Today's content.

(i) Pair sum = k.

(ii) Pair difference = k.

(iii) Subarray with sum = k.

} -> only the idea

(iv) Distinct elements in every window of size k.

1Q: Given 'n' array elements, check if there exists a pair [i, j] such that ar[i] + ar[j] = k & i != j.

a b

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

k = 11, : ar[4] + ar[8] = 11. → Yes.

k = 6, : ar[2] + ar[5] = 6. → Yes.

k = 22. : No.

idea: Check all pairs, see if sum = k. ⇒ TC: $O(N^2)$, SC: $O(1)$.

```
for i in range(0,N)                       // a+b=k.
    a = ar[i] , b = k-a
    for j in range(i+1, N)
        if( ar[j] == b)
            return true.

return false
```

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

idea 2: Optimize using hashset.

i) insert all elements into hashset.

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

k=11,

a	b[k-a]	b is present in hs.
8	3.	No.
9	2	No.
1	10	No.
-2	13	No.
4	7	True.

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

$a+b = 5 \leftarrow k.$

a	b[k-a].	is b present in hs.
8	-3	no.
9	-4	no.
1	4	yes.

$a+b = 22 \leftarrow k.$

a	b[k-a]	is b present in hs.
8	14	no.
9	13	no.
1	21	no.
⋮		
11	11	yes. ← This is wrong!

idea 3: using a dictionary / hashmap.

0	1	2	3	4	5	6	7	8	9
8	9	1	-2	4	5	11	-6	7	5

[8 9 1 -2 4 5 11 -6 7 5].

<8,1> <9,1> <1,1> <-2,1>
<4,1> <5,2>, <11,1> <-6,1>
<7,1>

k = 20.
a = 9   b = 11   ,

a+b = 22.

a	b [k-a]	is b present in hm (dictionary).
8	14	no.
9	13	no.
1	21	no.
⋮		
11	11	Yes ⟶ { if (a==b) freq(a) > 1 } true.

if (a==b) freq(a) > 1 } true.

if (a != b)
is b present or not.

Code:

```
def pairSum ( ar, k, n)
{
    dictionary = { }.
    // populate array elements with their freq into dictionary --> TODO.

    for i in range(0, N)
    {
        a = ar[i] , b = k-a .
        if (a==b && dictionary[a] > 1)
```

TC: O(n)
SC: O(n).

```
        return true
    if(a!=b && b in dictionary)
        return true
  }
  return false
}
```

idea 4: Can we solve it with a hashset?

	0	1	2	3	4	5	6	7	8	9	
ar(): {	8	9	5	-2	11	5	7	-6	4	1	}.

K = 22,

a	b [k-a]	hs	b is present in hs.
8	14	{ }	No.
9	13	{8}.	no.
5	17	{9,8}	no.
-2	24	{9,8,5}	no.
11	11	{9,8,5,-2}	no.

(working for our corner case).

K = 10.

a	b [k-a]	hs	b is present in hs or not.
8	2	{ }	No.
9	1	{8}	NO.
5	5	{9,8}.	NO.
-2	12	{9,8,5}	NO.
11	-1	{9,8,5,-2}	NO.
5	5	{9,8,5,-2,11}	True

0	1	2	3	4	5	6	7	8	9

ar[]: { 8 9 5 -2 11 5 7 -6 4 1 }.

code:

```
def pairSum(ar, k, n)
{
    set = set()
    for i in range(0,N)
    {
        a = ar[i], b = k-a
        if (b in set)
            return true
        set.add(a)
    }
    return false
}
```

TC: O(N).

SC: O(N).

2Q: Pair difference $= k$. , $(ar[i], ar[j]) = k$.

$b = k - a$.
$b = k + a$.

Ex: $ar[] = \{2, 4, 10, 20, 9, 3, 5, 2\}$.

$\begin{array}{cccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 2 & 4 & 10 & 20 & 9 & 3 & 5 & 2 \end{array}$

$k = 7$.

