IDAA432C Assignment - 6

By:

- Aditya Vallabh
- Kiran Velumuri
- Neil Syiemlieh
- Tauhid Alam

Question

You are given a 2D array such that elements occur at only certain indices while the remaining array is empty. Write a program to find the location or indices of the elements present in the array by recursively dividing the matrix into 4 parts(quad1, quad2, quad3, quad4) discarding at each step the quads that are clean or empty.

Algorithm Design

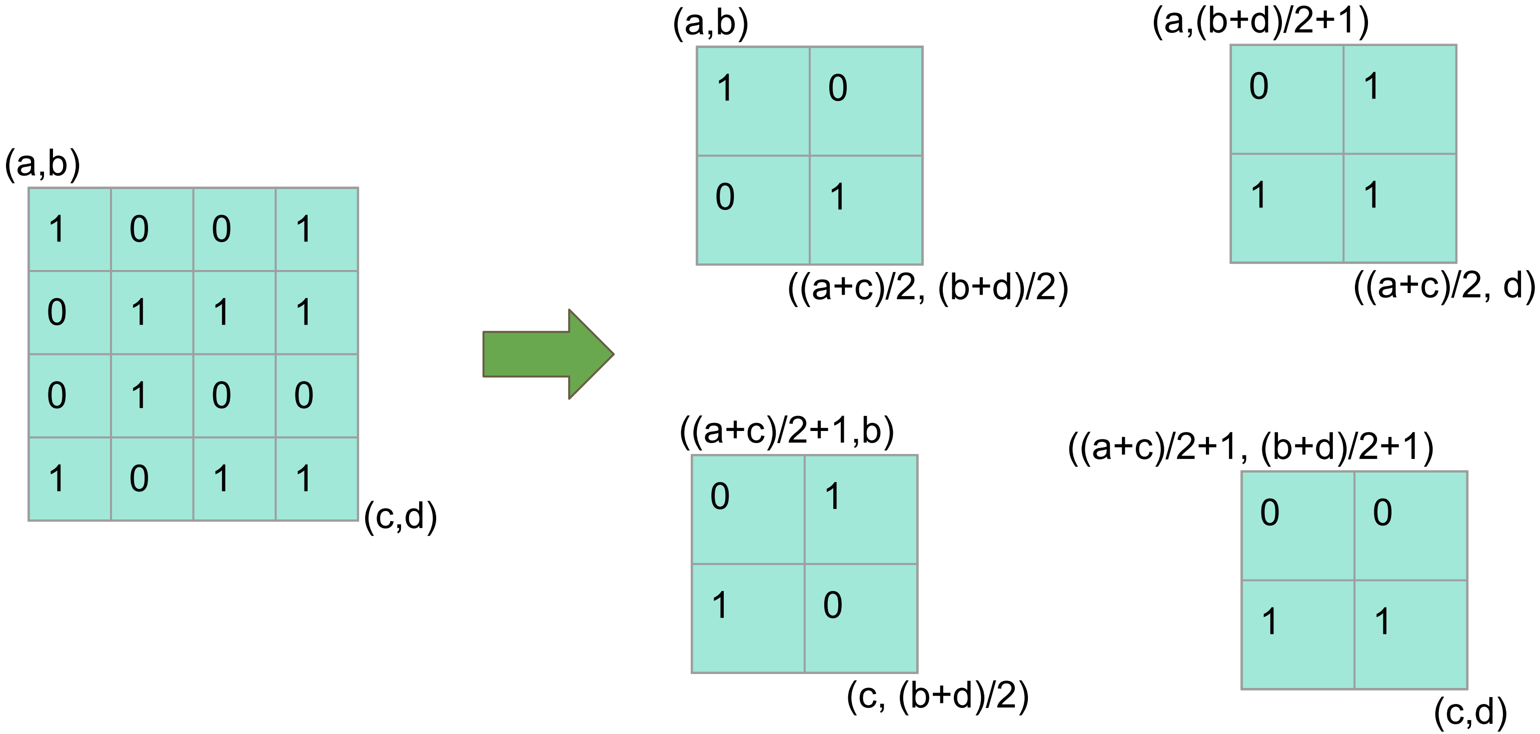
Quadrants of a matrix

Given an m x n matrix, the upper-left quadrant of the matrix would be a submatrix consisting of:

1) Rows: 0 to m/2

2) Columns: 0 to n/2

The other quadrants would be defined in the same fashion, with the appropriate start and end indices set.


