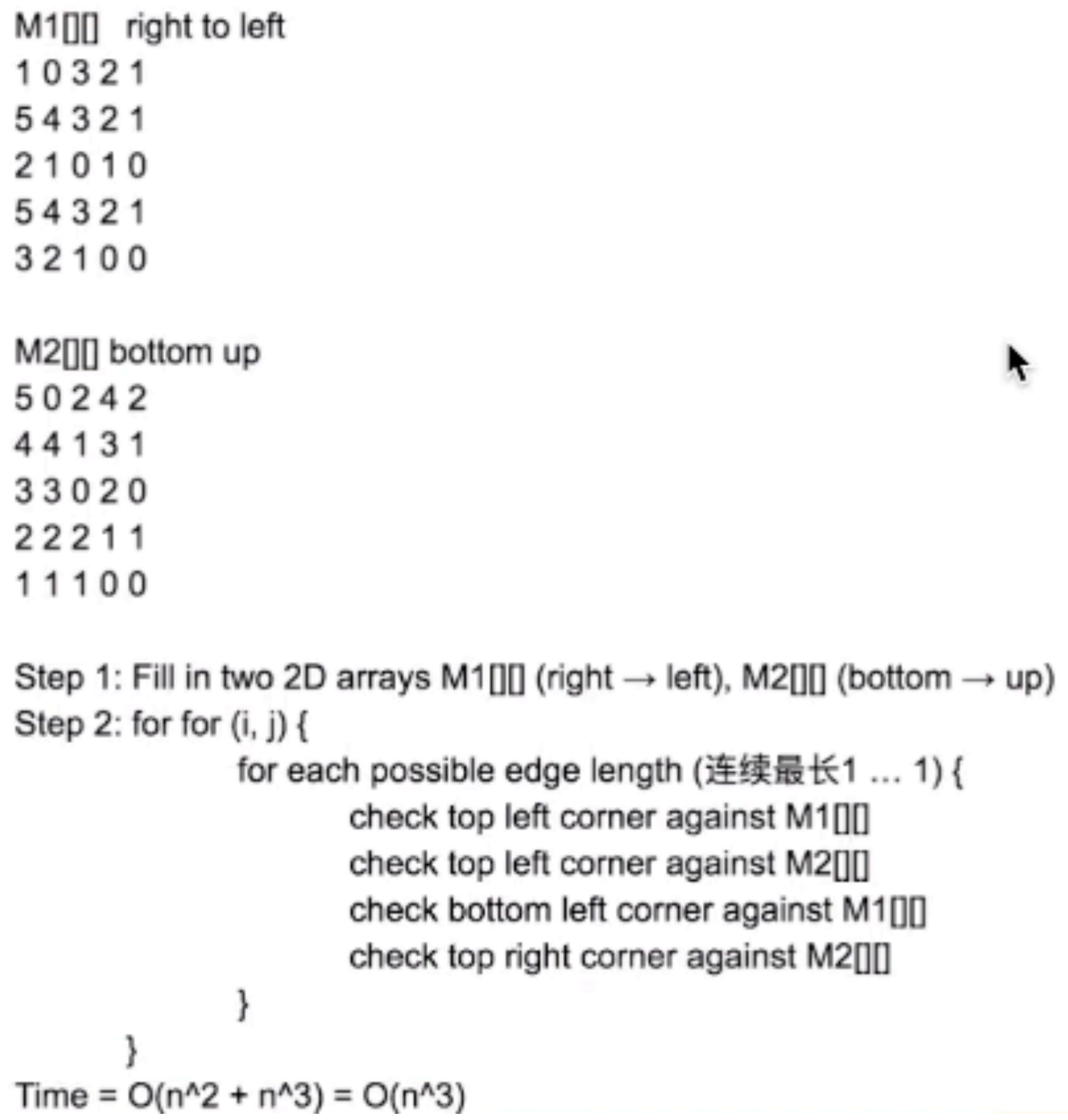
Q3a Given a matrix where every element is either ‘0’ or ‘1’, find the largest subsquare surrounded by ‘1’.
Examples:

Given a matrix where every element is either ‘0’ or ‘1’, find the largest subsquare surrounded by ‘1’.   Examples:
Input: mat[N][N] =

```
{ {'1', '0',  '1', '1', '1'},
  {'1', '1', '1',  '1', '1'},
  {'1', '1', '0',  '1', '0'},
  {'1', '1', '1',  '1,' '1'},
  {'1', '1', '1',  '0', '0'},
 };
```

