# Project 1

*Hetal Rakesh Lad*

## 1 Problem Statement

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

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

## 2 Theoretical Analysis

*The inner loop k is independent of the outer loop j. The inner loop k runs till n and grows at the rate of* 22^k*. The outer loop also runs n times but grows at the rate of*

*Mathematical expressions.*

*For inner loop k=j*

*When J=2 -> k=2,4,16,256..till n*

*This is a growth of* 22^k *, therefore T(n)=𝜃(log log n)*

*For outer loop j till j<n*

*J=2,4,8,16…. n*

*This is a growth of 2l, therefore(n)= 𝜃(log n)*

*Thus, total work= inner \* outer loop*

*= log n\* log log n*

## 3 Experimental Analysis

### 3.1 Program Listing

A screen shot of a computer program

AI-generated content may be incorrect.

My input is from 22 to 220.

### 3.2 Data Normalization Notes

Do you normalize the values by some constant? How did you derive that constant?

* Yes, I did normalize my theoretical values since the actual time depends on CPU speed, memory interpreter, extra.
* Theoretical-scaled​(n)=C⋅log2​n⋅log2​(log2​n)
* C= theoretical nmax / actual nmax
* **Division by theoretical(nmax)** is to normalize the curve
* **Multiply by actual(nmax)** is to scale it to real measured time

Nmax is the largest n in my data.

### 3.3 Output Numerical Data

| **Input Size (n)** | **Actual Time (seconds)** | **Theoretical Time (seconds)** |
| --- | --- | --- |
| 4 | 0.0000019170 | 0.0000001089 |
| 8 | 0.0000010420 | 0.0000002590 |
| 16 | 0.0000005830 | 0.0000004357 |
| 32 | 0.0000007080 | 0.0000006323 |
| 64 | 0.0000008750 | 0.0000008448 |
| 128 | 0.0000010830 | 0.0000010703 |
| 256 | 0.0000012500 | 0.0000013072 |
| 512 | 0.0000020410 | 0.0000015539 |
| 1,024 | 0.0000019160 | 0.0000018093 |
| 2,048 | 0.0000029590 | 0.0000020727 |
| 4,096 | 0.0000040410 | 0.0000023431 |
| 8,192 | 0.0000036250 | 0.0000026202 |
| 16,384 | 0.0000039590 | 0.0000029032 |
| 32,768 | 0.0000035000 | 0.0000031919 |
| 65,536 | 0.0000031250 | 0.0000034859 |
| 131,072 | 0.0000039580 | 0.0000037847 |
| 262,144 | 0.0000033750 | 0.0000040882 |
| 524,288 | 0.0000035830 | 0.0000043960 |
| 1,048,576 | 0.0000047080 | 0.0000047080 |

**3.4 Graph** A graph with a line and a line

AI-generated content may be incorrect.

(The graph is generated using matplotlib in python, the code is present in githublink)

**3.5 Graph Observations**

* For small input sizes, the times are extremely small (almost negligible) due to system and interpreter overheads, they become more noticeable for larger n.
* Initially the theoretical and actual times don’t match, but as the input size grows they tend to match, but they tend to converge for larger n values.
* Both the curves become nearly parallel for larger inputs, indicating the theoretical complexity is a good predictor for actual runtime.

## 4 Conclusions

The discrepancy at small input sizes is due to fixed overheads, but for larger input sizes the actual runtime aligns closely with the theoretical model.

This confirms that the nested loop’s time complexity is nearly log n \* log(log n).

GITHUB LINK :

<https://github.com/HetalLad/CSCI6212_project1/>