Stephen Belden

Meghan Haukaas

Chris Ruiz

**Project 4 Test Cases – Largest Submatrix of Ones**

1. A matrix of all 0’s

0 0 0

0 0 0

0 0 0

Expected Output: Largest Submatrix of Ones has area: 0

Actual Output: Largest Submatrix of Ones has area: 0

1. A matrix of all 1’s

1 1 1

1 1 1

1 1 1

Expected Output: Largest Submatrix of Ones has area: 9

Actual Output: Largest Submatrix of Ones has area: 9

1. A matrix with single 1 as the greatest 1’s rectangle

0 0 0

0 1 0

0 0 0

Expected Output: Largest Submatrix of Ones has area: 1

Actual Output: Largest Submatrix of Ones has area: 1

1. A matrix with a greatest 1’s square

0 1 1

0 1 1

0 0 0

Expected Output: Largest Submatrix of Ones has area: 4

Actual Output: Largest Submatrix of Ones has area: 4

1. A matrix with a greatest 1’s rectangle

0 1 1

0 1 1

0 1 1

Expected Output: Largest Submatrix of Ones has area: 6

Actual Output: Largest Submatrix of Ones has area: 6

1. A matrix with a single column as its greatest submatrix:

1 0 0

1 0 0

1 0 0

Expected Output: Largest Submatrix of Ones has area: 3

Actual Output: Largest Submatrix of Ones has area: 3

1. A matrix with a single row as its greatest submatrix:

1 1 1

0 0 0

0 0 0

Expected Output: Largest Submatrix of Ones has area: 3

Actual Output: Largest Submatrix of Ones has area: 3

1. A matrix with a single greatest 1’s rectangle

1 1 0

1 1 0

0 0 1

Expected Output: Largest Submatrix of Ones has area: 4

Actual Output: Largest Submatrix of Ones has area: 4

1. A matrix with two or more equal greatest 1’s rectangles

1 1 0 0

1 1 0 0

0 0 1 1

0 0 1 1

Expected Output: Largest Submatrix of Ones has area: 4

Actual Output: Largest Submatrix of Ones has area: 4

1. A matrix with two or more equal, overlapping greatest 1’s rectangles

1 1 0

1 1 1

1 1 1

Expected Output: Largest Submatrix of Ones has area: 6

Actual Output: Largest Submatrix of Ones has area: 6

**Discussion of Test Cases**

All test cases worked as expected, outputting the expected area of the largest submatrix of 1’s.

**Time Complexity Discussion**

Using a dynamic programming approach we were able to provide O(N) time complexity, where N is the number of elements in the matrix. [Time complexity is O(M2) if M is the length of one side of a square matrix.] For simplicity the rest of this document will use N as the number of elements in the matrix.

Our dynamic programming algorithm uses a histogram and a linear search through that histogram. The histogram records the number of contiguous 1’s above and including the current 1 at every position in the input matrix. Generating this histogram matrix requires a constant number of operations on every element in the input matrix, so the histogram generation is O(N). Next, we look at every value in the histogram to find the widest and tallest rectangular region of ones. Using a stack of pairs to keep track of important values, we only need to look at each value in the histogram once, so this search is O(N). This makes the entire algorithm O(N) + O(N), or just O(N).

The algorithm was tested with randomly and uniformly generated matrices with up to 16,777,216 elements and maintained time complexity and output accuracy.

**Time Complexity Measurement**

Analysis only includes the steps taken in findLargestSubmatrix() function. Matrix generation time is excluded.