Today's content

→ Check if there exists a pair (i, j) such that $A[i] + A[j] == K$ && $i != j$

→ Get no. of distinct elements in every window of size = k

→ Find length of longest subarray with sum = 0

1. Given N array elements, check if there exists a pair (i, j) such that ar[i] + ar[j] = k and i != j → bool

	0	1	2	3	4	5	6	7	8	9
ar[] :	8	9	1	-2	4	5	11	-6	7	5

	i	j	ar[i]	ar[j]	
k = 11	4	8	4	7	T
k = 6	2	5	1	5	T
k = 22					F

8 + ___ = 11

11 - 8

Idea 1 : Check all unique pairs

a + b = k

b = k - a

```
for (i = 0 ; i < n ; i++) {
        a = ar[i] , b = k - ar[i]
    for ( j = i + 1 ; j < n ; j++) {

        if (ar[i] + ar[j] == k)
            return true

    }
}
return false
```

TC: O(N²)

	0	1	2	3	4	5	6	7	8	9
ar[] :	8	9	1	-2	4	5	11	-6	7	5

Idea 2 : Create Hashset <int> hs

Insert entire data in hs

{8, 9, 1, -2, 4, 5, 11, -6, 7, 5}

$k = 11$

a	b → k-a	is b in HS
8	3	✗
9	2	✗
1	10	✗
-2	13	✗
4	7	✓ return True

$k = 22$

a	b	is b in HS
8	14	✗
11	11	✓ True ✗

a == b

	0	1	2	3	4	5	6	7	8	9
ar[]:	8	9	1	-2	4	5	11	-6	7	5

Obs: If we know freq, we can solve prob

1. Hashmap <int, int> hm → freq, → ar[i]

2. Iterate & insert in hm

{ 8 : 1, 9 : 1, 1 : 1, -2 : 1, 4 : 1, 5 : 2, 11 : 1, -6 : 1, 7 : 1 }

$k = 22$

a	b → k-a	is b in HM
8	14	✗
9	13	✗
1	21	✗
-2	24	✗

4	18	✗
5	17	✗
11	11	✗

if(a==b) freq(b) > 1

Ex 7 5 2 5 9 5 k=10

HM { 7 : 1, 5 : 3, 2 : 1, 9 : 1 }

a	b → k-a	is b in HM
7	3	✗
5	5	if (freq(5) > 1) ✓

Pseudocode

1. Create hm & insert ar[ ] → n
2. for (i = 0 ; i < N; i++) { → n

```
    a = ar[i]
    b = k - a
    if (a != b) {
        if (b is in HM)
            return true
    }
    else { // a == b
        if (freq(b) > 1)
            return true
    }
}
return false
```

1 iter → O(1)

n → n × O(1)

TC : O(n)

SC : O(n)

	0	1	2	3	4	5	6	7	8	9	10	11
arr[] :	8	9	1	-2	4	5	11	-6	7	5	11	11

a

Idea: when we are at i^{th} index, all elements from [0 - i-1] in HS

k = 22

a	b → k-a	HS	
8	14	< >	✗ Insert 8
9	13	<8>	✗ Insert 9
1	21	<8,9>	✗ Insert 1
-2	24	<8,9,1>	✗ Insert -2
4	18	<8,9,1,-2>	✗ Insert 4
5	17	<8,9,1,-2,4>	✗ Insert 5
11	11	<8,9,1,-2,4,5>	✗

k = 12

a	b		
8	4	< >	✗
9	3	<8>	✗
1	11	<8,9>	✗
-2	14	<8,9,1>	✗
4	8	<8,9,1,-2>	✓ T

Pseudocode -

```
hashset <int> hs;

for (i=0; i<n; i++) {
    a=ar[i]
    b = k-a
    if (b in HS)
        return true
    hs.insert(a)
}

return false
```

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

Q. calculate no. of i, j such that ar[i] + ar[j] = k

hashmap

elem, freq

TODO

Q. Given N array elements, check if there exists a pair (i, j) such that ar[i] - ar[j] = k and i != j

2. Given N array elements, calculate no. of distinct elements in every subarray of size k.

	0	1	2	3	4	5	6	7
ar[] :	2	4	3	8	3	9	4	9

$K = 4$

k

[0 - N]

	Distinct
[0-3]	4
[1-4]	3
[2-5]	3
[3-6]	4
[4-7]	3

Idea:
for every window, get no. of distinct elements

[0 k-1]
[1 k]
[2 k+1]
[3 k+2]
⋮
[N-k N-1]

print no. of dist element in every window of size = k

```
Hashset <int> hs;
for (s=0; s<=N-k; s++)
{
    for (j=s; j<s+k; j++){
        hs.insert(ar[j])
    }
    print(hs.size()
    hs.clear()
}
```

