Todays Content

1. Count of subarrays with sum = k
2. Distinct elements in each subarray of size = k

18 Given ar[N] & k

Calculate & return no: of subarrays with sum = k.

k = 4

	0	1	2	3	4	5	6	7	8
ar[] = {	3	4	1	3	-4	5	-1	-5	-6 }

Subarrays: {1 1} {2 3} {5 6} {3 5} {1 4} {2 6} ans = 6

Idea: Generate all subarrays & calculate sum & if sum == k: cnt++;

Way1: TC: O(N3) SC: O(1)

```
int ans = 0;
for(s=0; s<N; s++){
    for(e=s; e<N; e++){
        // [s..e]
        sum = 0;
        for(i=s; i<=e; i++){
            sum = sum + ar[i];
        }
        if(sum == k){
            ans++;
        }
    }
}
return ans;
```

TC: O(N2) SC: O(N)

```
int ans = 0;
Create pf[N];
for(s=0; s<N; s++){
    for(e=s; e<N; e++){
        // [s..e]
        sum = 0;
        if(s==0){ sum = pf[e] }
        else{ sum = pf[e] - pf[s-1] }
        if(sum == k){
            ans++;
        }
    }
}
return ans;
```

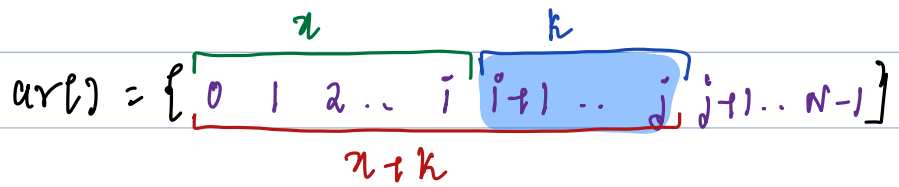
Idea2: Create pf[]

	0	1	2	3	4	5	6	7	8	9
ar[] = {	4	-2	4	-1	4	3	6	-4	1	5 }
psum[] = {	4	2	6	5	9	12	18	14	15	20 }

k = 10

Obs: if pf[j] - pf[i] == k # sum[i+1..j] = k

Assum pf[i] = x, pf[j] = x + k



Con: Calculate no: of pairs (i, j) are there such that

pf[j] - pf[i] == k && j > i

DryRun:

	0	1	2	3	4	5	6	7
ar[] = {	6	2	3	2	-7	2	3	2 }

k = 5

	0	1	2	3	4	5	6	7
pf[] = {	6	8	11	13	6	8	11	13 }

cnt1 = 0

cnt2 = 0

cnt6 = 1

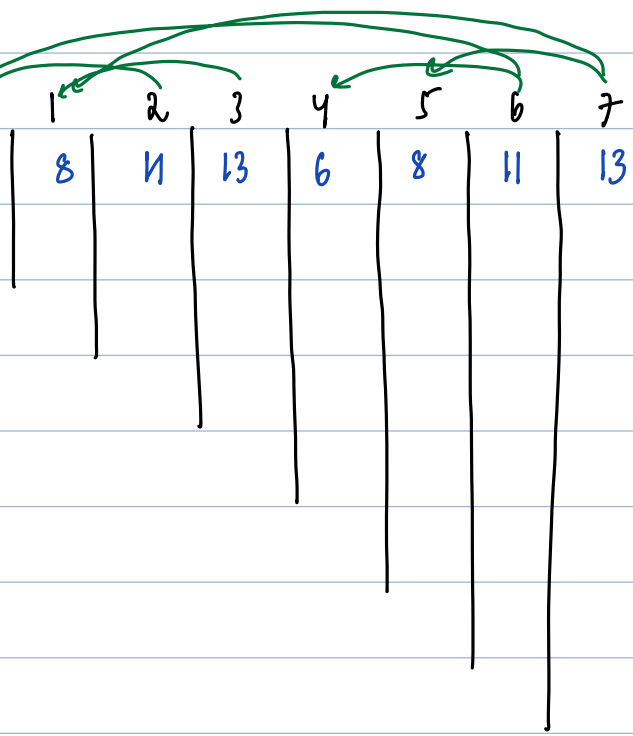
cnt8 = 1

cnt1 = 0

cnt3 = 0

cnt6 = 2

cnt8 = 2 ans = 6



Idea: For every pf[j]:
Count frequency of pf[j]-k in left of j: [0..j-1]

Opt: HashMap:
At pf[j]: We only search on [0..j-1]
At pf[j]: HashMap should only contain all ele from [0..j-1]

Dry Run: pf[j] - pf[i] == k

	0	1	2	3	3	4	5	6	7
ar[] =	6	2	3	-6	2	4	2	3	2

k = 5

	0	1	2	3	4	5	6	7	8
pf[] = 0	6	8	11	5	7	11	13	16	18
Target =	1	3	6	0	2	6	8	11	13
cnt =	0	0	1	0	1	0	1	2	1

= ans = 7

↳ 0 is coming at the start of pf() itself, hence insert <0:1> in hashmap, where 1 is indicating it's frequency.