(a,b)
1 0 0 1
0 1 1 1
0 1 0 0
1 0 1 1
(c,d)
(a,b)
1 0
0 1
((a+c)/2, (b+d)/2)
(a,(b+d)/2+1)
0 1
1 1
((a+c)/2, d)
((a+c)/2+1,b)
0 1
1 0
(c, (b+d)/2)
((a+c)/2+1, (b+d)/2+1)
0 0
1 1
(c,d)

Empty quadrants

- The cells of the matrix could either be filled or empty.
- In our implementation, a cell is considered filled if it has a value of 1 and is considered empty if it has a value of 0.
- The algorithm requires the ability to test if a quadrant is empty.
- An empty quadrant is one in which all cells are empty.
- For the algorithm in question, it is assumed that this test is of O(1) time.
- This assumption has to be made, or we'd be distracted from the analysis of the main algorithm.

The isEmpty Method

Algorithm 1 isEmpty Method

Input: $matrix, a, b, c, d$

for $i \leftarrow a\ to\ c$ **do**

  for $i \leftarrow b\ to\ d$ **do**

    if $matrix[i][j] \neq 0$ **then return** 0

return 1

Find algorithm

- Our algorithm has been implemented in the procedure called Find.
- The steps of the algorithm have been summarized in the steps below.
- Step 2 refers to the base case of the recursive function.

(a,b)

1	0	0	1
0	1	1	1
0	1	0	0
1	0	1	1

(c,d)

1) If input matrix has just one cell, go to step 2. Else, go to step 3.

2) If that cell is filled, print the cell's location and terminate. Else, just terminate.

3) Call Find on the upper-left quadrant

4) Call Find on the upper-right quadrant

5) Call Find on the lower-left quadrant

6) Call Find on the lower-right quadrant

Algorithm 2 find Method

Input: $matrix, m, n, a, b, c, d$

if $a \geq m$ or $c \geq m$ or $b \geq n$ or $d \geq n$ **then return**

if $a == c$ and $b == d$ **then**

  if $matrix[a][b] \neq 0$ **then**

    print(a,b)

  return

if $!isEmpty(matrix, a, b, c, d)$ **then**

  $find(matrix, m, n, a, b, (a + c)/2, (b + d)/2)$

  $find(matrix, m, n, a, (b + d)/2 + 1, (a + c)/2, d)$

  $find(matrix, m, n, (a + c)/2 + 1, b, c, (b + d)/2)$

  $find(matrix, m, n, (a + c)/2 + 1, (b + d)/2 + 1, c, d)$

Complexity Analysis

Time Complexity

The time taken by the algorithm in the worst case can be described by the following recurrence relation, where T(m, n) is the time taken to operate on an m x n matrix.

$$T(m, n) = 4\,T(\frac{m}{2}, \frac{n}{2}) + O(1)$$
$$T(1, 1) = O(1)$$

Here, the 4 T(m/2 , n/2) refers to the recurrent procedure on the four quadrants of the matrix, while O(1) is the complexity of the checking if the current sub-matrix is empty.

The base case refers to the operation on a 11 matrix and hence takes O(1) time.

We can solve the recurrence relation as follows:

$$
\begin{aligned}
T(m,n) &= 4\ T(\frac{m}{2}, \frac{n}{2}) + 1 \\
&= 4\ [4\ T(\frac{m}{4}, \frac{n}{4}) + 1] + 1 \\
&= 16\ T(\frac{m}{4}, \frac{n}{4}) + 4 + 1 \\
&= 16\ [4\ T(\frac{m}{8}, \frac{n}{8}) + 1] + 4 + 1 \\
&= 64\ T(\frac{m}{8}, \frac{n}{8}) + 16 + 4 + 1 \\
&\vdots \\
T(m,n) &= (2^k)^2\ T(\frac{m}{2^k}, \frac{n}{2^k}) + \sum_{i=1}^{k} 4^{i-1} \\
&= (2^k)(2^k)\ T(\frac{m}{2^k}, \frac{n}{2^k}) + \sum_{i=1}^{k} 4^{i-1}
\end{aligned}
$$

