Jeffrey Lansford

09-11-2019

Lab 3

Introduction:

For this lab, we further looking into using divide and conquer strategies except this time with matrix search problem. We again create two functions to show the differences of these algorithms. One is a brute force method and the other is divide and conquer. Then we plot it in R to show the results of the runtime of the algorithms.

Methods:

The brute force function is simple to do as it is just two nested loops. One goes through columns of the matrix and the other goes through rows. We declare the max to be the first element of the matrix, then within the column loop, we compare the max with an element of the matrix to see if we find a greater maximum.

```
Max = m(0,0)

For ( i = 0; i < m's number of rows; i++ ) {
        For ( j = 0; j < m's number of columns; j++) {

                If ( Max < m(i,j) {

                        Max = m(i,j)

                }

        }


}

Return max
```

The divivde and conquer has a bit more compleaxity to it has we have to find the middle row of the matrix, then we divide the matrix by two halves from the middle row and then we do recursion on both halves to get the maximum of each row.

//Base case

If (rows == 1)

    Max = m(0,0)

    For (I = 0; I < m's number of columns; i++)

        If max < m(I,0)

            Max = m(I,0)

    Return max

// Get middle row

MiddleRow = $\left\lceil \frac{m's\ number\ of\ rows}{2} \right\rceil$

If rows == 2 then middleRow = 2 // Do this because matrixs with rows of 2 cannot be have a middle row

middleMax = m(middleRow,0)

    For (I = 0; I < m's number of columns; i++)

        If middleMax < m(middleRow,0)

            middleMax = m(middleRow,0)

firstHalf = Created First Half of the matrix

secondHalf = Created the Second Half of the matrix

r = find_row_maxima(firstHalf)

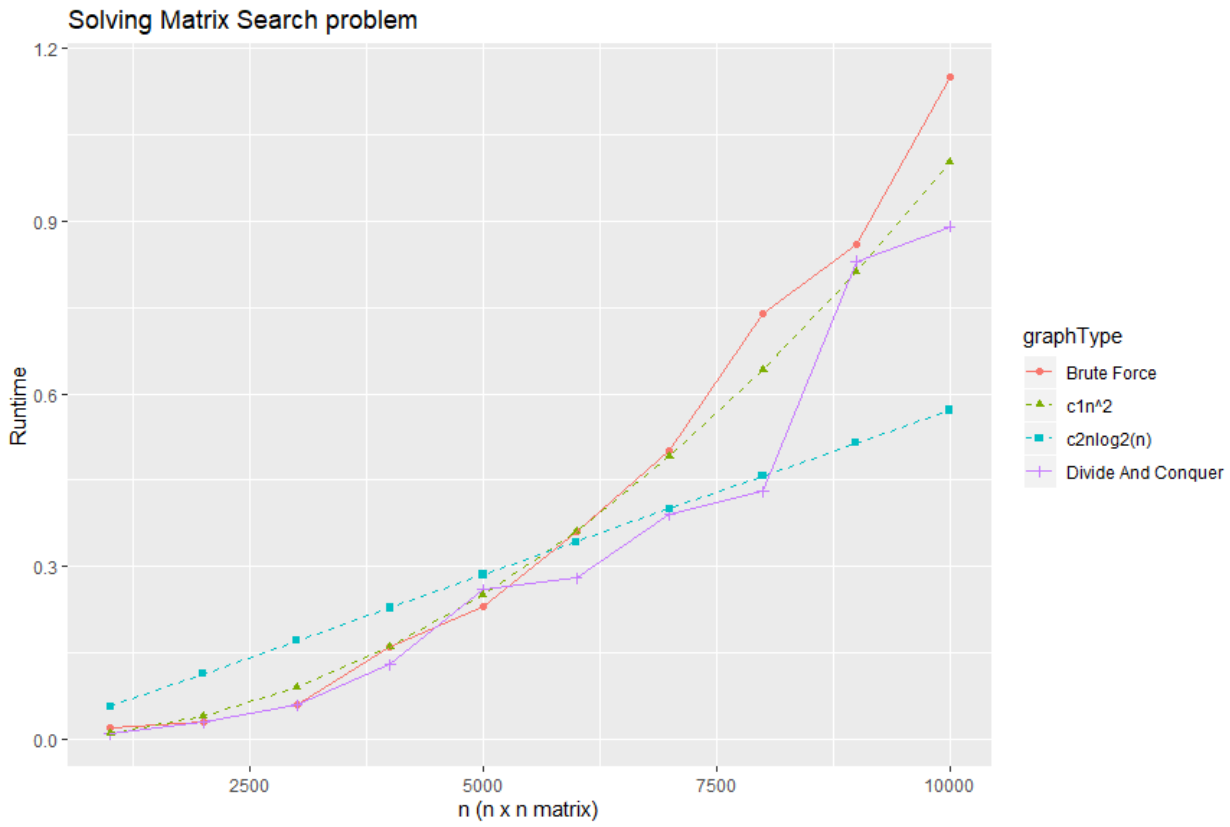
r + middleMax

if rows != 2:

    r + find_row_maxima(secondHalf)

return r

Results:



Here is the resulting graph on my machine an Intel i7-7700HQ @ 2.80GHz with 16GB of memory. We can see that the Brute force method is taking longer than divide and conquer method. The two theorical functions that we calculate in class are pretty close to their real-time functions. The $c_2 n \log_2 n$ could have been close if not for the spike towards the end. This may have occurred from my computer deciding to do something else or clock down for that moment. The c_1 is 1.001197e-08 and c_2 is 1.83119e-05.

Discussions:

This Lab was not as hard as the last Lab, but it had some of its own challenges. The brute force method was easy to do that I didn't even start with it first, I started with the divide and conquer as that was going to be difficult to do. I tried to think of the problem first instead of diving into the code and getting stuck multiple times. Most of my lab time was doing pseudocode on a sheet of paper, thinking how to divide the matrix correctly.

Conclusions:

This lab we learn more application of the divide and conquer algorithm at a different runtime than the last lab. We created both a brute-force method along with the divide and conquer to show the difference of the runtimes. We were shown a better divide and conquer problem with less subproblems in the program. The last lab had many sub-operations to it that bloated the runtime of the algorithm, this one did not have as many and gave a better representation of how the algorithm can reduce runtime. This lab gave us more experience using this algorithm and allowing us to use it more comfortably in the future.