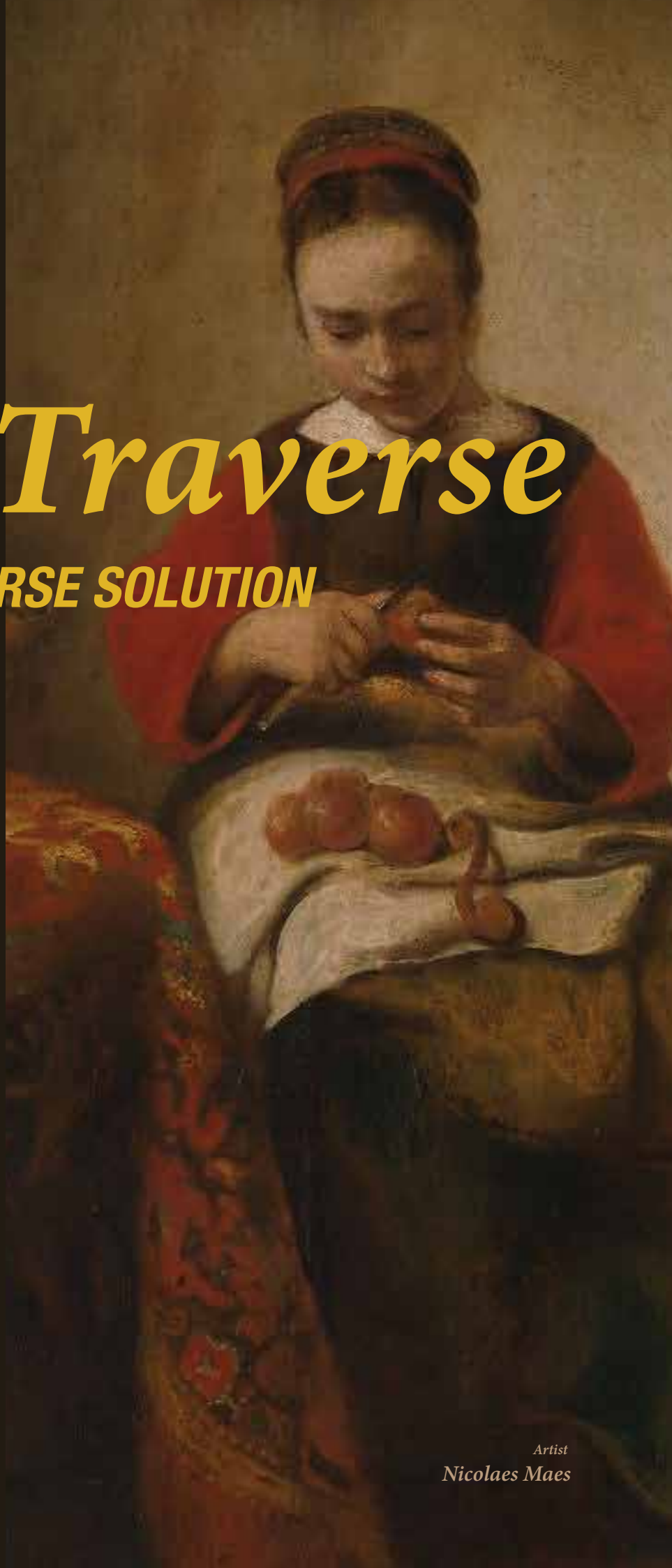
Peel Traverse

SPIRAL TRAVERSE SOLUTION

Tze Sien



FAKE PUBLICATION

Artist
Nicolaes Maes

Index

1.0 Concepts

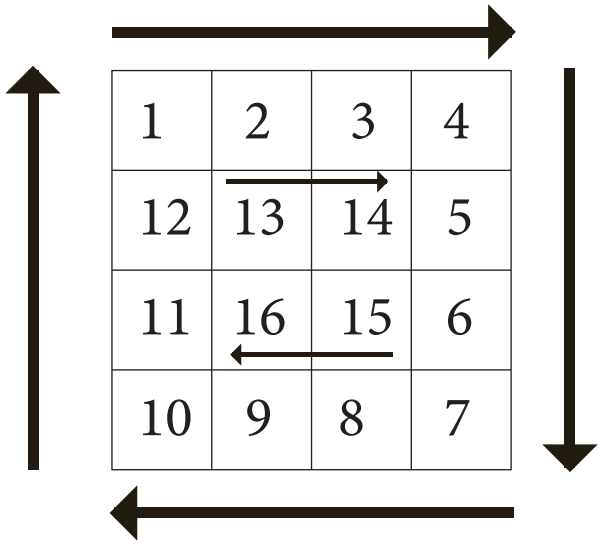
The data structure of our problem is in spiral shape, similar to onion, we have to peel it layer by layer, the only difference is the direction we peel it. For Onion, side peeling approach; Our data is in rectangular shape and spiral sequence, thus, we have to peel in "top-right-down-left-top-right-down.....".

