Song Suggestion : Allen [Winner last Quiz]

Agenda

Sliding Window

Contribution Technique

Question 1

Given an array of Integers, we need to find the sum of all possible subarrays of the array and return the maximum sum

example

arr = [1, 2, 3]

ans = 6

[1]	1
[1, 2]	3
[1, 2, 3]	6
[2]	2
[2, 3]	5
[3]	3

Brute force solution

List down all possible subarray Si & Ei and count the sum of subarray

```
arr = [ ];
max_sum = Int_Min;
for (i=0; i< n; i++) {
    for (j= i; j< n; j++) {
        sum =0
        for (k = i; k <= j; k++)  sum += arr[k];
        max_sum = max ( max_sum, ans);
```

T.C = $O(n^3)$

S.C = $O(1)$

Observation

arr =

1	2	3
0	1	2

Subarray	(si, ei)
[1]	(0, 0)
[1, 2]	(0, 1)
[1, 2, 3]	(0, 2)
[2]	(1, 1)
[2, 3]	(1, 2)
[3]	(2, 2)

claim :

We have solved this before

Q queries, given si & ei

Prefix Sum Arrays

pseudo code

```
void max subarray sum (int arr[], int N) {
    // calculate prefix sum Array
    psum[n];
    max_sum = INT_MIN;
    for (i=0; i<n; i++) {  //si
        for (j=i; j<n; j++) {  //ei
            if (i==0)
                sum = psum[j]
            else
                sum = psum[j] - psum[i-1];
            max_sum = max (max_sum, sum);
        }
    }
```

T.C = $O(n^2)$

S.C = $O(n)$

Can we do better ? // can we reduce space

arr :

-4	1	3	2
0	1	2	3

s	e	k	sum
0	0	1	sum = arr[0] (a)
0	1	2	sum = arr[0] + arr[1] (b)

Hint :

we have repeated calculations

Observation

arr :

-4	1	3	2
0	1	2	3

Run two loops i, j

i = 0 currsum = 0

j	subarray	currsum	value	Max
0	(0,0)	0 + arr[0]	-4	-4
1	(0,1)	-4 + arr[1]	-4 + 1 = -3	-3
2	(0,2)	-3 + arr[2]	-4 + 1 + 3 = 0	0
3	(0,3)	0 + arr[3]	-4 + 1 + 3 + 2 = 2	2

claim : use previous ans to get next ans

i = 1 currsum = 0

j	subarray	currsum	value	max
1	(1,1)	+= arr[1]	0+1	1
2	(1,2)	+= arr[2]	1+3 = 4	4
3	(1,3)	+= arr[3]	1+3+2 = 6	6

i = 2 currsum = 0

j	subarr	currsum	value	max
2	(2,2)	+= arr[2]	0+3 = 3	6
3	(2,3)	+= arr[3]	0+3+2 = 5	6

pseudo code

```
int ans = INT_MIN;
for (i=0; i<n; i++) {
    curr_sum = 0;
    for (j=i; j<n; j++) {
        curr_sum += arr[j]
        ans = max (ans, curr_sum)
    }
}
```

T.C = O(n²)

S.C = O(1)

}

Question 2

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

previously asked on Google & Facebook

example

arr = | 1 | 2 | 3 |
0 1 2

= 20

[1]	1
[1,2]	3
[1,2,3]	6
[2]	2
[2,3]	5
[3]	3

Brute force

We can use prev solution

pseudo code

```
int ans = 0;
for (i=0; i<n; i++) {
    curr_sum = 0;
    for (j=i; j<n; j++) {
        curr_sum += arr[j]
        ans = ans + curr_sum;
    }
}
```

T.C : $O(n^2)$

S.C = $O(1)$

}

Contribution Technique

example

arr = | 1 | 2 | 3 |
0 1 2

[1]	
[1,2]	
[1,2,3]	
[2]	
[2,3]	
[3]	

Observation

Calculate occurrence to get total sum

How many times 1 appears in subarray ?

3 times

How many times 2 appears in subarray ?

4 times 

How many times 3 appears in subarray ?

2 times

Total : $(1 * 3) + (2 * 4) + (3 * 3)$

$= 3 + 8 + 9 = 20$

How to calculate the # subarrays A[i] appears?

Quiz 1

In how many subarrays, the element at index [1] will be present?

arr =

3	-2	4	-1	2	6
0	1	2	3	4	5

10 times

(0,1) (0,2) (0,3) (0,4)
(0,5) (1,1) (1,2) (1,3)
(1,4) (1,5)

Quiz 2

In how many subarrays, the element at index [2] will be present?

arr =

3	-2	4	-1	2	6
0	1	2	3	4	5

12 times

(0,2) (0,3) (0,4) (0,5)
(1,2) (1,3) (1,4) (1,5)
(2,2) (2,3) (2,4) (2,5)

Generalizing calculation A[i]

start indices $[0, i] = i - 0 + 1 =$ i+1 options

end indices $[i, n-1] = n-1-i+1 =$ n-i options

total occurrence of A[i]

