Lecture 7 : 2D Arrays

(21) Given N*N matrix, return an array of its anti-diagonals

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \Rightarrow \begin{matrix} 1 & 0 & 0 \\ 2 & 4 & 0 \\ 3 & 5 & 7 \\ 6 & 8 & 0 \\ 9 & 0 & 0 \end{matrix}$$

Explanation on Pg. 31 $\longrightarrow$

Code on Pg. 32 $\longrightarrow$

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

$$\begin{bmatrix} \underline{1} & & \\ \underline{2} & \underline{4} & \\ \underline{3} & \underline{5} & \underline{7} \\ \underline{6} & 8 & \\ 9 & & \end{bmatrix}$$

j = 0 to 2

$i = 0 \quad j = 0 \quad Bj = 0$

while (j >= 0):

B[0].append(1)

$i \to 1$

$j = -1$ ×

$Bj \to 1$

$j = 1 \quad i = 0 \quad Bj = 1$

B[1].append(~~A~~[0, 1])

(2)

$i \to 1 \quad j = 0$

B[1].append(4)

$j = 2 \quad i = 0 \quad Bj = 2$

B[2].append(3)

(5)

(7)

i = 1, 2

$j = 2 \quad i = 1 \quad Bj = 3$

while (i < 3):

B[3].append(6)

(8)

$i \to 2$

$j = 1$

$Bj = 4$

$i = 2$

$j = 2 \quad i = 2 \quad Bj = 4$

B[4].append(9)

```python
# @return a list of list of integers
def diagonal(self, A):
    sizeOfMatrix = len(A)
    B = [[] for _ in range(2* sizeOfMatrix -1)]
    # print(B)
    Bj =0
    for j in range(sizeOfMatrix):   (Upper Triangle)
        i = 0
        while(j >= 0 ):
            # print(i,j)
            B[Bj].append(A[i][j])
            i+=1
            j-=1
        Bj+=1
    # print(B)                              (Lower Triangle)
    for i in range(1,sizeOfMatrix):
        j = sizeOfMatrix-1
        while(i < sizeOfMatrix ):
            # print(i,j)
            B[Bj].append(A[i][j])
            i+=1
            j-=1
        Bj+=1
    for Bi in range(2 *sizeOfMatrix -1):
        while (len(B[Bi])< sizeOfMatrix):
            B[Bi].append(0)
    # print(B)
    return B
```

Lecture 8 : Interview Problems

(22) Given matrix, print boundary in clockwise directn

```
i = 0 , j = 0
while N ≥ 1 :
    for k in range (0, N-1):
        print ( A[i][j] )
        j += 1

    for k in range (0, N-1):
        print ( A[i][j] )
        i += 1

    for k in range (0, N-1):
        print ( A[i][j] )
        j -= 1

    for k in range (0, N-1):
        print ( A[i][j] )
        i -= 1
    i += 1 , j += 1
    N = N - 2
if N % 2 == 1
    print A[N//2][N//2]
```

1	14	15	19
8	9	10	20
7	12	11	21
6	24	23	22

N = 4

(23) Make maximum no. of consecutive 1's by swapping a 0 with 1

1

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

```
n = len(A)
left = [0]*n
right = [0]*n
ans = 0

# count 1's
for i in range(n):
    if A[i] == "1":
        count1 += 1
if count1 == n:
    return n

# count 1's on left
if A[0] == "0":
    left[0] = 0
else:
    left[0] = 1

for i in range(1, n):
    if A[i] == "0":
        left[i] = 0
    else:
        left[i] = left[i-1] + 1
```

```
# count 1's on the right
right[n-1] = int(A[n-1])
for j in range(n-2, -1, -1)
    if A[j] == "0":
        right[j] = 0
    else:
        right[j] = right[j+1] + 1

L = 0
R = 0

for k in range(n):
    if A[k] == "0":

        if k == 0:
            L = 0
        else:
            L = left[k-1]

        if k == n-1:
            R = 0
        else:
            R = right[k+1]

        total = L + R + 1

        if total > count1:   # no 1's to swap
            total = count1

        ans = max(ans, total)
        total = 0
return ans
```

24 Given an integer A. Generate a square matrix 1 to A^2 elements

I/p = 4 $\Rightarrow \begin{bmatrix} 1 & 2 & 3 & 4 \\ 12 & 13 & 14 & 5 \\ 11 & 16 & 15 & 6 \\ 10 & 9 & 8 & 7 \end{bmatrix}$

Same logic as 22

(25) Return maximum size subarray of A w/ all non-negative elements. If there are multiple subarrays, choose the one with lowest starting index.

[5, 6, -1, 7, 8]
0 1 2 3 4

```
N = len(A)
maximum = 0
start_index = 0
count = 0

for k in range(N):
    if A[k] > 0:
        if count == 0:
            i = k  # starting pt. of +ve subarray
    count += 1

    if A[k] < 0 or (k == N-1):
        if count > maximum:
            start_index = i  # start pt. of answer
            maximum = count  # logs length
        count = 0

return A[start_index : start_index + maximum]
```