3)

In this problem, you will investigate a more efficient way to implement spatial filtering when all the filter coefficients have the same value. The motivation comes from the observation that as you slide the filter one pixel at a time over the image and compute the sum-of-products of image and filter values, you can use the results from the previous computation in the current computation.

Although the method can be generalized, we will consider the case in which all the filter coefficients have the value 1. And, we will also ignore the 1/n2 scaling factor that typically accompanies an averaging filter of size nxn.

(a) Describe the algorithm you would use to compute the output value at location (x,y) given that you have already computed the result for location (x-1,y), for example, for an averaging filter of size nxn (think about what changes when you shift the filter by one pixel).

Algorithm of efficient method

```
//Let the output value at (x-1,y) coordinate be prev_sum;
Int new_sum; // This will store the output value at (x,y) coordinate
Int image [N][M];
Int Filter.[n][n];

/*The origin is at top-left(0,0) and our x-axis increasing from top-left to top-right and y-axis is increasing from top-left to bottom-left*/

Int sum_of_product_of_previous_col=0;

For (int i = y; I>=(y-n-1); i - -) {
sum_of_product_of_previous_col+=images[x-n-1][1]
Filter [1][n-(y-i)];
}

//Filtering values for (x)th column
int sum_of_product_of_new_col=0;

For (int I=y; I>=(y-n+1); i - -){

sum_of_products_of_new_col += image [x][I]
}

new_sum = prev_sum - sum_of_product_of_previous_col +
sum_of_product_of_new_col
```

(b) How many additions (in terms of n) does this require for each output pixel. Count subtractions as additions.

As one can see from the above solution that we used plus n times for 2 times and one minus and one plus for final solution.

total_number_of _plus = n + n + 2 = 2 * n + 2

(c) Now, let's compare this with the standard approach of not using previous results. How many additions are required for each output pixel in this case. This should be in terms of n.

We see that the computation is done using a standard approach that will make use many more numbers of plus operators. We know that our filter matrix is the size of n*n, thes means that there are n^2 cells in the matrix so hence we will be doing n^2 cells of filter multiplication with n^2 cells of image and the result from these cells will be n^2 and those terms will be added.

Total_plus_operator = n^2

(d) Compute the computational advantage of the more efficient approach. This is simply the ratio of the number of additions required by the standard approach to the number of additions required by the more efficient approach. Again, this should be in terms of n.

We have to calculate the computational advantage which is equal to the ratio of numbers of plus operators that required by the standard approach and number of plus operators required by the efficient approach.

Computational advantage = n^2 / (2 * n + 2)