a. How does the execution time change from sequential to threaded to multiprocessing?

- **Sequential: 0.24s**
- **Threaded: 0.25s**
- **Multiprocessing: 0.77s**

Threading does not improve performance significantly because Python's GIL prevents true parallel execution of CPU-bound tasks. **Multiprocessing is actually slower** in this case due to the overhead of managing processes and communication.

b. Speedup, Efficiency, Amdahl's Law, Gustafson's Law

Metric	Threaded	Multiprocessing
Speedup	0.98x	0.31x
Efficiency	24.4%	7.8%
Amdahl's Speedup	3.08x	3.08x
Gustafson's Speedup	3.7x	3.7x

c. Are there performance differences between threaded and multiprocessing versions?

Yes, Threading is slightly better but doesn't give real parallelism due to the GIL. Multiprocessing is worse in this case due to process creation and memory sharing overhead

d. What challenges did you face when implementing parallelism?

- Threading issue
- Multiprocessing overhead
- Dividing workload properly

e. When to choose threading vs multiprocessing?

Scenario	Use Threading	Use Multiprocessing
CPU-bound tasks (heavy calculations)	□ No	□ Yes
I/O-bound tasks (file read, web requests)	□ Yes	□ No
Low memory usage required	□ Yes	□ No
High performance gain needed	□ No	□ Yes (if managed well)