Question 3a-0: how many squares are there in the matrix?
Answer: O(n^3)



M1[][]   right to left

1 0 3 2 1

5 4 3 2 1

2 1 0 1 0

5 4 3 2 1

3 2 1 0 0

M2[][] bottom up

5 0 2 4 2

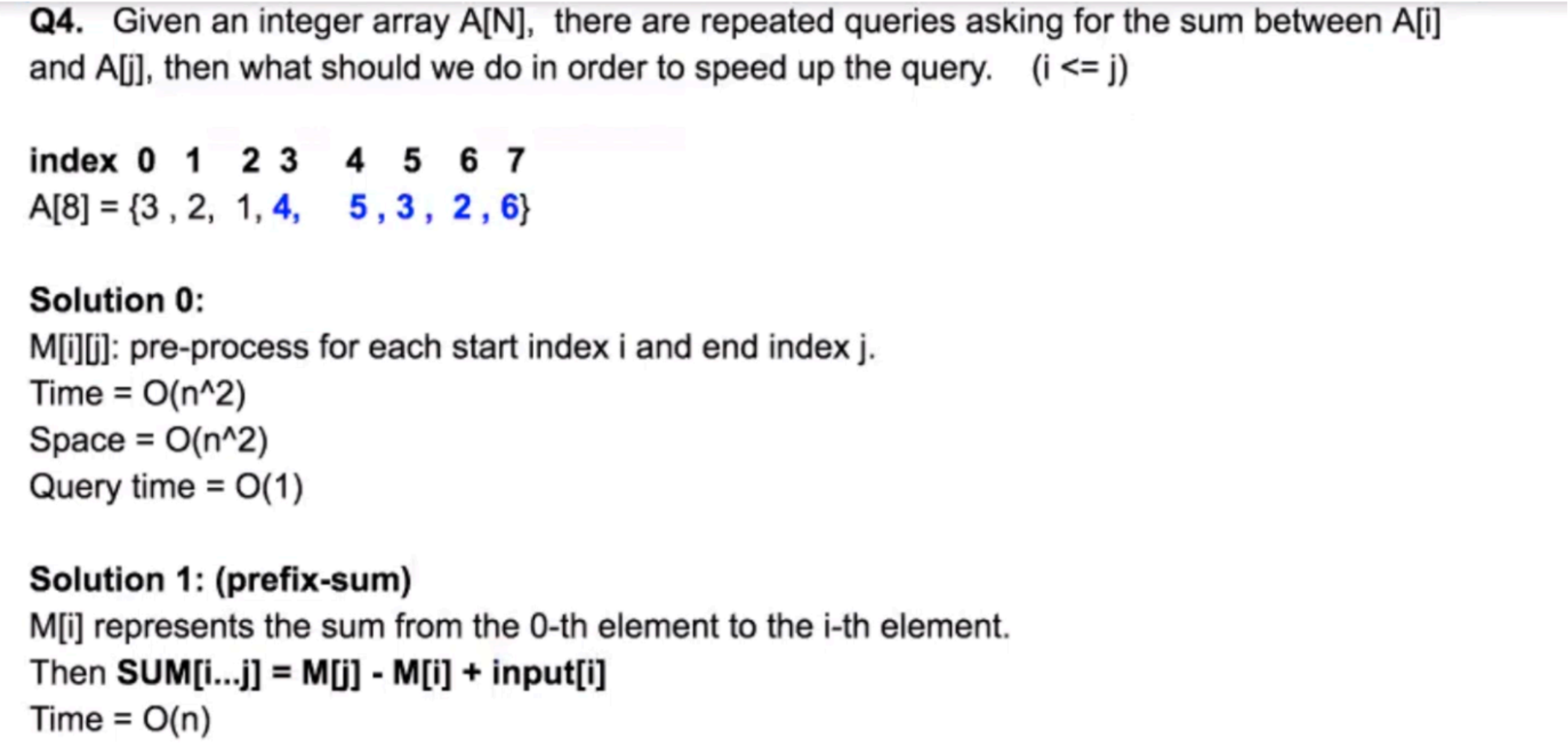
4 4 1 3 1

3 3 0 2 0

2 2 2 1 1

1 1 1 0 0

```
Step 1: Fill in two 2D arrays M1[][] (right → left), M2[][] (bottom → up)
Step 2: for for (i, j) {
                for each possible edge length (连续最长1 … 1) {
                        check top left corner against M1[][]
                        check top left corner against M2[][]
                        check bottom left corner against M1[][]
                        check top right corner against M2[][]
                }
        }
Time = O(n^2 + n^3) = O(n^3)
```



