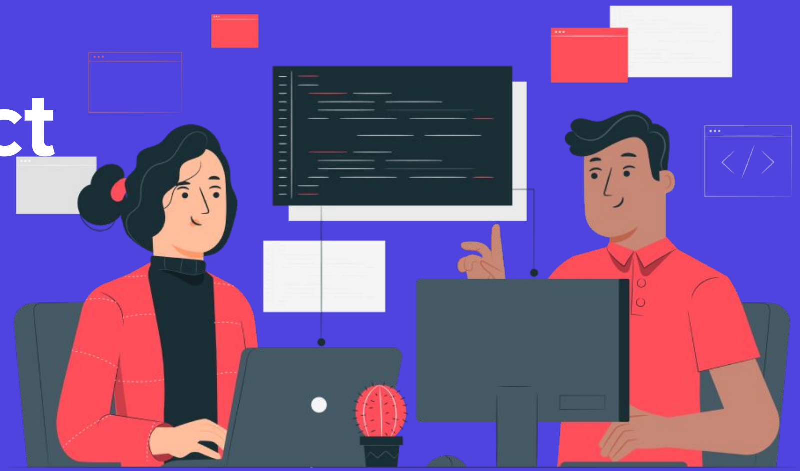
Arrays / Map / Object - Part 1



Relevel
by Unacademy

List of Problems Involved

- Egg Problem
- Ticket Queue Problem
- Chocolates Distribution Problem
- Awesome Arrays

Egg Problem

Problem Statement – Consider a tray of eggs which contains eggs in different placeholders. Your task is to find all the empty placeholders in the given tray.

Input Format -
A 2D array of M*N size will be given where each cell can contain either 0 or 1.
0 - represents empty cell
1 - represents egg present

Output Format -
Number of empty cells in given tray
NOTE - Each row and column is sorted in the given array

Input-
```
[[0, 0, 1],
[0, 1, 1],
[0, 1, 1]]
```

Output -
4

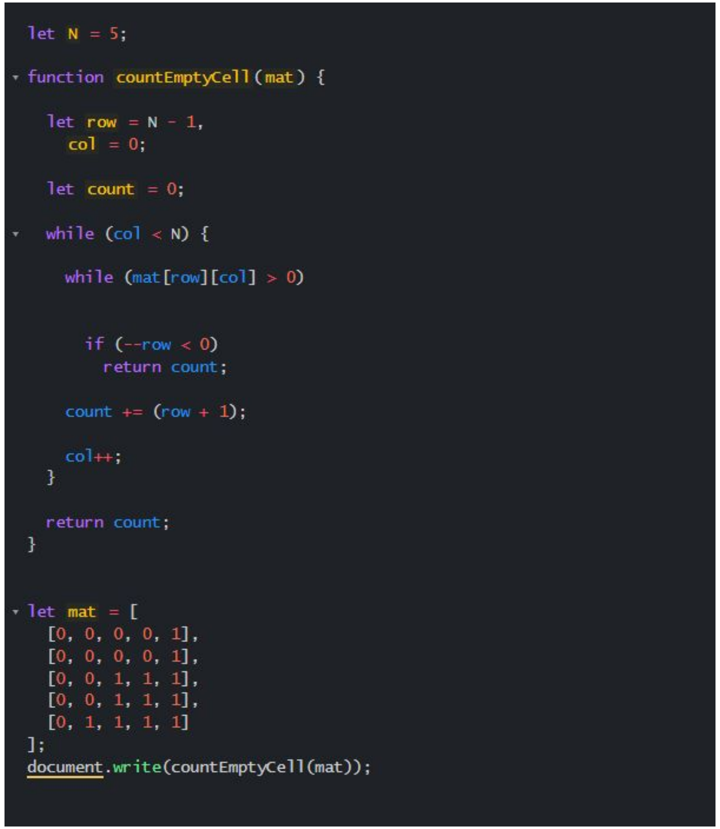
Approach – If we observe the problem, we need to count the number of zeroes present in the given input array.

Let's see each step –

1. Initialize a pointer from bottom left corner of given matrix
2. Initialize count = 0
3. Decrement row pointer until we get 0
4. Add current row index + 1 to count
5. Increment column pointer
6. Repeat step 3, 4 and 5 for all the columns

Code Link - https://jsfiddle.net/g9fpkx2d/

Ideone Link - https://www.ideone.com/swljhQ



```
let N = 5;

function countEmptyCell(mat) {

  let row = N - 1,
    col = 0;

  let count = 0;

  while (col < N) {

    while (mat[row][col] > 0)


      if (--row < 0)
        return count;

    count += (row + 1);

    col++;
  }

  return count;
}


let mat = [
  [0, 0, 0, 0, 1],
  [0, 0, 0, 0, 1],
  [0, 0, 1, 1, 1],
  [0, 0, 1, 1, 1],
  [0, 1, 1, 1, 1]
];
document.write(countEmptyCell(mat));
```

Ticket Queue Problem

Problem Statement – There are multiple train ticket counters and there are long queues in front of each counter. Tickets are given to each person in the queue. Each ticket will contain some number. Your task is to find the sum of numbers on the ticket in each queue.

