Project Plan: High-Performance Graph & Matrix Engine

Tech stack: C++17/20, Google Benchmark, GoogleTest, perf/Valgrind
Target duration: 6–7 weeks (~10–12 hrs/week)
Focus: Graph + Matrix operations with concurrency + SIMD optimizations

Phase 1: Setup & Foundations (Week 1)

- Setup GitHub repo, CI/CD (GitHub Actions or Make/CMake)
- Create project structure (/include, /src, /tests, /benchmarks, /docs)
- Implement CSR (Compressed Sparse Row) graph representation
- Implement blocked matrix representation (e.g., 64×64 tiles)
- Add aligned memory allocator (std::pmr::memory_resource or aligned_alloc)
- Add GoogleTest for unit tests + test CSR + matrix ops

Deliverable: GitHub repo with builds/tests passing, CSR + blocked matrix working

Phase 2: Thread Pool & Concurrency Base (Week 2)

- Implement thread pool with fixed thread count
- Work queue (mutex or lock-free) + condition variable
- Submit tasks API + worker loop
- Optional: Work stealing queues
- Unit tests: submit 100 tasks, validate correctness

Deliverable: Thread pool API with tests showing parallelism

Phase 3: Matrix Multiplication (GEMM) Optimizations (Weeks 3–4)

- Implement naive GEMM
- Implement blocked GEMM (cache-friendly tiling)
- Add SIMD GEMM (AVX2 intrinsics)
- Benchmark with Google Benchmark
- Profile with perf (cache misses, FLOPs)

Deliverable: Benchmarks with performance chart (SIMD ~4–5× faster)

Phase 4: Sparse Matrix-Vector Multiply (SpMV) (Week 4–5)

- Implement naive SpMV
- Implement optimized CSR SpMV with prefetch
- Parallelize row computations with thread pool
- Benchmarks with sparse matrices at varying densities

Deliverable: Optimized SpMV (2–3× faster than dense baseline)

Phase 5: Graph Algorithms (Week 5)

- Implement parallel BFS using thread pool + CSR frontier expansion
- Implement PageRank (parallel power iteration)
- Benchmark BFS throughput (MTEPS) & PageRank convergence

Deliverable: BFS (300–500M edges/s), PageRank (~2× faster)

Phase 6: Benchmarking, Docs & Final Polish (Week 6–7)

- Add benchmark suite + profiling scripts (perf, valgrind, gprof)
- Document design decisions, performance graphs
- Create README with overview + results
- Example usage program (matrix GEMM, BFS, PageRank)
- Polish with clang-format

Deliverable: GitHub repo with code, tests, benchmarks, docs, results

■ Final Deliverables

- GitHub Repo with /include, /src, /tests, /benchmarks, /docs
- Performance Benchmarks: GEMM speedup, SpMV vs density, BFS throughput (MTEPS)
- Documentation: Design decisions, usage examples, performance results
- Resume-ready project summary + example usage