Fig. 1.0 Onion Oil Canvas Art

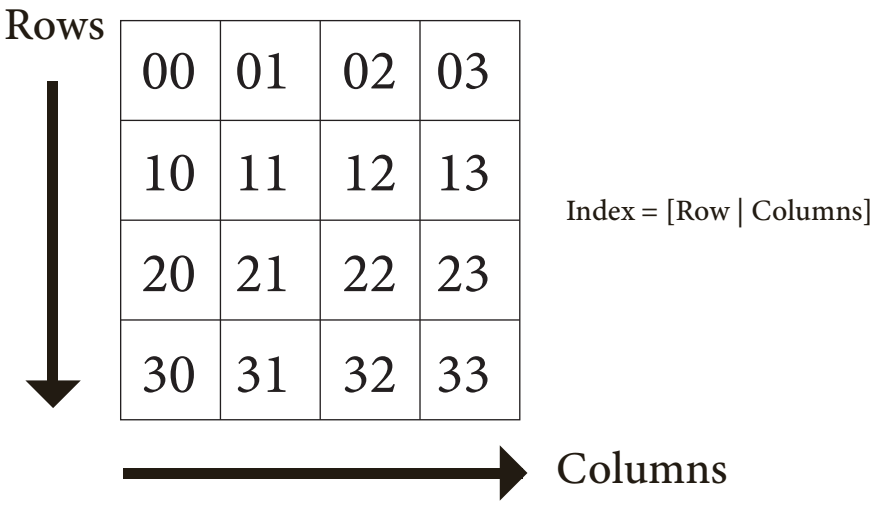
1	2	3	4
12	13	14	5
11	16	15	6
10	9	8	7

Fig. 1.1 Spiral Data Structure

Walk Through the Problem



1. Convert the matrix into numric indexes of the sequences.



2. Let's see how the index of spiral sequence will look like

00, 01, 02, 03, 13, 23, 33, 32, 31, 30,20,10,11,12,22,21

3. Split it into cycle and direction

00, 01, 02, 03, 13, 23, 33, 32, 31, 30,20,10,11,12,22,21

First Cycle
Top: 00, 01, 02, 03
Right: 13, 23
Bottom: 33, 32,31, 30
Left: 20,10

Second Cycle
Top: 11,12
Bottom: 22,21

Take a Cycle and Split it out

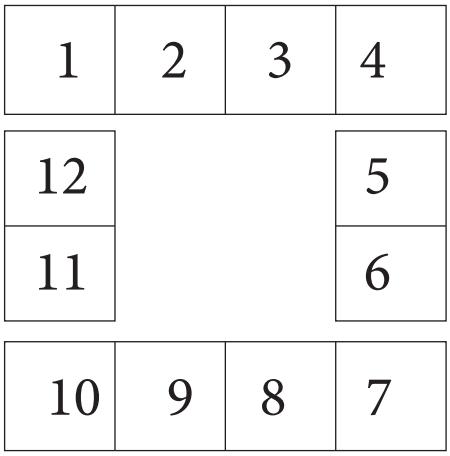
1	2	3	4
12			5
11			6
10	9	8	7

1	2	3	4
12			5
11			6
10	9	8	7

Do you notice they are in pairs?

Top - Bottom

First Row + Iteration of Column [0, 1, 2, 3]

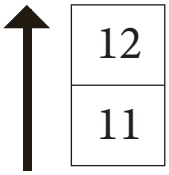


We are able use for loop once and get the data from top and bottom

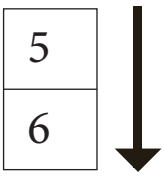
```
for (i=0; i <= width-1;i++){
    normal: i  //0,1,2,3
    reverse: width-i //3,2,1,0
}
```

Last Row +
Reverse Iteration of Column [3, 2, 1, 0]

Right - Left



First Column +
Reverse Iteration of
Update Array's Row
[1, 0]



Last Column +
Iteration of
Update Array's Row
[0, 1]

```
for (i=0; i <= height-1;i++){
    normal: i  //0,1
    reverse: height-i //1,0
}
```

The solution can run in pairs!

