1. Problem Statement

Find nth power of a complex number. The program should run in O(log n) time.

2. Theoretical Analysis

Reasoning

Multiplying (or squaring) complex numbers is a constant time operation:

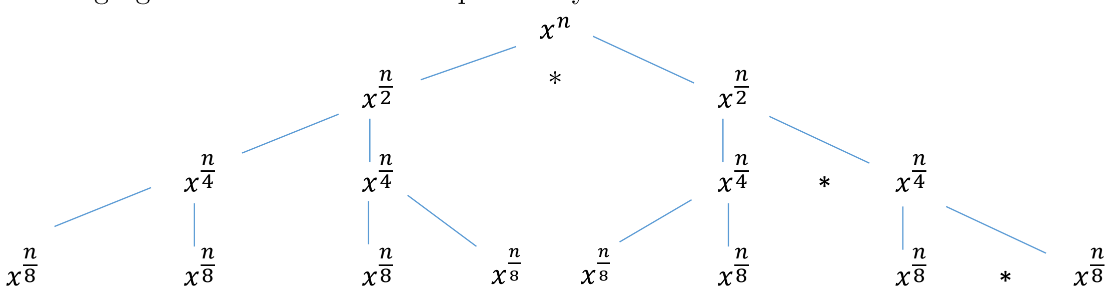
$z1 = a + ib$, $z2 = x + iy$ are two complex numbers which yield $z1 * z2 = (ax - by) + i(ay + bx)$ when multiplied → O(1).

Alternatively, a complex number in rectangular form can be converted to its polar form [O(1)]. Angle is simply multiplied with 'n' [O(1)], and power of the magnitude is computed just like that of any other real number. This approach was used and is explained below.

Multiplying a complex number 'n' times with itself results in O(n).

In order to achieve O(log(n)), a divide and conquer technique with recursion formula T(n)=T(n/b)+c (Second case of Master Theorem) is an option.

The following figure illustrates one such possibility where n=8:



At each step, only one multiplication is required [O(1)] and a total of log(n) steps exist because n is halved at each step. If, at any step, the power of x is an odd number, an additional x is multiplied at the subsequent step.

Mathematical Expressions

The recurrence relation can be summarized as T(n) = T(n/2) + c. Using the second case of Master Theorem, we can see that time complexity of the program is O(log(n)).

3. Experimental Analysis

Program Listing

The code is available at https://github.com/Irtaza-Sohail/CSCI-6212/blob/dccf56eeb5048e2432178079820c11df080b58ff/Project%202%20-%20Measuring%20Time%20Complexity.py

Data Normalization Notes

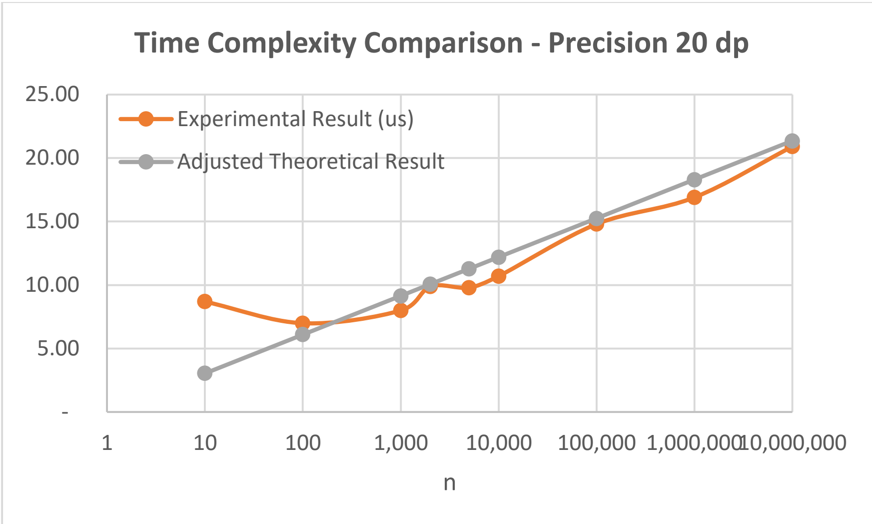
Data was normalized using a constant obtained from the ratio of average experimental to average theoretical data.

Output Numerical Data

N	Experimental Average Time (µs)	Theoretical Value	Adjusted Theoretical Values
10	8.70	3.32	3.05
100	7.00	6.64	6.10
1,000	8.00	9.97	9.15
2,000	9.90	10.97	10.06
5,000	9.80	12.29	11.28
10,000	10.70	13.29	12.19

100,000	14.80	16.61	15.24
1,000,000	16.90	19.93	18.29
10,000,000	20.90	23.25	21.34

Graph



Graph Observations

The experimental results agree with theoretical prediction over the course of tested values of n.

4. Conclusions

As expected, the theoretical predictions and experimental results are in agreement. Interestingly, using numbers with high precision results in deviation from the theoretical predictions. Using very high precision for numbers (to 99,999,999,999 decimal places) causes the program to execute in linear time. A smaller number of decimal places (to 1000 dp) causes significant deviation from theoretical predictions. These trends are shown in graphs below.