Q4. Given an integer array A[N], there are repeated queries asking for the sum between A[i] and A[j], then what should we do in order to speed up the query. (i <= j)

index 0 1 2 3 4 5 6 7
A[8] = {3 , 2, 1, **4, 5 , 3 , 2 , 6**}

Solution 0:
M[i][j]: pre-process for each start index i and end index j.
Time = O(n^2)
Space = O(n^2)
Query time = O(1)

Solution 1: (prefix-sum)
M[i] represents the sum from the 0-th element to the i-th element.
Then **SUM[i...j] = M[j] - M[i] + input[i]**
Time = O(n)

Question 5. Given a Matrix of integers (positive & negative numbers & 0s), how to find the submatrix with the **largest** sum?

input:

x x x x x x x x x

x x x x x x x x x

x x x **X x Z** x x x

x x x **W x Y** x x x

x x x x x x x x x

Solution 0:

Solution 0: (non-DP)

```
        There are O(n^4) 2D sub-matrix
        for for for for {
                We can use O(n^2) time to calculate the sum of the submatrix.
        }
Time = O(n^4 * n^2) = O(n^6)
```

Solution 1: (DP1: prefix-sum in 1D)

x x x x x x x x x
x x x x x x x x x
1 2 3 1 1 1 x x x
2 1 1 2 2 2 x x x
x x x x x x x x x

prefix sum
1 3 6 7 8 9 … 9 - 7 + 1 = 3
2 3 4 6 8 10 … 10 - 6 + 2 = 6
sum = 3 + 6 = 9

```
Step 1: pre-process the input matrix by using the prefix-sum trick
                M[i][j]         O(n^2)
Step 2: for for for for {
                Use M[i][j] to calculate the sum of the submatrix       O(n)
        }
Total time = O(n^4 * n) = O(n^5)
```

```
(0,0)
x x x x x x x x x           red = [0][0] -- [k-1][j]
x x x x x x x x x           green = [0][0] -- [i][t-1]
      (k,t)                 pink = [0][0] -- [k-1][t-1]
x x x X x Z x x x
x x x W x Y x x x
          (i,j)
x x x x x x x x x
```

SUM = PrefixSum[00][ij] - (red + green - pink)
= PrefixSum[00][ij] - PrefixSum[00][k-1, j] - PrefixSum[00][i, t-1] + PrefixSum[00][k-1, t-1]

input[n][m] =

```
1  2 3 4 5
-1 1 2 1 2
3  1 2 1 3
```

PrefixSum[][] =

```
1 3 6 10 15
0 3 8 13 20          sum_so_for_this_row = 5
                     for each M[i][j] = M[i-1][j] + sum_so_for_this_row
```

Solution:

```
for for for for {
        Use O(1) to calculate the SUM.
}
```

Time = O(n^4)

Solution 3: (DP3)

Question 0 (most popular DP question) **Largest sum of a subarray**

```
x x x x x x x x x
x x x x x x x x x
x   x    x X x Z x x x
x   x    x x x x x x x
x   x    x W x Y x x x
S0 S1  S2  ....   S5   S19     ⇒ new input for Question 0
x x x x x x x x x
```

```
for top {
        for bottom {
                Call Question 0                O(n)
```

```
input =
1 2 3 4 5
2 1 1 1 1
2 3 1 1 2

Column-wise PrefixSum[i][j] =
1 2 3 4 5
3 3 4 5 6
5 6 5 6 8

S[] = 4 4 2 2 3        ⇒ Question 0    (left = 0, right = 4)
```

Step 1: (Pre-process) for for loop to calculate the prefix sum **for each column**. $O(n^2)$

```
Step 2:
for (top = 0 … n) {
        for (bottom = top … n) {
                Step 2.1: (拍扁) Calculate S[] by using the column-wise prefix sum.        O(n)
                                                                                          +
                Step 2.2 Use S[] as input to call Question 0. ⇒ (sum, left, right)        O(n)
                        if (sum > global_max) {
                                update global_max
                                update solu_top, solu_bottom, solu_left, solu_right
                        }
        }
}
return the global_max and the solution
```