Arrays: Sliding Window & Contribution Technique

Problem

Given array of integers, find sum of all possible subarrays of the array & maintain max sum.

a: [1, 2, 3]

[1] = 1

[1, 2] = 3

[1, 2, 3] = 6

[2] = 2

[2, 3] = 5

[3] = 3

Brute force

```
max_sum = ? -INF / a[0]
for(i=0; i<n; ++i) {
    for(j=i; j<n; ++j) {
        //(i,j) subarray
        sum=0;
        for(k=i; k<=j; ++k) {
            sum += a[k]
        }
```

TC: O(N^3)

SC: O(1)

```
        max_sum = max(max_sum, sum);
    }
}
print(max_sum)
```

Using prefix sum

```
pf[n] → creat prefix sum array TODO

max_sum = a[0]
for(i=0; i<n; ++i) {
    for(j=i; j<n; ++j) {
        //[i,j] subarray
        sum=0;
        if(i==0) {
            sum = pf[j]
        } else {
            sum = pf[j] - pf[i-1]
        }
        max_sum = max(max_sum, sum);
    }
}
print(max_sum)
```

TC : $O(N^2)$

SC : $O(N)$

Using Carry-forward

find sum of all subarrays starting from index 0.

[0,0) [0,1) [0,2) . . . [0, n-1)

```
sum=0
for(j=0; j<n; ++j){
    sum += a[j];
    print(sum)
}
```

a = [1 2 3 4]

sum=0

j=0 sum += a[0] = 1

j=1 sum += a[1] = 3

j=2 sum += a[2] = 6

j=3 sum += a[3] = 10

```
max_sum = a[0]
for(i=0; i<n; ++i){
    sum=0
    for(j=i; j<n; ++j){
        sum += a[j];
        ~~print(sum)~~ → print sum of all subarrays
        max_sum = max(max_sum, sum);
    }
}
print(max_sum)
```

TC : $O(N^2)$

SC : $O(1)$

$O(N^2)$ subarrays in an array

Problem

Given array of integers, find total sum of all possible subarrays.

arr = [6, 8, -1, 7]

[0,0] → 6

[0,1] → 6+8 = 14

[0,2] = 6+8-1 = 13

[0,3] = 20

[1,1] = 8

[1,2] = 7

[1,3] = 14

[2,2] = -1

[2,3] = 6

[3,3] = 7

6+14+13+20+8+7+14

-1+6+7

= 94

Use prefix sum

```
total_sum = 0
for (i=0; i<n; ++i) {
    sum = 0
    for (j=i; j<n; ++j) {
        sum += a[j];
        total_sum += sum
    }
}
print (total_sum)
```

TC : $O(N^2)$

SC : $O(1)$

	0	1	2	3	4	5
A[6] =	3	-2	4	-1	2	6

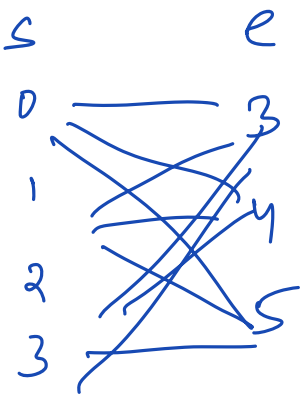
In how many subarrays, index 3 is present?

[0,3] [0,4] [0,5]

[1,3] [1,4] [1,5] = 12 subarrays

[2,3] [2,4] [2,5]

[3,3] [3,4] [3,5]



$= 4 \times 3 = 12$

In how many subarrays, index 1 is present?

s	e
0	1
1	2
	3
	4
	5

$\Rightarrow 2 \times 5 = 10$.

In how many subarrays, index 2 is present?

s	e
0	2
1	3
2	4
	5

$\Rightarrow 3 \times 4 = 12$

Generalize: Given N elements, # of subarrays where i^{th} index is present.

