PROJECT - 2

GitHub Link

https://github.com/varadasainikhil/CS6212-Project02

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 have to find the largest number in a circularly shifted array in O(log n) time. Since the array is already sorted the best way to achieve O(log n) is to do Binary Search.

1. Initialize two pointers 'low' and 'high', to find the first and last indices of the array.
2. Check if there is only one element left in the array.
3. Find the middle index and check if the middle index is the largest value.
4. Now check if the middle index is larger than the first or the last index.
5. If the middle is greater than the first index, then call the function with low and mis-1 as the pointers.
6. Otherwise, call the function with the mid+1 and high as the pointers.

This algorithm will run in O (log n) because this is a modified Binary Search algorithm.

3 Experimental Analysis

3.1 Program Listing

The values of n that I have taken are 8, 256,2048,4096,8192,16384,32768 and 65536 because they are all the powers of 2. The time complexity calculated is O(log n).

3.2 Data Normalization Notes

Since the experimental values were in nanoseconds and the theoretical results were constants, to compare the data, we need to normalize data using the scaling constant.

$$Scaling\ Constant = \frac{Average\ of\ Experimental\ Results\ in\ ns}{Average\ of\ Theoretical\ Results}$$

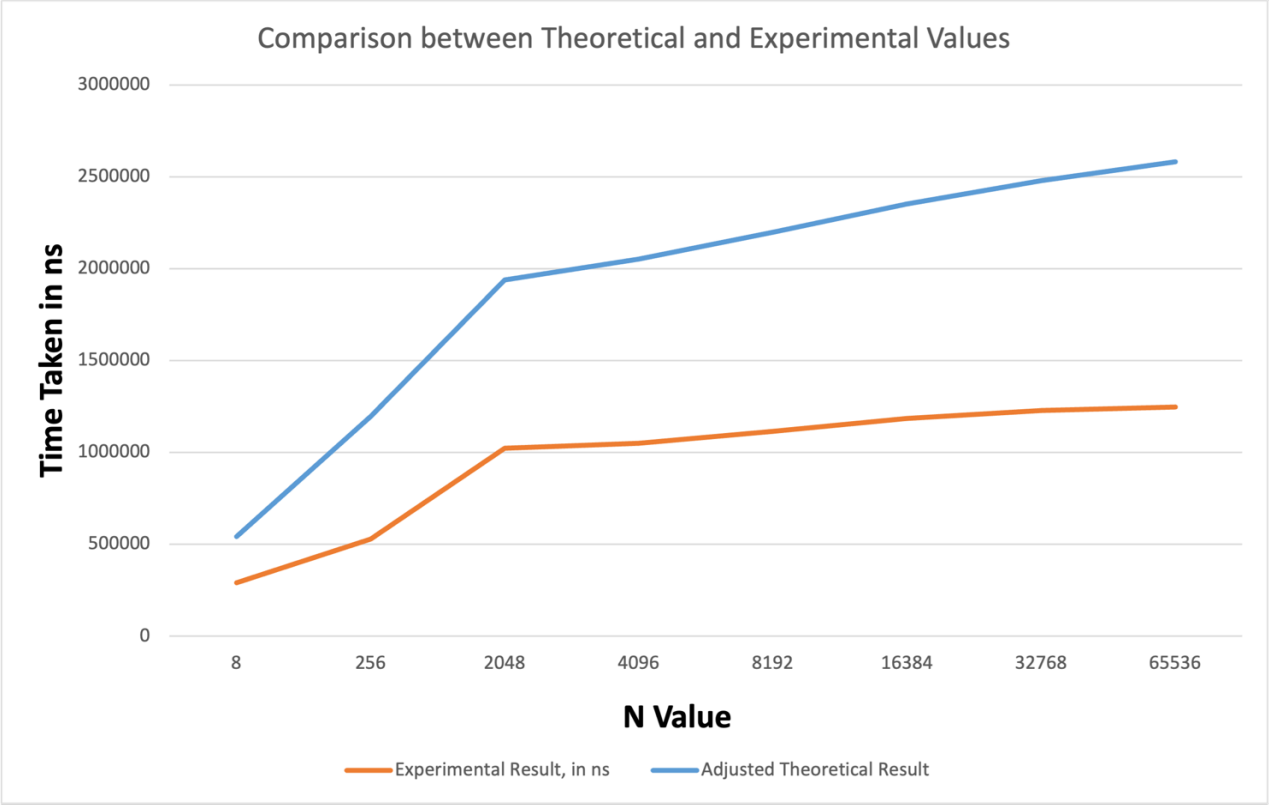
The Scaling Constant after the calculations comes out to be 83403.8913.

3.3 Output Numerical Data

n	Experimental Result, in ns	Theoretical Result	Scaling Constant	Adjusted Theoretical Result
8	292639	3		250211.674

256	529973	8		667231.13
2048	1022548	11		917442.804
4096	1050521	12		1000846.7
8192	1114045	13		1084250.59
16384	1185199	14		1167654.48
32768	1229186	15		1251058.37
65536	1249047	16		1334462.26
	959144.75	11.5	83403.8913	

3.4 Graph



3.5 Graph Observations

From the graph, we can observe that the theoretical calculations and the experimental calculations are close to each other. The adjusted theoretical result grows slightly faster than the experimental result. The graph also resembles the log n graph.

4 Conclusion

The Time Complexity of the Code is **O(log n)**, so we can conclude that the theoretical and experimental graphs grow very similarly and resemble the log n graph. Hence, we can conclude that our calculations are correct.