# Shadows Execution Time Analysis

Paul Scanlon and Matt Murdoch

*Colorado State University, Research for Dr. Sanjay Rajopadhye, Summer 2019*

---

## Original Implementation

Our original implementation of Shadows requires $i * j$ iterations where $j < i$. Thus, it has a run-time of $\mathbb{O}(N^2)$. Data supporting this is shown below.

| | | | | | |
|---|---|---|---|---|---|
| 1 | 5 | 1.00E-06 | 31 | 5000 | 0.019512 |
| 2 | 5 | 1.00E-06 | 32 | 5000 | 0.008757 |
| 3 | 5 | 1.00E-06 | 33 | 5000 | 0.009168 |
| 4 | 5 | 1.00E-06 | 34 | 5000 | 0.009227 |
| 5 | 5 | 1.00E-06 | 35 | 5000 | 0.009493 |
| 6 | 5 | 1.00E-06 | 36 | 5000 | 0.017708 |
| 7 | 5 | 1.00E-06 | 37 | 5000 | 0.016529 |
| 8 | 5 | 1.00E-06 | 38 | 5000 | 0.019779 |
| 9 | 5 | 1.00E-06 | 39 | 5000 | 0.018541 |
| 10 | 5 | 1.00E-06 | 40 | 5000 | 0.01943 |
| 11 | 50 | 8.00E-06 | 41 | 50000 | 1.164899 |
| 12 | 50 | 5.00E-06 | 42 | 50000 | 1.200309 |
| 13 | 50 | 6.00E-06 | 43 | 50000 | 1.18111 |
| 14 | 50 | 7.00E-06 | 44 | 50000 | 1.159519 |
| 15 | 50 | 6.00E-06 | 45 | 50000 | 1.175823 |
| 16 | 50 | 7.00E-06 | 46 | 50000 | 1.1572 |
| 17 | 50 | 6.00E-06 | 47 | 50000 | 1.13815 |
| 18 | 50 | 4.00E-06 | 48 | 50000 | 0.290836 |
| 19 | 50 | 4.00E-06 | 49 | 50000 | 0.296523 |
| 20 | 50 | 6.00E-06 | 50 | 50000 | 1.176133 |
| 21 | 500 | 0.000422 | 51 | 500000 | 113.62362 |
| 22 | 500 | 0.00038 | 52 | 500000 | 113.661903 |
| 23 | 500 | 0.000388 | 53 | 500000 | 28.576854 |
| 24 | 500 | 0.000285 | 54 | 500000 | 113.668454 |
| 25 | 500 | 0.000385 | 55 | 500000 | 28.591569 |
| 26 | 500 | 0.00038 | 56 | 500000 | 113.579596 |
| 27 | 500 | 0.000643 | 57 | 500000 | 113.65126 |
| 28 | 500 | 0.000447 | 58 | 500000 | 113.832439 |
| 29 | 500 | 0.00038 | 59 | 500000 | 28.587749 |
| 30 | 500 | 0.000466 | 60 | 500000 | 113.535567 |

Figure 1: Original Implementation Execution Times for Experiment

## Optimized Implementation

The Shadows problem adheres well to a triangular reusable memory structure like that of *Scan* from $CS320$. Stepping through the given array of heights, generated randomly here, the state of height $i$ depends on the state of height $i - 1$, and it will determine the state of height $i + 1$. Imagining the sub-problem as a portion of an already solved structure, the

*reduce* statement can be broken out into a linear algorithm similar to *Scan* that only needs to iterate on the order of $i$ rather than $i * j$. The optimized algorithm has a run time of $\mathbb{O}(N)$. Data supporting this is shown below.

The realization of a triangular reusable memory implementation in *AlphaZ* was provided by the $NormalizeReduction(prog)$ feature. This feature first replaced the original reduction statement with a reduction for the shifting of the triangle; however, this is still quadratic by nature of the reduction. The algorithm has the form of *Scan*, though now, and can be modified to reuse. The reduce is replaced by an inequality and binary maximum operation.

| | | |
|---|---|---|
| 1 | 5 | 6.00E-06 |
| 2 | 5 | 6.00E-06 |
| 3 | 5 | 6.00E-06 |
| 4 | 5 | 6.00E-06 |
| 5 | 5 | 6.00E-06 |
| 6 | 5 | 7.00E-06 |
| 7 | 5 | 6.00E-06 |
| 8 | 5 | 6.00E-06 |
| 9 | 5 | 6.00E-06 |
| 10 | 5 | 6.00E-06 |
| 11 | 50 | 1.00E-05 |
| 12 | 50 | 8.00E-06 |
| 13 | 50 | 7.00E-06 |
| 14 | 50 | 1.30E-05 |
| 15 | 50 | 6.00E-06 |
| 16 | 50 | 1.30E-05 |
| 17 | 50 | 7.00E-06 |
| 18 | 50 | 7.00E-06 |
| 19 | 50 | 7.00E-06 |
| 20 | 50 | 7.00E-06 |
| 21 | 500 | 2.20E-05 |
| 22 | 500 | 1.90E-05 |
| 23 | 500 | 2.00E-05 |
| 24 | 500 | 2.00E-05 |
| 25 | 500 | 2.70E-05 |
| 26 | 500 | 2.10E-05 |
| 27 | 500 | 2.00E-05 |
| 28 | 500 | 2.00E-05 |
| 29 | 500 | 2.00E-05 |
| 30 | 500 | 2.80E-05 |

