PROJECT 1

Name: Yashwanth Raj Varadharajan
Class: CS6212
GWID: G47635180

Problem Statement

We are required to analyze the following program sample.

```
int j = 2
while (j < n) {
  int k = j
  while (k < n) {
    Sum += a[j]*b[k]
    k=k*k
  }
  j=2*j
}
```

Q1) What is the time complexity of this algorithm, in terms of n?

Ans: O((log n)*(log(log n)). //Big Oh

Q2) Explain your theoretical analysis?

Ans: When we analyze the code, we will notice that there are two loops that we have to consider.

- The first is the **outer loop**, while loop with j variable. With every iteration, the j is multiplied by 2. And the value of j is given as 2.
  Value of j $\Rightarrow 2 , 2^2 , 2^3 \ldots\ldots 2^x$ //where $2^x >= n$
  Now we assume that after x iterations, the value of $2^x >= n$. At which point it comes out the loop.
  Calculations to find x - $2^x = n \Rightarrow x = \log n$
  Hence the time complexity for this loop can be calculated as: log n
- The second is the **inner loop**, while loop with k variable. With every iteration, the k is squared. And as we know the value of k = j given before the loop, the k starts with 2.
  Value of k $\Rightarrow 2 , 2^2 , 2^4 , 2^8 , \ldots 2^y$ //where $2^y >= n$
  If we look at the exponents only, $1 , 2 , 2^2 , 2^3 \ldots\ldots 2^z$ Where z denotes the number of iterations.
  Hence the expressions for iteration becomes $(2^2)^z$. Assuming after z iterations the value becomes > n.
  Calculations to find z - $(2^2)^z = n. \Rightarrow z = \log (\log n)$
  Hence the time complexity can be calculated as: log (log n)

Since one loop is inside the other Now after multiplying time complexity of both the loops,
we get O((log n)*(log(log n)).

Q3) Create a program to include and run the code for different values of n and document time taken for different values of n.

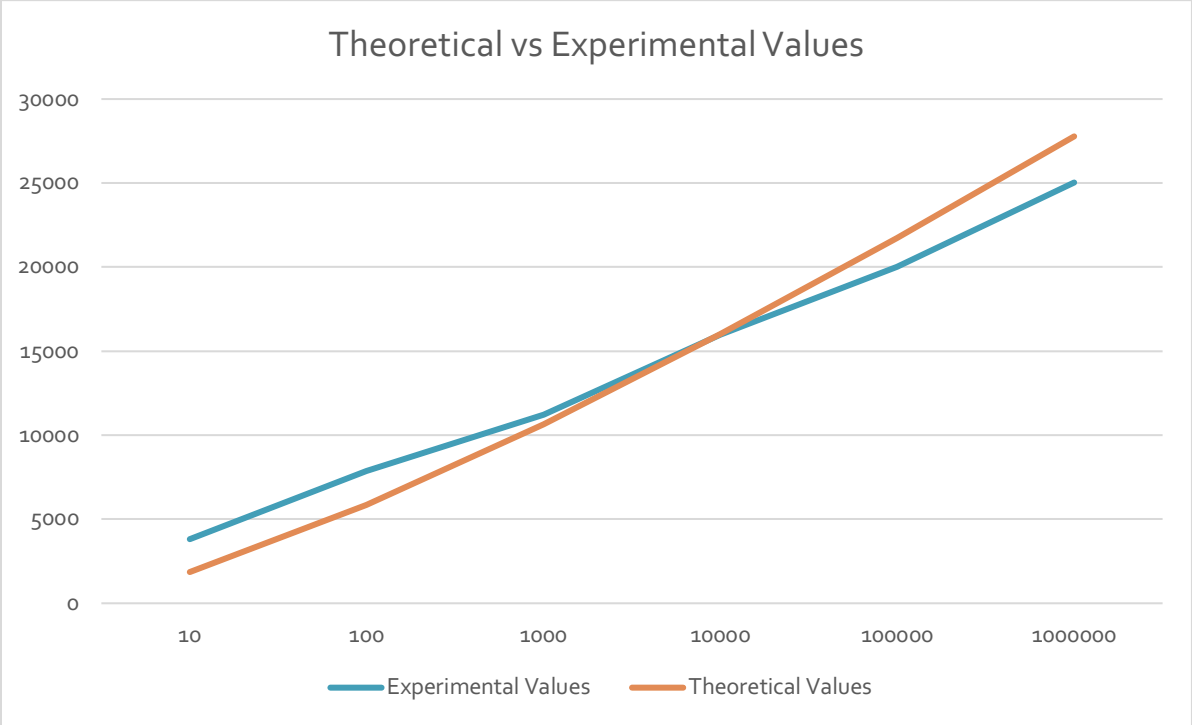
Ans: We have formulated and run the code for different values of n ranging from 10 to 10^6, and noted down the time taken for each n value. We also substitute the values of n in time complexity expression that we have derived and tabulate all the results.
Now we notice that the experimental time is in nanoseconds, but the theoretical time are constants, hence, to plot graphs, we need to multiply all the theoretical values by scaling constant.

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

The average of experimental values is 13987.2233 and the average of theoretical values is 43.2499167. Hence dividing them, we get the scaling constant as 323.404631, which we multiply with theoretical values to get the adjusted values that we can plot.

n	Experimental Value in ns	Theoretical value ((log n)*(log(log n))	Scaling constant	Adjusted Theoretical Result (Average Method)
10	3814.697	5.7463	323.404631	1858.38003
100	7867.813	18.1272	323.404631	5862.420424
1000	11205.673	32.9676	323.404631	10661.87451
10000	15974.045	49.5344	323.404631	16019.65435
100000	20027.161	67.2301	323.404631	21742.52567
1000000	25033.951	85.8939	323.404631	27778.48502

Q4) Draw a graph of theoretical results and experimental results to compare.
Ans:



Graph Observation
From the graph we can observe that the theoretical calculation and the experimental calculations are very close to each other. Till n = 10000, the experimental values are higher, after which the theoretical values are higher. Overall, theoretical result grows slightly faster than the experimental result.

Conclusion

The provided code has a time complexity of **O((log n)*(log n))**. We can draw the conclusion that both experimental and theoretical graphs grow quite similarly to one another, indicating that our calculations are accurate.