CSCI 6212: Design and Analysis of Algorithm – Project 1

Shubham Jadhav | G30570862 | Sept 13. 2022

1. Problem Statement

We are required to analyze the following program/code sample.

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

2. Theoretical Analysis

We can break down the program into 2 parts, i.e., Inner loop and outer loop.

For inner loop:

We can ignore **Sum += a[k]*b[k],** as the computation of this equation will be constant.

The loop runs from **k to n** (excluding **n**), by updating the value of **k** by **k*sqrt(k)**

Here k=2 at start.

For iteration 1: $k = k*sqrt(k) = k*k^{1/2} = k^{3/2} = 2^{3/2}$

For iteration 2: $k = k*sqrt(k) = k*k^{1/2}$, substituting value of $k=k^{3/2}$ then $k=(k^{3/2})*(k^{3/2})^{1/2} = k^{9/4} = 2^{9/4}$

Similarly For iteration 3: $k = k^{27/8} = 2^{27/8}$

We can re write the pattern as for l^{th} iteration $k = 2^{\frac{3^l}{2^l}}$

So, $n > 2^{\frac{3^l}{2^l}}$. That is same as $n > 2^{\left(\frac{3}{2}\right)^l}$

Taking log on both sides

$\log(n) > \left(\frac{3}{2}\right)^l * \log(2)$ ----- we can ignore log(2) as it a constant

Again, taking log on both sides

$\log(\log(n)) > l * log\left(\frac{3}{2}\right)$ ----- similarly, we can ignore log(3/2) as it a constant

$\log(\log(n)) > l$

Therefore, the time complexity for inner loop is **O(log(log(n)))**

For outer loop:

The loop runs from **j to n** (excluding **n**), in increment of $\frac{j}{2}$

Here j=2 at start

For iteration 1: j = $j = j + \frac{j}{2} = \frac{3j}{2} = 3$

For iteration 2: k $= j + \frac{j}{2}$, substituting value of j=$\frac{3j}{2}$ then j= $\frac{9j}{4} = \frac{9}{2}$

Similarly For iteration 3: j = $\frac{27j}{8} = \frac{27}{4}$

We can re write the pattern as for l^{th} iteration **j =2** $* \left(\frac{3^l}{2^l}\right) = \mathbf{2} * \left(\frac{3}{2}\right)^{\boldsymbol{l}}$

So, **n > 2** $* \left(\frac{3}{2}\right)^{\boldsymbol{l}}$ ----- we can ignore 2 as it a constant

n > $\left(\frac{3}{2}\right)^{\boldsymbol{l}}$

Taking log on both sides

log(n) > $\boldsymbol{l} * \boldsymbol{log}(\frac{3}{2})$ ----- we can ignore log(3/2) as it a constant

Therefore, the time complexity for outer loop is **O(log(n))**

Combining both the time complexity:

The final time complexity of the entire code is **O(log(n)*log(log(n))).**

3. Experimental Analysis

3.1 Program Listing

For experimental analysis, a python program was coded using the sample code from the project question. The program was executed with values of n as; 10^2, 10^3, 10^4, 10^5, and 10^6. The program also calculates the value of the theoretical time complexity derived above for the same n values. The values of theoretical, experimental, and adjusted values of theoretical are presented in sub section 3.3.

3.2 Data Normalization

The theoretical and experimental results for selected n values vary and cannot be compared directly or does not present a viable insight. Therefore, we need to derive a scaling constant to adjust either or the list of values. To derive the constant, we calculated the mean of both list of time values and found the ratio as follows

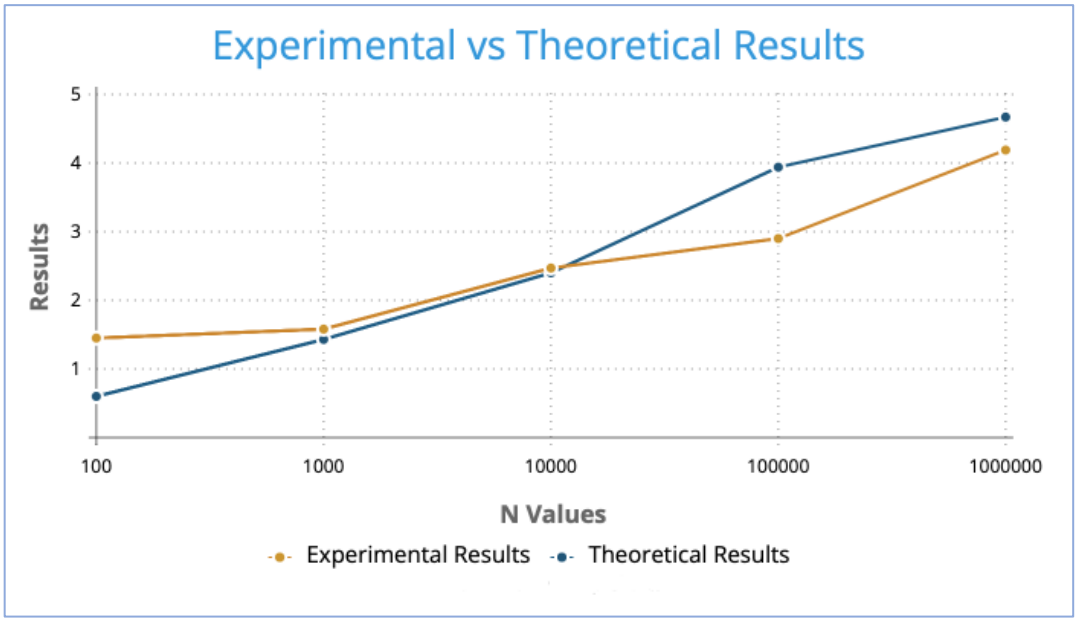
Scaling constant = $\frac{mean\ of\ experimentally\ derived\ time\ values}{mean\ of\ theoretically\ derived\ time\ values} \approx 110000$

Note: we cannot calculate a fixed constant as experimental time values keeps on changing

3.3 Output Numerical Data

n	Theoretical	Experimental	Adjusted Experimental
100	0.60	$1.19 \times e^{-5}$	1.35
1000	1.43	$1.47 \times e^{-5}$	1.67
10000	2.40	$2.21 \times e^{-5}$	2.51
100000	3.49	$2.62 \times e^{-5}$	2.97
1000000	4.66	$3.62 \times e^{-5}$	4.10

3.4 Graph



3.5 Graph Observation

It can be seen from the above graph that, as value of n increases both, theoretical as well as experimental results poses the same pattern/behavior; increases with value of n.

4. Conclusion

In conclusion, derived theoretical complexity of the program seems accurate and consistent when compared with the normalized experimental complexity results using scaling constant from the observations and insights gained from the visualization in section 3.4.

GitHub Link: *https://github.com/shubhjadhav/CSCI_6212/blob/main/Project1.py*