1. Top - Bottom
2. Remove Top-Bottom from Array
3. Right - Left
4. Remove Right and Left from Array

4. Formulate it

Declatations
Width of Array = W - 1
Height of Array = H - 1
iterate(W) = 0,1,2,3....
reverse(iterate(W) = ...,3,2,1,0

Formulae
Top: 00, 01, 02, 03 [0][iterate(W)]
Bottom: 33, 32,31, 30 [H][reverse(iterate(W)]

Right: 13, 23 [iterate(H)][W]
Left: 20,10 [reverse(iterate(H)][0]

5. Cycle Sequence

```
array = [
      [1,   2, 3, 4],
      [12,13,14,5],
      [11,16,15,6],
      [10, 9, 8,  7]
   ]
```

While (array.length != 0)

1.	top = array[0][iterate(W)]
	bottom = array[H][reverse(iterate(W)]
2.	array = remove top and bottom from array
3.	right = array[iterate(H)][W]
	left = array[reverse(iterate(H)][0]
4.	array = remove right and left from array
5.	result array = top + right + bottom + left

2.0 Implementations

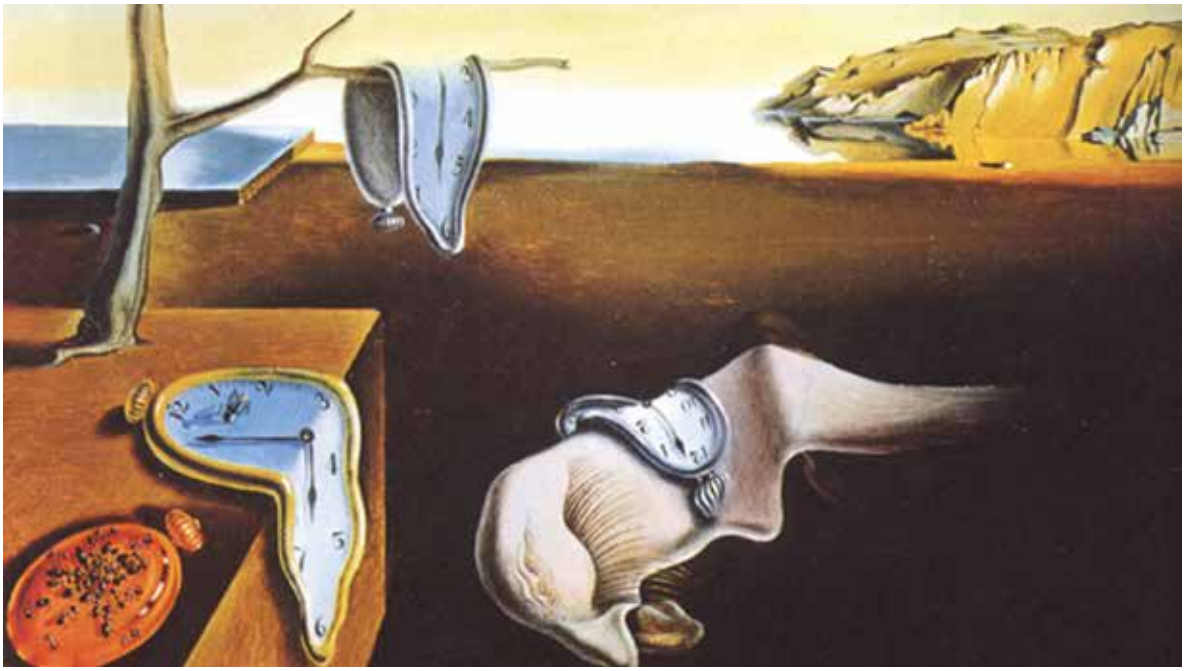
Blur? Let's code it out!

```
function spiralTraverse(array){

    arrangedArray = []
    end = false;

    while (!end){

        a = []
        b = []
        c = []
        d = []

        // 1.0 Top-Bottom Pair

        // 1.1 Get Top Array
        a = array[0]

        // 1.2 Get Bottom Array
        for(let i = 0; i <= array[0].length-1; i++){
            if(array.length > 1){
                b.push(array[array.length - 1][array[0].length-1-i])
            }
        }

        // 2.0 Remove Top and Bottom from Array
        array = array.slice(1, -1)



        // Check if the array is empty
        if(array.length != 0) {
            //3.0 Right-Left Pair

            for(let h = 0; h <=array.length-1; h++)
            {
                // 3.1 Get Right Array
                c.push(array[h][array[0].length-1])

                // 3.2 Get Left Array
                d.push(array[array.length -1 - h][0])

                // 4.0 Remove Right Element
                array[h].pop()
                // steps++;
            }

            // 4.0 Remove Left Element
            for(let h = 0; h <=array.length-1; h++)
            {
                array[h].shift()
            }

        }else{
            end = true
        }

        // Combine into an sorted array
        arrangedArray = arrangedArray.concat([...a , ...c , ...b , ...d])
    }

    return arrangedArray;
}
```

3.0 Time-Space Complexity

How good is the algorithm?

