Convex Hull Project - Task Breakdown & Assignments

Team Size: 3 People

Project Duration: ~3-4 weeks (adjust based on your deadline)

Phase 0: Initial Setup & Planning (Week 1 - Days 1-2)

EVERYONE TOGETHER - 4-6 hours

Tasks:

1. Interface Design Session (2-3 hours)
   - Define common data structures for points and convex hulls
   - Agree on function signatures for each algorithm
   - Define input/output formats (file formats, data structures)
   - Agree on language choice (or stay language-agnostic initially)
   - Design module structure and file organization
2. Testing Strategy (1 hour)
   - Decide on test input sizes (e.g., 10^3, 10^4, 10^5, 10^6, 10^7)
   - Plan test case generation (square, circle, parabola $Y=-X^2$)
   - Discuss additional test cases (if any)
3. Repository Setup (1 hour)
   - Create Git repository
   - Initialize project structure
   - Set up basic README
   - Create initial branch structure

Deliverables:

- [ ] Interface specification document (markdown)
- [ ] Repository structure
- [ ] Initial project timeline

Phase 1: Core Implementation (Week 1-2)

EVERYONE TOGETHER - Implementation sessions

Collaborative Implementation Tasks:

All three team members work together on implementing the algorithms through pair/mob programming sessions. This ensures:

- Knowledge sharing across all algorithms
- Consistent code quality
- Better bug detection
- Shared understanding of implementation details

Session 1: Graham's Scan (4-6 hours)

- [ ] Implement point sorting by x-coordinate
- [ ] Implement orientation test (CCW/CW)
- [ ] Implement incremental hull construction
- [ ] Basic correctness testing

Session 2: QuickHull (4-6 hours)

- [ ] Implement point-to-line distance calculation
- [ ] Implement pruning logic
- [ ] Implement recursive partitioning
- [ ] Basic correctness testing

Session 3: Marriage-before-Conquest (6-8 hours)

- [ ] Implement median finding (or random point selection)
- [ ] Implement Linear Programming for bridge finding (critical!)
- [ ] Implement pruning logic
- [ ] Implement MbC_CH basic version
- [ ] Basic correctness testing

Session 4: MbC2 Enhancement (2-3 hours)

- [ ] Add extra pruning step to MbC
- [ ] Test MbC2_CH version
- [ ] Verify correctness

Phase 2: Parallel Workstreams (Week 2-3)

SPLIT WORK - Each person takes ownership of one track

Track A: CI/CD & Metrics Pipeline

Assigned to: Person 1

Tasks:

1. Test Case Generator (4–5 hours)
   - [ ] Implement point generator for square distribution
   - [ ] Implement point generator for circle distribution
   - [ ] Implement point generator for parabola $Y=-X^2$
   - [ ] Implement additional test cases (optional)
   - [ ] Save generated test cases to files

2. Automated Testing Pipeline (3–4 hours)
   - [ ] Create script to run all algorithms on all test cases
   - [ ] Implement correctness verification (compare outputs)
   - [ ] Set up timing measurements
   - [ ] Implement software counters (optional but recommended)

3. Metrics Collection (3–4 hours)
   - [ ] Collect running times (total, reading, computation, sorting)
   - [ ] Count points on convex hull for each test
   - [ ] Export metrics to structured format (CSV/JSON)

4. Plotting Scripts (4–5 hours)
   - [ ] Plot running time vs input size for each algorithm
   - [ ] Plot running time comparisons across algorithms
   - [ ] Plot number of hull points vs input size (square & circle cases)
   - [ ] Create plots for different input classes
   - [ ] Use appropriate logarithmic scales
   - [ ] Generate publication-ready figures

5. CI/CD Setup (2–3 hours)
   - [ ] Create GitHub Actions / GitLab CI configuration
   - [ ] Automate testing on commits
   - [ ] Generate and store plots as artifacts
   - [ ] Set up automated reporting (optional)

Time Estimate: 16–21 hours

Track B: Reporting & Analysis

Assigned to: Person 2

Tasks:

1. Algorithm Descriptions (4-5 hours)
   - [ ] Write description of Graham's Scan
   - [ ] Write description of QuickHull
   - [ ] Write description of MbC and MbC2
   - [ ] Document theoretical running times
   - [ ] Document worst-case analysis for QuickHull
   - [ ] Document any implementation changes/optimizations

2. Input Class Analysis (3-4 hours)
   - [ ] Describe each input class (square, circle, parabola)
   - [ ] Formulate hypotheses about hull sizes
   - [ ] Analyze expected number of hull points (theoretical)
   - [ ] Discuss impact on algorithm performance

3. Experimental Design Documentation (2-3 hours)
   - [ ] Document experiment plan
   - [ ] Describe performance metrics chosen
   - [ ] Explain hypothesis for each experiment
   - [ ] Document machine specifications

4. Results Analysis (5-6 hours)
   - [ ] Analyze plots and metrics
   - [ ] Compare empirical results with theoretical predictions
   - [ ] Identify interesting patterns or anomalies
   - [ ] Investigate unexpected behaviors
   - [ ] Write findings and interpretations