Input Format -
A 2D array of M*N size will be given (where N = number of ticket counters and each column will represent a queue on the counters)

Output Format -
An array having sum of each queue

Input:

```
10, 20, 30, 40
15, 25, 35, 45
24, 29, 37, 48
32, 33, 39, 50
```

Output -
81, 107, 141, 228

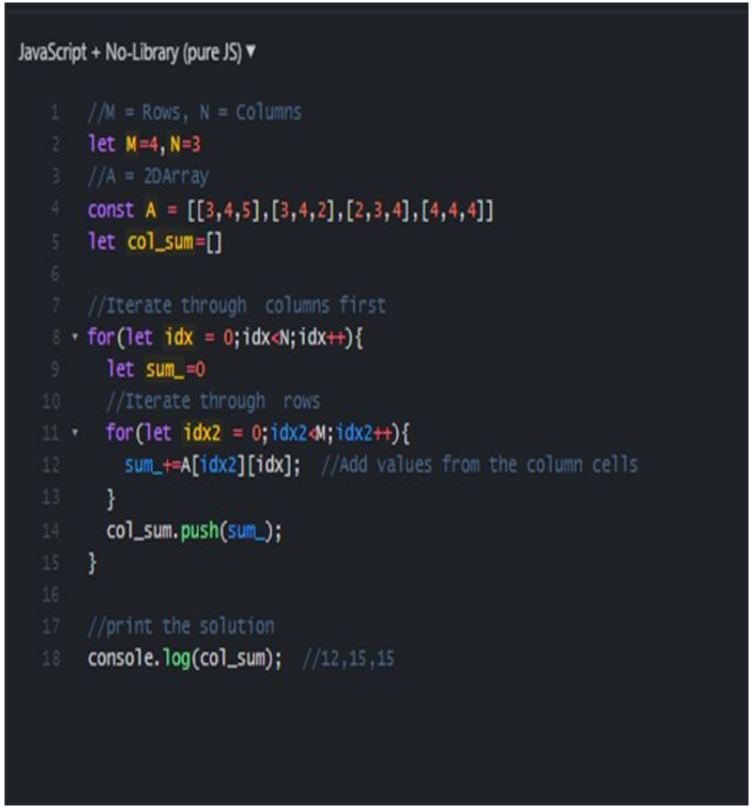
Approach – If we observe the problem, we need to find the sum of each column respectively.

Let's see each step –

1. Initialize an empty outputArray
2. Iterate over the column first
3. Iterate over the row
4. Calculate sum of column values and store it in sum
5. Save sum in outputArray

Code Link - https://jsfiddle.net/2mgetfps/

Ideone Link - https://www.ideone.com/Cnlrtv



```javascript
//M = Rows, N = Columns
let M=4,N=3
//A = 2DArray
const A = [[3,4,5],[3,4,2],[2,3,4],[4,4,4]]
let col_sum=[]

//Iterate through  columns first
for(let idx = 0;idx<N;idx++){
  let sum_=0
  //Iterate through  rows
  for(let idx2 = 0;idx2<M;idx2++){
    sum_+=A[idx2][idx];  //Add values from the column cells
  }
  col_sum.push(sum_);
}

//print the solution
console.log(col_sum);  //12,15,15
```

Chocolates Distribution Problem

Problem Statement – In a class, students are sitting on chairs which are arranged in the form a matrix i.e. 2D array. Chocolates have been distributed to them such that in each row and column, the number of chocolates are in increasing order.
You will be given a matrix which is a sitting arrangement of students having a number of chocolates and a number K.
You need to find the Kth smallest number of chocolates which a student can have.