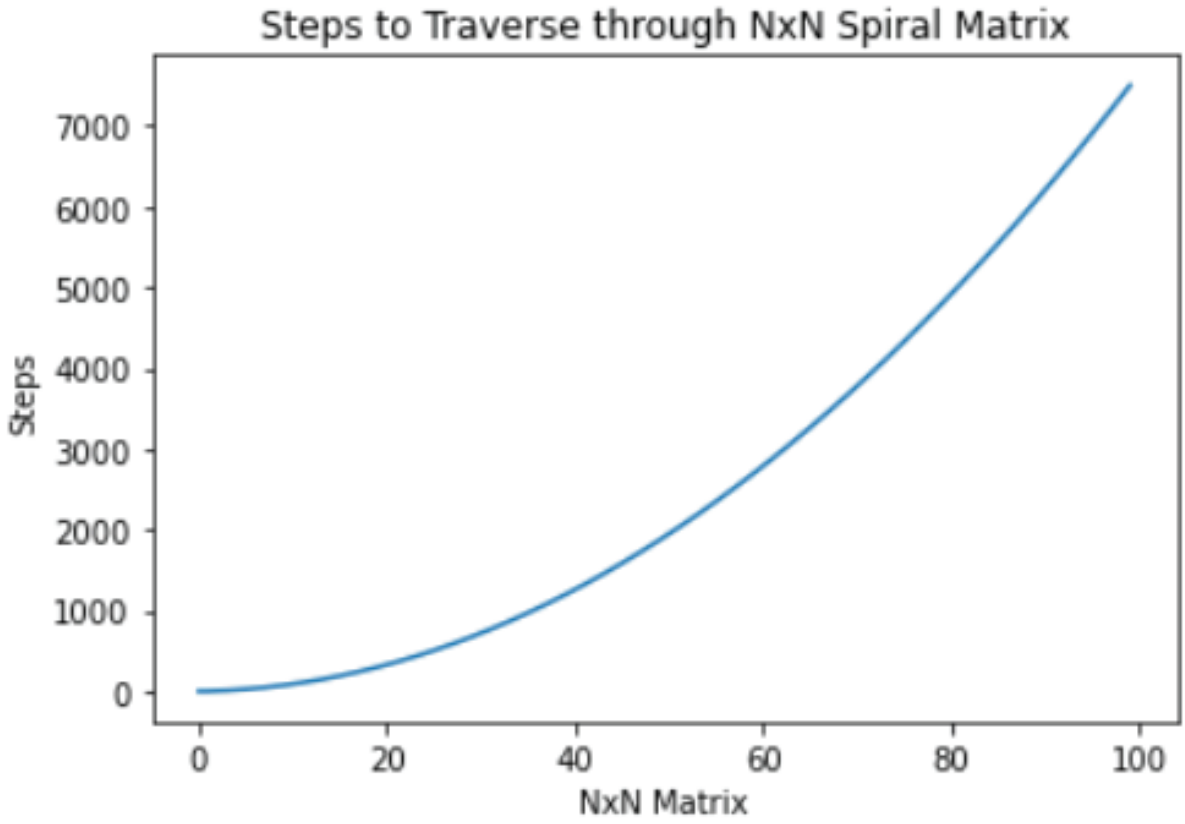
The excellent algorithm have low time space complexity.
Let's check with this Peel Traverse Algorithm!



