David Tran
251169871

HW 1

CS 4402
February 27, 2024

1. Let A be a lower-triangular $n \times n$ matrix. The following computes and returns its inverse.

```
fn lower_triangular_inverse(row, col, n, A)
{
    A1_inv = spawn lower_triangular_inverse(row, col, n/2);
    A3_inv = lower_triangular_inverse(row + n/2, col + n/2, n/2);
    sync;
    A2 = A.submatrix(row + n/2, col, n/2);
    result = matrix with (A1_inv) starting at (0,0),
        (-A3_inv * A2 * A1_inv) starting at (n/2, 0),
        (A3_inv) starting at (n/2, n/2)
    return result;
}
```

2. The main algorithm is shown below.

```
#include "SquareMatrix.hpp"
#include "SquareMatrixView.hpp"

auto lower_triangular_inverse(SquareMatrixView const &matrix, size_t
    multithread_threshold = 2)
    -> SquareMatrix {
    auto const n = matrix.size();
    if (n <= multithread_threshold) {
        return matrix.inverse_forward_substitution();
    }

    auto const A1_inv = lower_triangular_inverse(
        matrix.submat(0, 0, n/2), multithread_threshold);
    auto const A2 = matrix.submat(n/2, 0, n/2);
    auto const A3_inv = lower_triangular_inverse(
        matrix.submat(n/2, n/2, n/2), multithread_threshold);
    auto const prod = -A3_inv * A2 * A1_inv;
    auto result = SquareMatrix {n};
    result.write_submat(0, 0, A1_inv);
    result.write_submat(n/2, n/2, A3_inv);
    result.write_submat(n/2, 0, prod);
    return result;
}
```

Note that `SquareMatrixView::submat` runs in constant time, matrix multiplication runs in $\Theta(n^3)$, and `Matrix::write_submat` runs in $\Theta(n)$. The recurrence is

$$T(n) = 2T(n/4) + c(n/4)^3$$