For $T(1,1)$:

$$\implies (\frac{m}{2^k}, \frac{n}{2^k}) = (1,1)$$
$$\implies m = 2^k \ and \ n = 2^k$$
$$\implies k = \log_2 n$$

Entering the value of 2^k in the equation above, we obtain the solution:

$$T(m,n) = mn + \sum_{i=1}^{\log_2 n} 4^{i-1}$$
$$= 2mn$$

Hence, the time complexity of the algorithm is O(m x n).

- The worst case occurs when all the cells of the matrix are filled, requiring a recursive call for every quadrant of every sub-matrix.

Worst Case

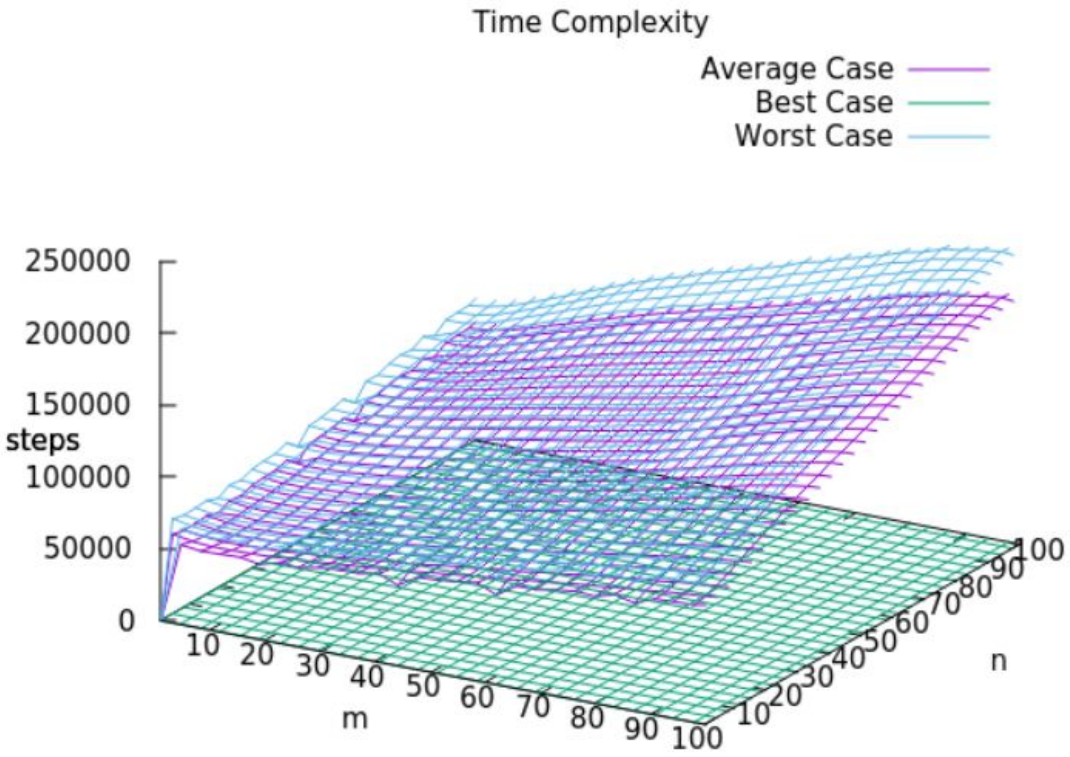
$$O(mn)$$

- The best case occurs when the entire matrix is empty. This would prevent any recursive calls in the algorithm, resulting in a time complexity of O(1).

Best Case

$$O(1)$$

Time Complexity



Worst Case:

$$O(mn)$$

Best Case:

$$O(1)$$

Conclusion

- Through this assignment we proposed the algorithm to find the location of non-zero elements in a 2D array by recursively dividing the matrix into 4 quadrants.
- In this process, we discarded those quadrants which are filled only with zeros.
- Moreover our algorithm has both time and space complexities of O(m n) where m, n are the dimensions of the matrix.


Thank you