Variable Used

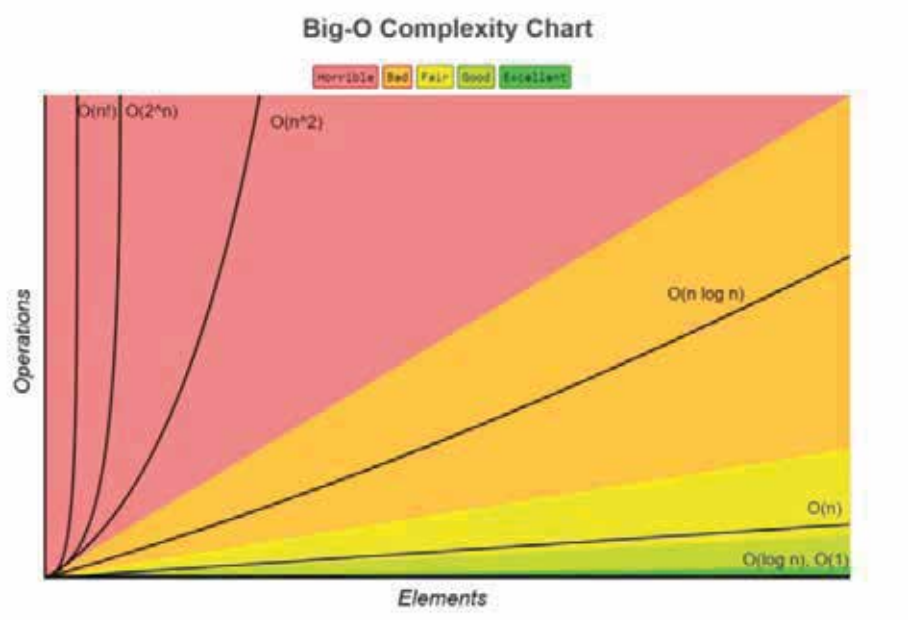
array	Decrease at most 2 rows & columns /cycle [array]
arrangedArray	A combined array pushed/cycle [array]
end	Only get rewite in the last cycle [bool]
a	Rewrite once/ cycle [array]
b	Rewrite once/ cycle [array]
c	Rewrite once/ cycle [array]
d	Rewrite once/ cycle [array]

The steps count for N x N Matrix (N = 0 to 100)

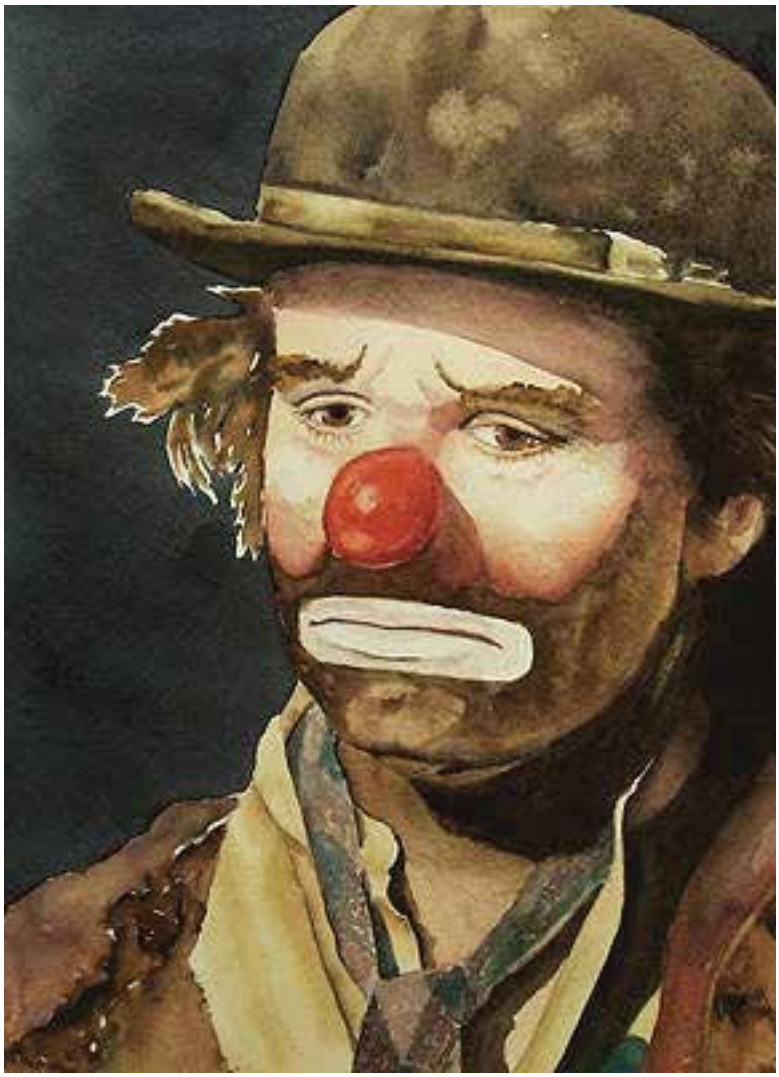


Exponential Time

Time-Space Complexity = $O(n^2)$



https://www.bigocheatsheet.com/



A Sad Conclusion

According to the graph of Big-O Complexity, the peel traverse algorithm falls in the HORRIBLE category, that's so sad. That's true horrible, because when I try to run with 1000 x 1000 matrix, it get result after 4-5 seconds, that is even longer when I try to use 10000 x 10000 matrix.

Thanks for Reading