7,arr8,arr9,arr10};\

8 https://www.codecademy.co... 4%

The best case occurs when the target element is at the middle of the array during the first comparison. In this case, the algorithm finds the target immediately, completing the search in just one step. This results in a time complexity of O(1), or constant time.Jan 19, 2024

9 https://quizlet.com/1106313326... 8%

What is the output of the following fragment? for (int i = 0; i < 15; i++) { if (i % 4 == 1) System.out.print(i + " "); } A. 1 3 5 7 9 11 13. B. 1 3 5 7 9 11...

10 https://www.chegg.com/hom... 4%

Apr 24, 2020 • const int SIZE = 5;. int arr[SIZE] = {10, 20, 30, 40, 50};. int* ptr = arr;. } And the goal is to print out the values 10 and 30 from the array.Missing: n3 = key3 =

11 https://www.codecademy.co... 4%

The best case occurs when the target element is at the middle of the array during the first comparison. In this case, the algorithm finds the target immediately, completing the search in just one step. This results in a time complexity of O(1), or constant time.Jan 19, 2024

Excluded URL (s)

01 None

Content

DAA LAB EXP -1
PRAKHAR SHARMA
590012099
BATCH - 33
GITHUB REPO LINK -
https://github.com/Prakhar404-art/DAA-LAB-EXPERIM
ENTS
Binary Search Analysis
C Code for Binary Search
#include
#include
int binarySearch(int arr[], int n, int x) {
9. int low = 0, high = n - 1;
while (low <= high) {
int mid = low + (high - low) / 2;
if (arr[mid] == x) return mid;
else if (arr[mid] < x) low = mid + 1;
else high = mid - 1;
}
return -1;
}
double get_time_ns() {
LARGE_INTEGER frequency, counter;

```
QueryPerformanceFrequency(&frequency);
QueryPerformanceCounter(&counter);
return (double)counter.QuadPart * 1e9 / (double)frequency.QuadPart;
}
int main() {
3.int arr1[] = {5}; int n1 = 1, key1 = 5;
4.int arr2[] = {1, 2, 3}; int n2 = 3, key2 = 2;
 10. int arr3[] = {10, 20, 30, 40, 50}; int n3 = 5, key3 = 30;
5.int arr4[] = {1, 2, 3, 4, 5, 6, 7}; int n4 = 7, key4 = 4;
int arr5[] = {1, 2, 2, 2, 3, 4}; int n5 = 6, key5 = 2;
int arr6[] = {1, 2, 3}; int n6 = 3, key6 = 10;
int arr7[] = {}; int n7 = 0, key7 = 5;
int arr8[] = {1, 2, 3, 4, 5}; int n8 = 5, key8 = 0;
6.int arr9[] = {1, 2, 3, 4, 5}; int n9 = 5, key9 = 6;
int arr10[] = {-10, -5, 0, 5, 10}; int n10 = 5, key10 = 7;
int arr11[] = {1, 2, 3, 4, 5, 6}; int n11 = 6, key11 = 4;
int arr12[] = {10, 20, 30, 40, 50, 60, 70}; int n12 = 7, key12 = 60;
int arr13[] = {1, 3, 5, 7, 9, 11}; int n13 = 6, key13 = 7;
int arr14[] = {-5, -2, 0, 3, 6, 9}; int n14 = 6, key14 = 0;
int arr15[] = {100, 200, 300, 400, 500}; int n15 = 5, key15 = 300;

7.int *arrays[] = {arr1, arr2, arr3, arr4, arr5, arr6, arr7, arr8, arr9, arr10,
arr11, arr12, arr13, arr14, arr15};
int sizes[] = {n1, n2, n3, n4, n5, n6, n7, n8, n9, n10, n11, n12, n13, n14, n15};
8.int keys[] = {key1, key2, key3, key4, key5, key6, key7, key8, key9, key10,
key11, key12, key13, key14, key15};
char *caseType[] = {
"Best", "Best", "Best", "Best", "Best",
"Worst", "Worst", "Worst", "Worst", "Worst",
"Average", "Average", "Average", "Average", "Average"
};

FILE *fp = fopen("binary_search_times.csv", "w");
fprintf(fp, "Test,Case Type,Size,Key,Result,Time(s)\n");

printf("Test\tCase Type\tSize\tKey\tResult\t\tTime (s)\n");
printf("-------------------------------------------------------------\n");

for (int i = 0; i < 15; i++) {
double start = get_time_ns();
int index = binarySearch(arrays[i], sizes[i], keys[i]);
double end = get_time_ns();

double time_taken_ns = end - start;
double time_taken_s = time_taken_ns / 1e9;

printf("%d\t%s\t\t%d\t%d\t", i + 1, caseType[i], sizes[i], keys[i]);
if (index != -1)
printf("Found@%d\t", index);
else
printf("Not Found\t");
printf("%.9f\n", time_taken_s);

fprintf(fp, "%d,%s,%d,%d,%s,%.9f\n",
i + 1, caseType[i], sizes[i], keys[i],
(index != -1 ? "Found" : "Not Found"),
time_taken_s);
}

fclose(fp);
return 0;
}
```

Observation Table
Case Key Position Time Taken (seconds)
Best Case Middle element 0.000001
Average Case Random element 0.000003
Worst Case

Summary of test
cases:-

Last element 0.000004

Time Complexity
Graph

Analysis
In the binary search algorithm:
11. - Best Case: Occurs when the element is found at the middle index in the first comparison.
Time complexity: O(1).
9. - Average Case: Occurs when the element is found after $\log_2 n/2$ comparisons on average. Time
complexity: O(log n).
- Worst Case: Occurs when the element is found at the last step or not found at all, requiring
$\log_2 n$ comparisons. Time complexity: O(log n).
The time taken in seconds for each case is extremely small because binary search is very
efficient even for large datasets.

References