Iterations

$(N-k+1) * k$ } Max

$k \to [1 \quad N]$

if $k = N/2$

$(N - N/2 + 1) \times N/2$

$= (N/2)(N/2)$ → TC : $O(N^2)$

SC : $O(k)$ → $O(n)$

	0	1	2	3	4	5	6	7
ar[ ] :	2	4	3	8	3	9	4	9

$k = 4$

		Remove	Add	HS	no.
Initially	[0 - 3] :			<2, 4, 3, 8>	4
	[1 - 4] :	0th	4th	<4, 3, 8>	3
	[2 - 5]	1st	5th	<3, 8, 9>	3
	[3 - 6]	2nd	6th	<8, 9, 4>	3

Obs : If we remove an ele, indirectly all occurrences of same element is also removed.

	0	1	2	3	4	5	6	7
ar[ ] :	2	4	3	8	3	9	4	9

	Remove	Add	HM	
[0-3]			<2:1, 4:1, 3:1, 8:1>	⇒ 4
[1-4]	0th	4th	<2:0, 4:1, 3:2, 8:1>	⇒ 3
[2-5]	1st	5th	<4:0, 3:2, 8:1, 9:1>	⇒ 3
[3-6]	2nd	6th	<3:1, 8:1, 9:1, 4:1>	⇒ 4
[4-7]	3rd	7th	<8:0, 3:1, 9:2, 4:1>	⇒ 3

del 2

del 4

del 8

In hm, if freq = 0, remove from hm

Pseudocode

```
Given N elements, K
Hashmap <int, int> hm;

for (i=0; i < k ; i++) {
    // ar[i] insert in hm
    if (ar[i] is in hm) {
        hm[ar[i]]++
    }
    else {
        hm.insert(ar[i], 1)
    }
}

print(hm.size())
```

[0 k-1]

k

[s e]

$\underbrace{[s \quad e]}_{k}$

$$len = e - s + 1$$

$$k = e - s + 1$$

$$k + s - 1 = e$$

```
for(s=1 ; s <= N-k ; s++) {
    // subarray [s  e]                    → N-k

    e = s + k - 1

    // rm  s-1 , add e                     k+N-k
                                            = N
    hm [ar [s-1]] --;
    if (hm [ar[s-1]] == 0) {                Tc: O(N)
        hm.remove (ar[s-1])                 SC: O(k)
    }                                           ↓
                                                N
    if (ar[e] is in hm) {                   (k=N)
        hm[ar[e]] ++;
    }
    else {
        hm.insert (ar[e],1);
    }

    print (hm.size())
}
```

11:17

3. Given N array elements, find length of longest subarray sum = 0

	0	1	2	3	4	5	6	7	8	9	10	11
ar[ ] :	3	3	4	-5	-2	2	1	-3	3	-1	5	-4
pf[ ]	3	6	10	5	3	5	6	3	6	5	10	6

subarray with sum = 0

- pf[i] = pf[j] → Repetition → len = j - i
- pf[i] = 0 → sum[0 i] → len = i + 1

In case of repetition, consider farthest index → (first, last index)

	0	1	2	3	4	5	6	7	8	9	10	11
ar[ ] :	3	3	4	-5	-2	2	1	-3	3	-1	5	-4
pf[ ]	3	6	10	5	3	5	6	3	6	5	10	6
curlen					4	2	5	1	1	6	8	10
maxlen					4	4	5	1	1	1	8	10

HM → pf sum, 1st index

<3, 0>
<6, 1>
<10, 2>
<5, 3>

pf[i] in HM → repetition

len = i - hm[pf[i]]

Edge case

	0	1	2	3
Ex	-2	2	1	-1
pf[]	-2	0	1	0
				2

ans = 4

HM <ele, first index>

<-2, 0>
<0, 1>
<1, 2>

Pseudocode

1. Create pf[] → N
2. Hashmap <int, int> hm

```
ans = 0
for (i=0; i<N; i++) {        → N
    if (pf[i]==0) {
        ans = max(ans, i+1)
        continue
    }
    if (pf[i] is in hm) {
        // repetition
        ans = max(ans, i - hm[pf[i]])
    }
    else {
        hm.insert(pf[i], i)
    }
}
return ans
```

TC : O(N)

SC : O(N)

pf[] hm

worst case → n