CSCI 421 Design and Analysis of Algorithms Spring 2019

Lecture 1 Activity 2

1. Give the order of growth (as a function of N) of the running times of each of the following code fragments:

a

|  |
| --- |
| int sum = 0;  for (int n = N; n > 0; n /= 2)  for (int i = 0; i < n; i++)  sum++; |

Run Time: Simply N

b

|  |
| --- |
| int sum = 0;  for (int i = 1; i < N; i \*= 2)  for(int j = 0; j < i; j++)  sum++; |

Run Time : O(N)

c

|  |
| --- |
| int sum = 0;  for (int i = 1; i < N; i \*= 2)  for (int j = 0; j < N; j++)  sum++; |

Run time : o(N log(N))

1. We have a special array that is comprised of a decreasing sequence of integers followed immediately by an increasing sequence of integers.

Write a program that, given a special array (defined above) of N distinct int values, determines whether a given integer is in the array. Describe and justify how many compares your program uses in the worst case. Run your program against the input array as follows, suppose you would like to check whether “-21” and “18” are in the array. Attach your code and screenshots of your output here.

18 17 14 9 6 5 1 -5 -9 -17 -22 -25 -20 -18 -10 -7 -3 0 2 4 8 10 15 16

import java.util.Arrays;

import java.util.Scanner;

public class search2{

public static int binarySearch(int[] a, int key){

int lo = 0, hi = a.length-1;

while (lo <= hi){

int mid = lo + (hi - lo) / 2;

if (key < a[mid]) hi = mid - 1;

else if (key > a[mid]) lo = mid + 1;

else return mid;

}

return -1;

}

public static void main(String [] args){

int[] searchArray = {18,17,14,9,6,5,1,-5,-9,-17,-22,-25,-20,-18,-10,-7,-3,0,2,4,8,10,15,16};

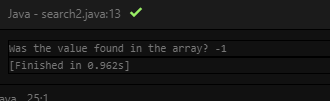
Arrays.sort(searchArray);

System.out.println("Was the value found in the array? " + binarySearch(searchArray, -23));

}

}

When searching for value not in array -23



When searching for value in array (18) it returns the position in the array:

