COA Assignment 1

Q1: Fibonacci Numbers Performance

Results:

1. **Time Taken:**
   - **Recursion:** 163.349 seconds
   - **Recursion with Memoization:** 0.00006442 seconds
   - **Loop:** 0.00007166 seconds
   - **Loop with Memoization:** 0.000062671 seconds
2. **Speedup Comparison (Relative to Recursion):**
   - **Memoization:** 2,535,687 times faster
   - **Loop:** 2,295,517 times faster
   - **Loop with Memoization:** 2,606,452 times faster

Observations

1. **Efficiency:**
   - **Recursion** is the slowest (163.349 seconds), highlighting its inefficiency.
   - **Recursion with Memoization** (0.00006442 seconds) and **Loop** (0.00007166 seconds) are significantly faster, with memoized recursion being slightly quicker.
   - **Loop with Memoization** (0.000062671 seconds) is the fastest, combining the benefits of iteration and memoization.
2. **Speedup:**
   - **Memoization** is 2.54 million times faster than recursion.
   - **Loop** is 2.30 million times faster than recursion.
   - **Loop with Memoization** is 2.61 million times faster than recursion.

Q2: Matrix Multiplication Performance

Results:

1. **C++ Performance:**
   - **Integer Matrix:**
     - 64x64: 0.00303 seconds
     - 128x128: 0.033435 seconds
     - 256x256: 0.226099 seconds
     - 512x512: 1.89039 seconds
     - 1024x1024: 14.0054 seconds
   - **Double Matrix:**
     - 64x64: 0.005422 seconds

        - 128x128: 0.027777 seconds
        - 256x256: 0.247847 seconds
        - 512x512: 2.00949 seconds
        - 1024x1024: 17.3577 seconds

2. **Python Performance:**
    - **Integer Matrix:**
        - 64x64: 0.000196 seconds
        - 128x128: 0.003137 seconds
        - 256x256: 0.022210 seconds
        - 512x512: 0.387345 seconds
        - 1024x1024: 5.907104 seconds

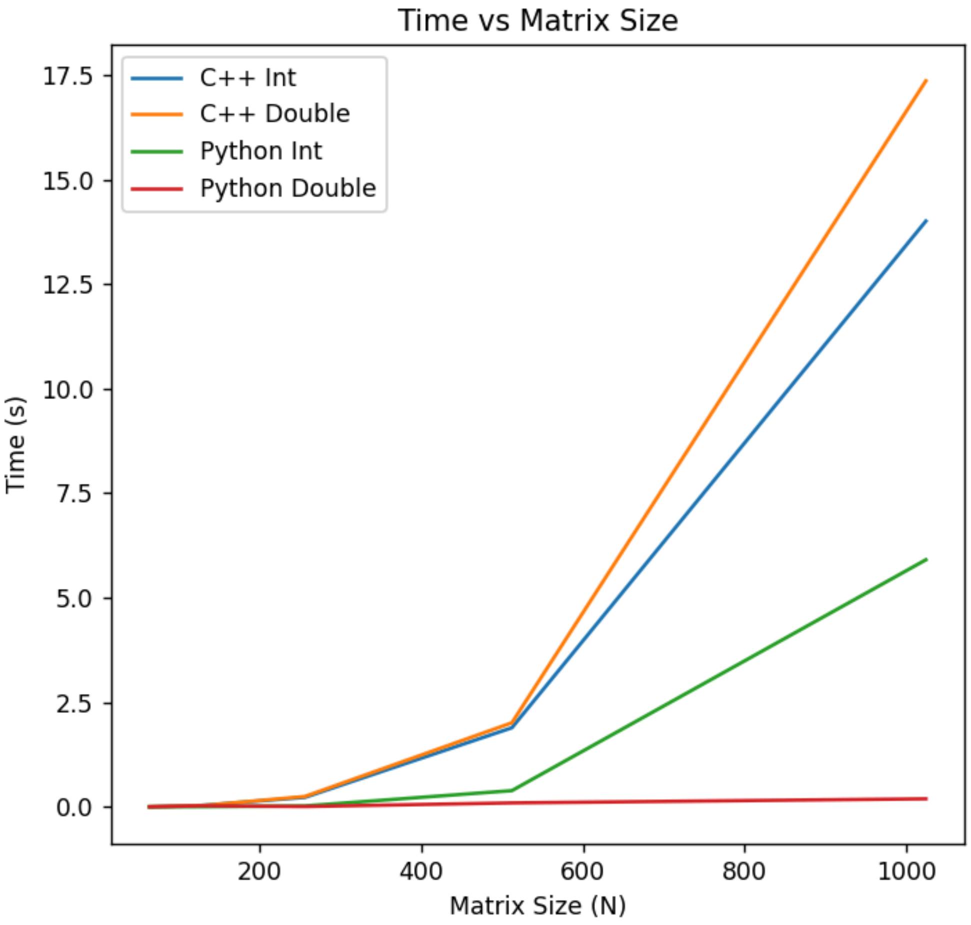
    Double Matrix:

        - 64x64: 0.000210 seconds
        - 128x128: 0.025841 seconds
        - 256x256: 0.008276 seconds
        - 512x512: 0.094132 seconds
        - 1024x1024: 0.191961 seconds

Observation:

1. **C++ Performance:**
    - **Integer Matrix**: Time increases from 0.00303 seconds (64x64) to 14.0054 seconds (1024x1024), reflecting quadratic growth.
    - **Double Matrix**: Slightly slower than integer matrices, with times rising from 0.005422 seconds to 17.3577 seconds.
2. **Python Performance:**
    - **Integer Matrix**: Faster for small matrices but slower for large ones, with times from 0.000196 seconds to 5.907104 seconds.
    - **Double Matrix**: Similar trend to integer matrices, with times from 0.000210 seconds to 0.191961 seconds.

Time vs. Matrix Size Plot: The plot shows the relationship between matrix size and computation time for both Python and C++ implementations, for integer and double precision matrices.


Time vs Matrix Size
C++ Int
C++ Double
Python Int
Python Double
Time (s)
Matrix Size (N)