| | | |
|---|---|---|
| 31 | 5000 | 0.000131 |
| 32 | 5000 | 0.000148 |
| 33 | 5000 | 0.000113 |
| 34 | 5000 | 0.000146 |
| 35 | 5000 | 0.000162 |
| 36 | 5000 | 0.000188 |
| 37 | 5000 | 0.000141 |
| 38 | 5000 | 0.000145 |
| 39 | 5000 | 0.000144 |
| 40 | 5000 | 0.000216 |
| 41 | 50000 | 0.002211 |
| 42 | 50000 | 0.002119 |
| 43 | 50000 | 0.001799 |
| 44 | 50000 | 0.001617 |
| 45 | 50000 | 0.001348 |
| 46 | 50000 | 0.001802 |
| 47 | 50000 | 0.001454 |
| 48 | 50000 | 0.001459 |
| 49 | 50000 | 0.001558 |
| 50 | 50000 | 0.002227 |
| 51 | 500000 | 0.006477 |
| 52 | 500000 | 0.006513 |
| 53 | 500000 | 0.010628 |
| 54 | 500000 | 0.005857 |
| 55 | 500000 | 0.007042 |
| 56 | 500000 | 0.007793 |
| 57 | 500000 | 0.008867 |
| 58 | 500000 | 0.007645 |
| 59 | 500000 | 0.008662 |
| 60 | 500000 | 0.005511 |

Figure 2: Optimized Implementation Execution Times for Experiment

**Execution Time Comparison Graph**

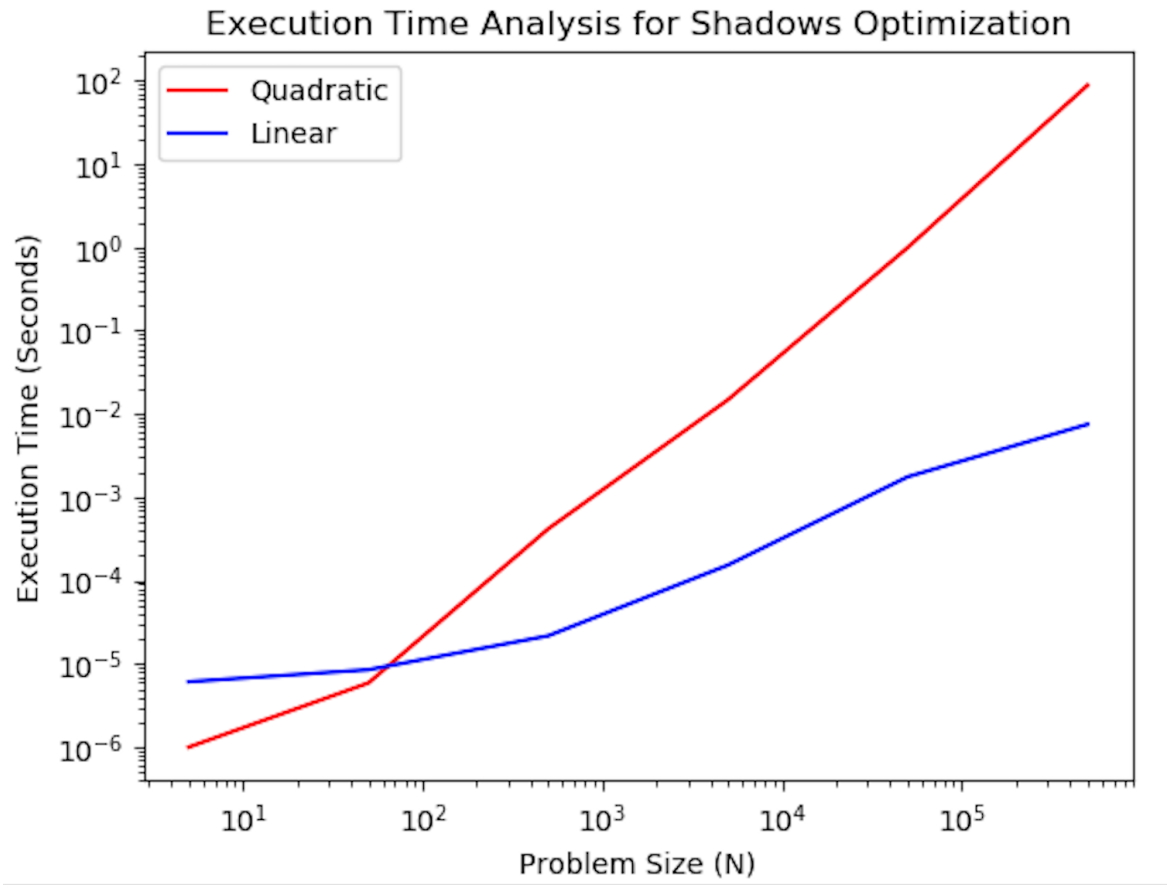


Figure 3: Logarithmic Scaled Graph Comparing Execution Time of Optimized and Non-Optimized Shadows Implementation

An analysis of the slope after problem size 500 reveals the order of the non-optimized version to be 1.8872 and the order of the optimized version to be 0.8446. These factors are significantly close to 2 and 1 - indicating quadratic and linear performance respectively.