a[n] : $a_0\ a_1 \ldots a_{i-1}\ a_i\ a_{i+1} \ldots a_{n-1}$

$s : [0, i]$ $\quad$ $e : [i, n-1]$

$= i - 0 + 1$ $\quad$ $= n-1-i+1$

$= i+1$ $\quad$ $= n-i$

$\Rightarrow (i+1) \times (n-i)$

	0	1	2	3	$n=4$
A[4] :	6	8	-1	7	
i+1 :	1	2	3	4	
n-i :	4	3	2	1	
(i+1)*(n-i) :	4	6	6	4	

individual contribution $24 + 48 + -6 + 28 = 94$

N=3

a[3] :	4	3	7
i+1 :	1	2	3
n-i :	3	2	1
(i+1)*(n-i) :	3	4	3
contri :	12	12	21

$= 45$

[4] $= 4$

[4 3] $= 7$

[4 3 7] $= 14$

[3] $= 3$

[3 7] $= 10$

[7] $= 7$

```
total_sum = 0

for (i=0; i<n; ++i) {
    count = (i+1)*(n-i);
    contri = count*a[i]
    total_sum += contri
}
print (total_sum)
```

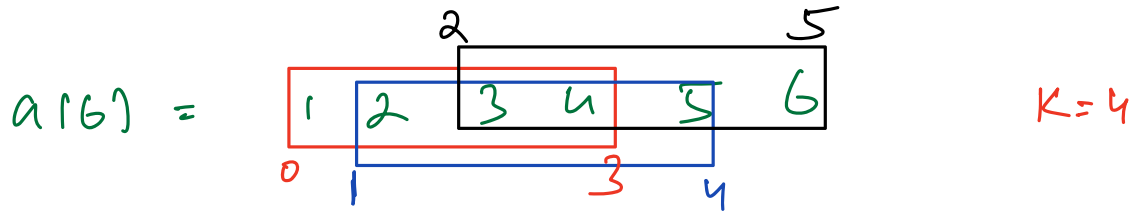
Total number of subarrays of length K

K=1 [0,0] [1,1] [2,2] . . . [n-1, n-1] = N

K=2 [0,1] [1,2] [2,3] . . . [n-2, n-1] = N-1

[0, K-1] [1, K] . . . . [n-K, n-1] = N-K+1

total subarrays of len K = N-K+1



N=7, K=4 $\quad\quad$ 7-4+1 = 4

Problem

Given an array of size N, print start & end indicies of subarrays of len K.

n=8 $\quad$ K=3

i	j
0	2
1	3
2	4
3	5
4	6
5	7

j = i+K-1

```
for (i=0; i<n-K+1; ++i) {
    j = i+K-1;
    print (i,j)
}
```

TC : O(N)

SC : O(1)

Problem

Given array of size N, print max subarray sum of subarrays of len=K.

N=10 K=5

-3 4 -2 5 3 -2 8 2 -1 4

[0 4) → 7

[1 5) → 8

Brute force

```
max_sum = -INF
for (i=0; i<n-k+1; ++i) {
    j = i+k-1;
    sum=0
    for (k=i; k<=j; ++k) {
        sum += a[k]
    }
    max_sum = max(max_sum, sum)
}
print(max_sum)
```

$O(N^2)$

$O(1)$

Prefix Sum

```
pf[n] -> TODO
max_sum = -INF
for (i=0; i<n-k+1; ++i) {
    j = i+k-1;
    if (i==0)
        sum = pf[j]
    else
        sum = pf[j] - pf[i-1]
    max_sum = max(max_sum, sum)
}
print(max_sum)
```

TC: O(N)
SC: O(N)

Optimization

Carry forward aka Sliding Window

K=5

a[10] = | -3 | 4 | -2 | 5 | 3 | -2 | 8 | 2 | -1 | 4 |

i=0, i=1, j=4, j=5

$sum_1 = -3+4-2+5+3 = 7$ [0,4]

$sum_2 = 7 - (-3) + (-2) = 8$ [1,5]

$sum_2 = sum_1 - a[i-1] + a[j]$

$Sum_3 = 8 - 4 + (8) = 12$ [2,6]

```
sum=0
for(i=0; i<K; ++i) {
    sum+=a[i]          → first subarray sum [0,K-1]
}
max_sum = sum;
i=1, j=K;              → second subarray [1,K]
while(j<n) {
    sum = sum - a[i-1] + a[j];
    max_sum = max(max_sum, sum);
    i++, j++
}
print(max_sum)
```

TC: O(N)

SC: O(1)

Tips / Observation for subarray problems

→ subarray are contigous, can be denoted using [start, end] indicies.

→ total subarrays = $\frac{n(n+1)}{2}$

→ to print all possible subarrays = $O(N^3)$

→ sum of all subarrays can be done

- $O(N^2)$
  - $O(N)$
  - prefix sum
- $O(N^2)$
  - $O(1)$
  - carry forward

→ total subarrays of len K = $n-K+1$