Input- k = 4

```
10, 20, 30, 40
15, 25, 35, 45
24, 29, 37, 48
32, 33, 39, 50
```

Output - 24

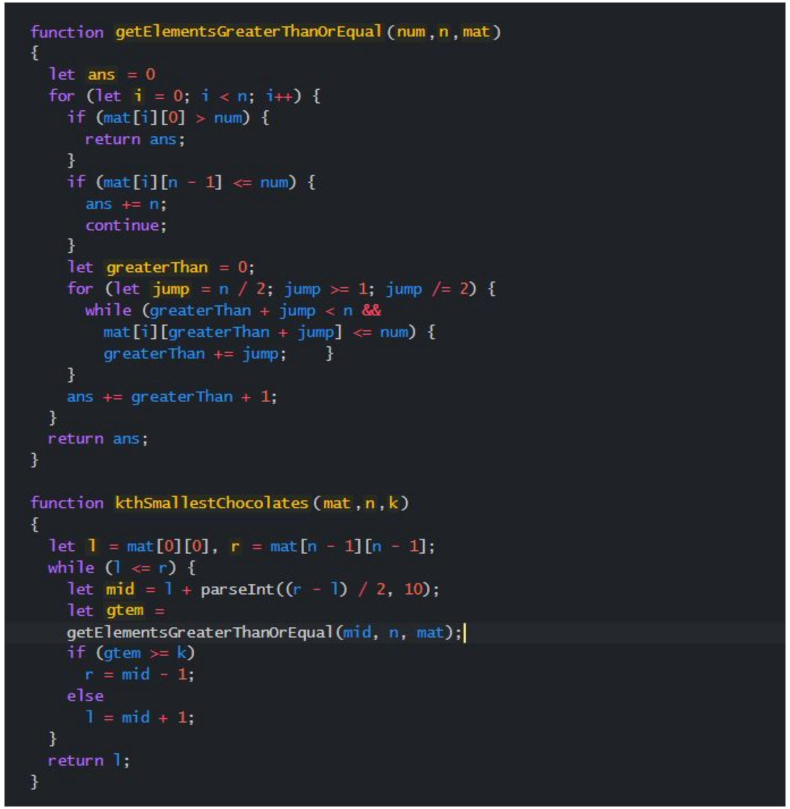
Approach – Since a given matrix is sorted row-wise and column-wise, we can use this pattern to find the solution. Our task is to search Kth smallest number of chocolates in a given matrix. We can use the Binary Search algorithm to solve this problem.

Let's see each step –

1. We will define range as first and last student which is mat[0][0] and mat[n-1][n-1]
2. Calculate the number of chocolates which a middle student can have and store it in mid.
3. Find the number of chocolates which is greater than or equal to mid and store it in gtem.
4. If gtem >= k -> update r = mid - 1
5. Else l = mid + 1
6. Repeat steps 2 to 5 till l <= r
7. Print l as output

Code Link - https://jsfiddle.net/3ho974mw/

Ideone Link - https://www.ideone.com/90E4Tt



```
function getElementsGreaterThanOrEqual(num,n,mat)
{
  let ans = 0
  for (let i = 0; i < n; i++) {
    if (mat[i][0] > num) {
      return ans;
    }
    if (mat[i][n - 1] <= num) {
      ans += n;
      continue;
    }
    let greaterThan = 0;
    for (let jump = n / 2; jump >= 1; jump /= 2) {
      while (greaterThan + jump < n &&
        mat[i][greaterThan + jump] <= num) {
        greaterThan += jump;    }
    }
    ans += greaterThan + 1;
  }
  return ans;
}

function kthSmallestChocolates(mat,n,k)
{
  let l = mat[0][0], r = mat[n - 1][n - 1];
  while (l <= r) {
    let mid = l + parseInt((r - l) / 2, 10);
    let gtem =
    getElementsGreaterThanOrEqual(mid, n, mat);
    if (gtem >= k)
      r = mid - 1;
    else
      l = mid + 1;
  }
  return l;
}
```

Awesome Arrays

Awesome Arrays – An array is known as an awesome array if for every pair of the adjacent numbers, the sum of numbers is a perfect square.

Awesome Arrays Problem - We need to find a number of permutations of the input array which are awesome arrays.

Example -
Input - [1,17,8]
Output - 2
[1,8,17] and [17,8,1] are 2 awesome arrays

Intuition – Since we need to check all adjacent pair sums when we are at the i position, we need to check for the i-1 position and i+1 position number.

Steps -

1. Create a temporary and boolean array that will keep track of visited nodes.
2. Sort the input array.
3. Create a recursive function that will take a sorted input array, a temporary array, and a boolean array
4. If temporary array size is equal to input array, increment the output and return
5. Iterate through the input array
6. If the current number and previous number are equal and the previous number is not visited, then continue
7. Check if the sum is a perfect square; if not, continue.
8. If the current number has already been visited, continue
9. Add the number to the temporary array and mark the visited array as true
10. Call recursive function again to check

Code Link - https://jsfiddle.net/6w9mjo43/1/

Ideone Link - https://www.ideone.com/8AWxTk

Time Complexity –
Time complexity will be O(N^N)

Space Complexity –
Space complexity will be O(N)

Thank You!