Project 2 **Kevin Varghese Kuruvilla(G21925446)**
10/04/24

Github:https://github.com/kevin-uxx/Finding-Max-Number-in-Circular-Shifted-Array

Project 2

Kevin Varghese Kuruvilla

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

We can find the largest element in A in O(log n) time as well. With the help of binary search approach. We start with two pointers low = 0 and high = n-1 and using this we also calculate the middle element by mid = (low +high)/2. Using mid pointer we can check if our maximum element in the array A lies in the left half or right half. If A[mid] > A[high] it means maximum element is in right side of the array, so we change the low to mid + 1 and search right side. If A[mid] <= A[high] it means right side is sorted and maximum element is in the left side. So we change the high pointer to mid and search left side. Now again we calculate mid with new low and high and continue the search until low becomes equal to high. At that point we will find our maximum element. Hence the time complexity will be O(logn) since each step of binary search reduces the size of search by half. The time to compare between mid and high will be constant that is O(1). The recurrence relation can be written as T(n) = T(n/2) + O(1) , by masters theorem we get the solution to be O(logn)

3 Experimental Analysis

3.1 Program Listing

I used Java to code the above program and the experimental and theoretical time for different values of n like 500000, 750000, 1000000, 2000000,4000000

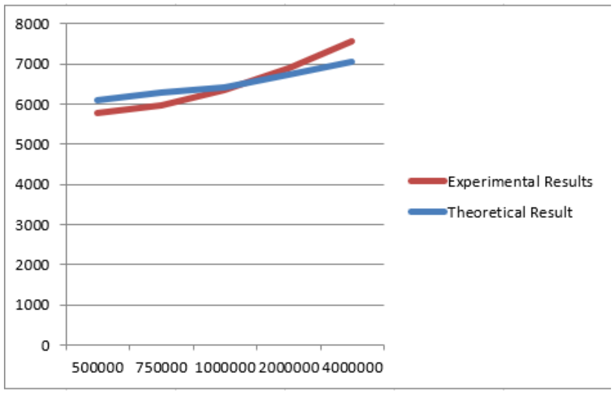
3.2 Data Normalization Notes

I normalized the theoretical values. I derived the scaling constant by dividing the average of experimental values (x) by the average of theoretical values (y) i.e x/y = scaling constant. 6529/20.2474 = 322.4611555. I multiplied each theoretical values with this constant to get the normalized value.

3.3 Output Numerical Data

n	Experimental Result, in ns	Theoretical Result	Scaling Constant	Adjusted Theoretical Result
500000	5789	18.93		6089.4177
750000	5977	19.517		6278.24434
1000000	6337	19.93		6411.09851
2000000	6899	20.93		6732.77932
4000000	7564	21.93		7054.46013
	6513.2	20.2474	321.680808	

3.4 Graph



3.5 Graph Observation

From the above graph we can see that my experimental value and theoretical value have similar growth function and both grows in log n time

4 Conclusions

The above graph demonstrate that for smaller values of n both the values closely align reflecting the expected O(log n) behavior is correct. But as n increases the experimental values slightly diverge due to external factors. Though the overall growth trend remains same with theoretical prediction.