$(i+1) * (n-i)$

Contribution of A[i]

$A[i] * (i+1) * (n-i)$

pseudo code

```
int sumOfSubarraySum(int[] arr, int n){
    int total_sum = 0;
    for(i=0; i<n; i++){
        // occurrence of A[i]
        occ = (n-i) * (i+1)
        total_sum += (arr[i] * occ);
    }
}
```

T.C = O(n)

S.C = O(1)

Total number of subarrays with length = K

arr =

1	2	3	4	5	6	7	8	9	10
0	1	2	3	4	5	6	7	8	9

length (K)	start of 1st window	start of last window
1	0	(n-1)
2	0	(n-2)
3	0	(n-3)
K	0	(n-K)

total number of subarray with len = K

range of start index [0, n-K]

n - K - 0 + 1

n - K + 1 → total subarray with leng = K

Quiz 3

Given N = 7 and K = 4, what will be the total number of subarrays of len K?

7 - 4 + 1 = 4

Question 3

Given an array of size N, print start & end indices of subarray of length K

example

N = 8 and K = 3

start	end
0	2
1	3
2	4
3	5
4	6
5	7

Generalizing

[start , end] = K

end - start + 1 = K

end = K + start - 1

pseudo code

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

Question 4

Given an array of N elements. Print maximum subarray sum with length = K

arr =

-3	4	-2	5	3	-2	8	2	-1	4
0	1	2	3	4	5	6	7	8	9

K = 5

s	e	sum
0	4	7
1	5	8
2	6	12
:	:	

pseudo code

T.C = (n-k+1)·k

```
ans = INT_MIN
// subarray with length k
start = 0 ; end = k-1 // first window
while (end < n) {
    sum=0;
    for ( Idx = start ; Idx <= end ; Idx++) {
        sum += arr[Idx];
    }
```

start ++; end ++;

}

Substitute value of k to get accurate T.C

Time Complexity

k = 1	k = n	k = n/2
$(n-k+1) \cdot k$	$(n-k+1) \cdot k$	$(n-k+1) \cdot k$
$(n-1+1) * 1$	$(n-n+1) \cdot n$	$(n-\frac{n}{2}+1) \cdot \frac{n}{2}$
$O(n)$	$O(n)$	$(\frac{n}{2}+1)\frac{n}{2}$
		$O(n^2)$

Optimization

Approach 1

Use prefix sum approach to get the sum of contigious subarray

1. I can generate all Si & Ei in O(n)
2. Applying Prefix sum on top of Sp and Ep will be O(n)

Quiz 4

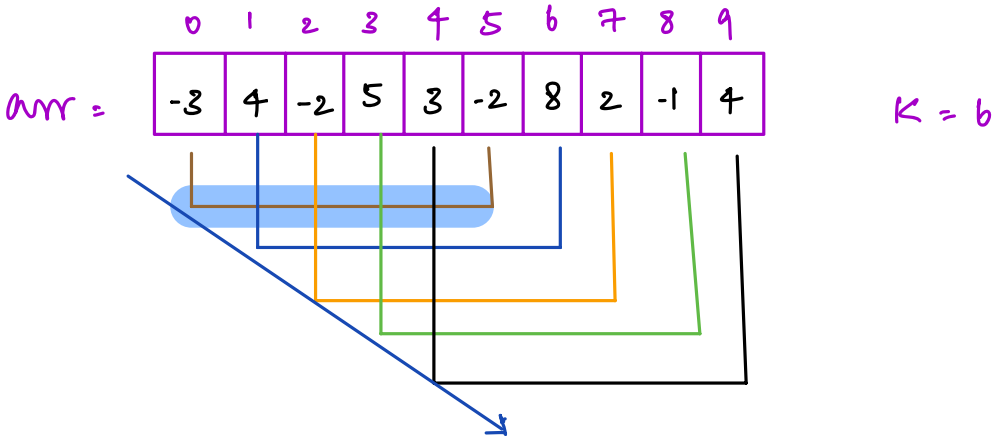
What is the time complexity & space complexity respectively ?

O(n) T.C

O(n) S.C

Approach 2

Observation



sp	ep	ans
0	5	-3+4-2+5+3-2
1	6	4-2+5+3-2+8
2	7	-2+5+3-2+8+2

Sliding window approach

we add the new last element

subtract the old first element

#pseudo code

```
ans = INT_MIN
i = 0;   j = k-1;   // first window
sum = 0;
for ( i = i;   i <= j;   i++) {
        sum = sum + arr[i]
}
ans = max (ans, sum);
// we get first windows sum

i++ ;  j++
while (j < n) {
    sum = sum + arr[j] - arr[i-1];
    ans = max (ans, sum);
    i++; j++;
}
print (ans);
```

Observations

prefix sum : Repetedly peform range sum in given array

Contribution technique: Sum of all sums

sliding window : fixed length of subarray

Carry Forward

If you redo the same calculation

Why 2D matrices ?

Grid like structure

DP