Lecture 6 : Subarrays

How many subarrays are there? $\frac{N*(N+1)}{2}$

15 Find sum of every subarray

Brute Force

```
n = len(A)
for i in range(n):
    for j in range(i, n):
        sum = 0
        for k in range(i, j+1):
            sum += A[k]
        print(sum)
```

Using prefix sum

```
ps = [0]* n
ps[0] = A[0]

for i in range(1,n):
    ps[i] = ps[i-1] + A[i]

for i in range(n)
    for j in range (i, n);
        if  i == 0 :
            print ( ps[j]):

        else :
            print ( ps[j] - ps[i-1])
```

(16) Find the sum of all subarrays

Brute Force

```
n = len(A)
res = 0
for i in range(n):
    for j in range(i, n):
        sum = 0
        for k in range(i, j+1):
            sum += A[k]
        print(sum) ← sum of a subarray
        res += sum
print(res) ← sum of all subarrays
```

$[a_0, a_1, a_2, a_3]$

⇒ $[a_0]$

$[a_0\ a_1]$

$[a_0\ a_1\ a_2]$

$[a_0\ a_1\ a_2\ a_3]$

$[a_1]$

$[a_1\ a_2]$

$[a_1\ a_2\ a_3]$

$[a_2]$

$[a_2\ a_3]$

$[a_3]$

$$\underset{(1)(4)}{4^* a_0} + \overset{(2)(3)}{6^* a_1} + \overset{(3)(2)}{6^* a_2} + \overset{(4)(1)}{4^* a_3}$$

Optimized

```
n = len (A)

ans = 0
for i in range(n):
    ans += (i+1)*(n-i) * A[i]

return ans
```

17 Find the contiguous non-empty subarray within an array A of length N, with the largest sum

[1, 2, 3, 4, -10] ⇒ 10

```
n = len(A)

max-sum = min(A)
sum = 0

for i in range(n) :
        sum += A[i]
        max-sum = max( max-sum, sum)

        if sum < 0 :
            sum = 0
return max-sum
```

(18) Given a subarray of size N, find the subarray of size K with the least average return the first Index of the subarray

```
n= len(A)

ps = [0]* n
ps[0] = A[0]

min-sum = min(A) , temp-sum = 0
ans = 0
for i in range(1,n):

    ps[i] = ps[i-1] + A[i]

for j in range (0, n-k+1):
    if j==0:
        temp-sum = ps[k-1]

    else :

        temp-sum = ps[j+k-1] -ps[j-1]

    if temp-sum < min-sum:
        min-sum = temp-sum
        ans = j
return ans
```

Kadane Algorithm

19 Find the maximum sum of contiguous array A where sum < B

```
n= len(A)
if B > min(A):
    max = min(C)
    for i in range(n):
        sum = 0
        for j in range(i,n):
            sum += A[j]
            if (sum <= B) and (sum >max):
                max = sum
    return max
else :
    return 0
```

20 Given an array A of N elements, containing only 0's and 1's. Return all the Indices which can act as center of alternating subarrays of length 2*B+1

A = [1 , 0 , 1 , 0 , 1]

B = 1

length = 2*1+1 = 3

⇒ indexes [1, 2, 3]

```
n = len(A)
if B == 0 :
    return list(range(0, n))

subarray_len = 2*B+1
if subarray_len % 2 == 0 :
    return []

ans = []
prev = A[0]
start = 0

for i in range(1, n):
    if start > n - subarray_len :
        break

    if A[i] == prev :
        start = i

    if (i - start + 1 == subarray_len):
        ans.append(start + (subarray_len//2))
        start += 1
    prev = A[i]
return ans
```