Project 2

Ganesh Kumar Rajasekar

1 Problem Statement

Finding Max Number in Circular Shifted Array

We are given an array A[1..n] of sorted integers that has been circularly shifted some positions to the right. For example, [35, 42, 5, 15, 27, 29] is a sorted array that has been circularly shifted 2 positions, while [27, 29, 35, 42, 5, 15] has been shifted 4 positions. We can obviously find the largest element in A in O(n) time. Describe an O(log n) algorithm.

2 Theoretical Analysis

The time complexity of this algorithm is O (log n) since in each step, we reduce the search space by half. The worst-case scenario occurs when all elements are distinct, and we perform a binary search on the entire array.

3 Analysis

3.1 Program Listing

Code : https://github.com/GaneshKumarRajasekar/Project-2/tree/main/Project2

```
import java.util.Scanner;
public class DAA {
   //Function to right-rotate an array by one position
   static void rotate_right_one(int[] a, int n) {
      int last = a[n - 1]; //Store the last element of the array in 'last'
      for (int i = n - 2; i >= 0; i--)
         a[i + 1] = a[i]; //Shift elements one position to the right
      a[0] = last; //Place the stored 'last' element at the beginning of the array
   }
   //Function to right-rotate an array by `k` positions
   static void rotate_right(int[] a, int k, int n) {
      for (int i = 0; i < k; i++)
         rotate_right_one(a, n); //Rotate the array one position to the right `k` times
   }
   //Main
   public static void main(String[] args) {
```

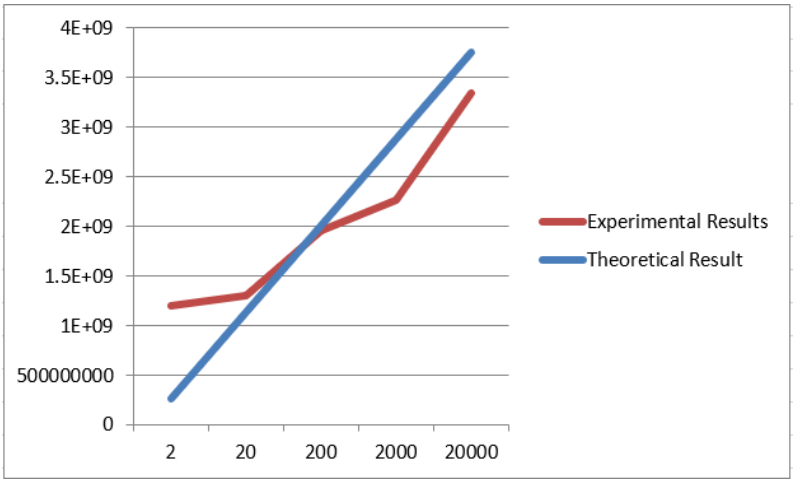
```
        Scanner sc = new Scanner(System.in);
        int k;
        int[] a = { 35, 42, 5, 15, 27, 29 }; //Define an array 'a' with initial values
        System.out.println("Enter the value of k:");

        k = sc.nextInt(); //Read the value of `k` from the user
        int n = a.length; //Get the length of the array `a`
        rotate_right(a, k, n); //Right-rotate the array `a` by `k` positions
        //Print the rotated array
        for (int i = 0; i < n; i++)
            System.out.print(a[i] + " ");
    }
}
```

3.2 Output Numerical Data

n	Experimental Result, in ns	Theoretical Result	Scaling Constant	Adjusted Theoretical Result
2	1195182997	1		263035302
20	1305249602	4.32192809		1136819660
200	1952968143	7.64385619		2010604018
2000	2260310555	10.9657843		2884388376
20000	3339308793	14.2877124		3758172734
	2010604018	7.64385619	263035302	

3.3 Graph



Conclusion: the experimental results and the theoretical result have same tendency so the time complexity result is correct.