5. LP Implementation Documentation (2-3 hours)
   - [ ] Document LP solution approach for MbC
   - [ ] Include code snippets
   - [ ] Explain implementation decisions
   - [ ] Discuss efficiency considerations

6. Report Assembly (3-4 hours)
   - [ ] Integrate all sections
   - [ ] Add plots and figures
   - [ ] Proofread and edit
   - [ ] Format according to requirements
   - [ ] Create final PDF

Time Estimate: 19-25 hours

Track C: Testing, Debugging & Optimization

Assigned to: Person 3

Tasks:

1. Comprehensive Testing (5–6 hours)
   - [ ] Create unit tests for geometric primitives (CCW test, distance, etc.)
   - [ ] Create correctness tests with known outputs
   - [ ] Test edge cases (collinear points, duplicates, etc.)
   - [ ] Test small inputs manually
   - [ ] Verify all algorithms produce correct results

2. Large-scale Testing (3–4 hours)
   - [ ] Test with $n = 10^5$ points
   - [ ] Test with $n = 10^6$ points
   - [ ] Test with $n = 10^7$ points (if feasible)
   - [ ] Monitor memory usage
   - [ ] Identify any performance bottlenecks

3. Debugging & Bug Fixes (4–5 hours)
   - [ ] Fix any correctness issues found
   - [ ] Address edge case failures
   - [ ] Resolve numerical precision issues
   - [ ] Fix performance issues

4. Code Quality (3–4 hours)
   - [ ] Add comments and documentation
   - [ ] Refactor for clarity
   - [ ] Ensure consistent coding style
   - [ ] Create API documentation

5. Performance Profiling (3–4 hours)
   - [ ] Profile each algorithm
   - [ ] Identify hotspots
   - [ ] Measure time for algorithm sub-components
   - [ ] Document findings

6. Optional Optimizations (2–3 hours)
   - [ ] Optimize critical paths if needed
   - [ ] Consider algorithmic improvements
   - [ ] Balance optimization vs. report timeline

Time Estimate: 20–26 hours

Phase 3: Integration & Finalization (Week 3–4)

EVERYONE TOGETHER – 6–8 hours

Tasks:

1. Integration Meeting (2–3 hours)
    - Review all completed work
    - Integrate plots into report
    - Discuss findings and interpretations
    - Resolve any inconsistencies
2. Final Experiments (2–3 hours)
    - Run final experiment suite
    - Generate final plots
    - Verify all data is collected
3. Report Finalization (2–3 hours)
    - Final review of report
    - Add any missing sections
    - Proofread
    - Format and polish
4. Code Submission (1 hour)
    - Clean up repository
    - Add final documentation
    - Create submission package
    - Test submission on fresh environment

Deliverables:

- [ ] Complete implementation (all 4 algorithms)
- [ ] Comprehensive test suite
- [ ] CI/CD pipeline
- [ ] All plots and figures
- [ ] Final report (PDF)
- [ ] Clean, documented code repository

Work Balance Summary

Person	Primary Track	Estimated Hours
Person 1	CI/CD & Metrics	16–21 hours
Person 2	Reporting & Analysis	19–25 hours
Person 3	Testing & Quality	20–26 hours

Plus shared time:

- Phase 0: 4–6 hours (everyone)
- Phase 1: 16–23 hours (everyone)
- Phase 3: 6–8 hours (everyone)

Total per person: ~46–60 hours

Key Milestones

- End of Week 1: Interface defined, Graham's Scan implemented
- End of Week 2: All algorithms implemented, parallel tracks 50% complete
- End of Week 3: All parallel tracks complete, integration started
- End of Week 4: Final report submitted

Communication & Coordination

Weekly Sync Meetings

- Monday: Plan week's work, address blockers
- Thursday: Progress check, adjust plans if needed
- Sunday: Review completed work

Daily Communication

- Use Slack/Discord for quick questions
- Share progress updates in shared doc
- Flag blockers immediately

Code Review Process

- All code changes reviewed by at least one other team member
- Use pull requests for major changes
- Quick reviews for bug fixes

Important Notes

1. LP Implementation: This is critical for MbC. Person 2 should help document this thoroughly since it's required in the report.
2. Large Input Sizes: The project emphasizes testing with large inputs (>10⁴). Person 3 should coordinate with Person 1 to ensure the CI/CD pipeline can handle these.
3. Algorithmic Horse-Race: Be mindful of this trap (mentioned in project description). Person 2 should discuss this in the report.
4. Hull Size Plots: These are explicitly required for square and circle cases. Person 1 must prioritize these plots.
5. Flexibility: If someone finishes early, they should help others or work on "optional" improvements.

Risk Management

Risk	Mitigation
LP implementation too complex	Start early, seek TA help, review lecture exercises
Algorithms too slow for large inputs	Start with smaller sizes, optimize incrementally
Person unavailable	Cross-train during Phase 1, document everything
Report takes longer than expected	Start writing early, use templates
Merge conflicts	Frequent small commits, clear module boundaries

Success Criteria

- ✅ All 4 algorithms implemented and correct
- ✅ Tests pass for inputs up to 10^6+ points
- ✅ All required plots generated
- ✅ Report includes all required sections
- ✅ LP implementation documented with code snippets
- ✅ Empirical results compared with theory
- ✅ Clean, documented codebase