HashMap:

<0:1>
<6:1> <13:1>
<8:1> <16:1>
<11:2> <18:1>
<5:1>
<7:1>

```
int subarrays(vector<int> &ar, long k){ TC: O(N+N) = O(N)
                                        SC: O(N+N) = O(N)
    long pf[N];
    long sum=0;
    for(int i=0; i<ar.size(); i++){
        sum = sum + ar[i];
        pf[i] = sum;
    }

    unordered_map<long, int> hm;
    hm[0]=1;
    int c=0;
    for(int j=0; j<N; j++){
        # Pf[j] Target = Pf[j] - k
        if(hm.find(Pf[j]-k) != hm.end()){
            c = c + hm[Pf[j]-k];
        }

        # Insert Pf[j];
        hm[ Pf[j] ]++;
    }

    return c;
}
```

Q1: Given ar[N] & k

Print no: of distinct elements in every subarray of size = k

Constraints:

$1 <= N <= 10^6$

$-10^9 <= ar[i] <= 10^9$

Eg:

	0	1	2	3	4	5	6	7	8
ar[9] = {	2	4	2	8	4	5	3	4	5 }

k = 5

Output:

s	e	Distinct
[0	4]	3
[1	5]	4
[2	6]	5
[3	7]	4
[4	8]	3
[5	9]	stop

Ideal: For every subarray of len = k

Calculate no: of distinct elements ?

↳ Create hashset insert all subarray elements, hashset size will get no: of distinct elements.

TC: O(N-k+1) * O(k) =

No: of subarrays of len = k Insert subarray in hashset

k = 1 TC: O(N-1+1) * O(1) = O(N)

k = N TC: O(N-N+1) * O(N) = O(N)

k = N/2 TC: O(N-N/2+1) * O(N/2) = O(N²) $1 <= N <= 10^6$

↳ $(10^6)^2 = 10^{12}$ TLE

Optimize: Sliding Window

```
      0  1  2  3  4  5  6  7  8
En: ar[10] = { 8  5  2  4  2  4  3  6  3 }
  k = 5
```

#Subarrays	#Remove	#Add	#Hashset	#size
[0 4]	Insert elements in HS		{8 5 2 4}	4
[1 5]	ar[0]	ar[5]	{~~8~~ 5 2 4}	3
[2 6]	ar[1]	ar[6]	{~~5~~ 2 4 3}	3
[3 7]	ar[2]	ar[7]	{~~2~~ 4 3 6}	3 *, count = 4
[ ]				

Issue: Using hashset when we remove an element, it will indirectly remove all it's occurences, to avoid we need to store freq of each element, hence go with hashmap

```
      0  1  2  3  4  5  6  7  8
En: ar[10] = { 8  5  2  4  2  4  3  6  3 }
  k = 5
```

#Subarrays	#Remove	#Add	#HashMap	#size
[0 4]	Insert all ele in hm		{8:1 5:1 2:2 4:1}	4
s e				
[1 5]	ar[0]	ar[5]	{~~8:1~~ 5:1 2:2 4:2}	3
[2 6]	ar[1]	ar[6]	{~~5:1~~ 2:2 4:2 3:1}	3
[3 7]	ar[2]	ar[7]	{2:1 4:2 3:1 6:1}	4
[4 8]	ar[3]	ar[8]	{2:1 4:1 3:2 6:1}	4
[5 9]	: Exceeding ar[] stop process			

Note: if freq == 0, we need to remove pair from hashmap.

s-1 [s e]	ar[s-1] *	ar[e]	#Attach in future	

```
void DistinctWindow(vector<int> &ar, int k){ TC: O(k + N-k) = O(N)
    unordered_map<int, int> hm;                    SC: O(k)
    # Insert 1st subarray in hm [0..k-1]
    for(int i=0; i<k; i++){
        hm[ar[i]]++;
    }
    print(hm.size());

    # Apply sliding window
    int s=1, e=k;
    while(e<N){
        # [s..e] : remove ar[s-1] and ar[e]
        hm[ar[s-1]]--;
        if(hm[ar[s-1]]==0){
            hm.erase(ar[s-1]);
        }

        hm[ar[e]]++;
        print(hm.size());
        s++; e++; # Goto next subarray
    }
}
```