$ar[i] - \underline{ar[j]} = k$.

$ar[j] = \underline{ar[i] - k}$.

Ex2: $ar[] = \{5, 20, 3, 2, 5, 80\}$.

$k = 78$.

$\underline{ar[j]} - ar[i] = k$.

$ar[j] = ar[i] + k$.

3Q: Given an array of n elements, check if there exists a subarray whose sum = k.

Ex: k = 12 , [1, 2, 3, 7, 5]. , Yes or No. , TC: O(n).

Pf [1, 3, 6, 13, 18].

hint: extension of subarray whose sum is equal to zero.

Break:

10: 09: 00

10 : 14 : 00 , 10:15:00

4Q: Given N array elements, calculate the no. of distinct elements in every window of size K.

Ex:

	0	1	2	3	4	5	6	7	8	9
ar[10] =	2	4	3	8	3	9	4	9	4	10

k = 4.

Subarrays	Output.
[0,3]	4
[1,4]	3
[2,5]	3
[3,6]	4
[4,7]	3
[5,8]	2
[6,9]	3

idea: For every window of size k, insert into hashset & get the no. of distinct elements.

TC: $O(n-k+1)(k)$, SC: $O(k)$.

$k=1$, $(n-\cancel{1}+\cancel{1})(1) = n.$

$k=N$, $(\cancel{n}-\cancel{n}+1)(n) = n$

$k=N/2$ $(n-n/2+1)(n/2) \simeq O(n^2).$

Optimization: Optimization using sliding window with hashset.

	0	1	2	3	4	5	6	7	8	9
ar[10] =	2	4	3	8	3	9	4	9	4	10

window (s e)	remove (s-1)	add (e)	hashset	#distinct
[0-3]			[2 4 3 8]	4.
[1-4]	ar[0]	ar[4]	[4 3 8]	3.
[2-5]	ar[1]	ar[5]	[3 8 9]	3.
[3-6]	ar[2]	ar[6]	[8 9 4]	3. → Not correct.

When can we safely remove from set?

if freq(element) = 1, we can safely remove.

We need to store frequency along with ~~the~~ elements $\Rightarrow$ map / dictionary.

idea 3: Optimization with sliding window using hashmap /dictionary.

	0	1	2	3	4	5	6	7	8	9
ar[10] =	[2	4	3	8	3	9	4	9	4	10]

s	e	remove ar[s-1]	add ar[e]	hashmap.	size.
[0	3]			{(2,1),(4,1)(3,1)(8,1)}	4.
[1	4]	ar[0]	ar[4]	{(4,1)(3,2)(8,1)}	3.
[2	5]	ar[1]	ar(5)	{(3,2)(8,1)(9,1)}	3.
[3	6]	ar(2)	ar[6]	{(3,1)(8,1)(9,1)(4,1)}	4.
[4	7]	ar(3)	ar[7]	{(3,1)(9,2)(4,1)}	3.
[5	8]	ar[4]	ar[8]	{(9,2)(4,2)}	2.
[6	9]	ar[5]	ar[9]	{(9,1)(4,2)(10,1)}	3.

Code.

```
def  distinctWindow (ar, k, n)
{
    hm = { }.
    for  i  in  range(0, k)
        if( ar[i]  in  hm)
            hm [ar[i]]  += 1
        else
            hm [ar[i]]  = 1
    print ( len(hm))
    // sliding  window.
    s = 1,   e = k.
    while (e < N)
    {
        // remove  ar[s-1]
        hm[ar[s-1]]  =  hm[ar[s-1]] -1 ;
        if ( hm[ar[s-1]) ==0)
            hm.remove ( ar[s-1]) .
        // add  ar[e]  to  hm.
        if ( ar[e]  in  hm)
            hm [ar[e]]  =  hm[ar[e]] +1
        else
            hm [ar [e]]  = 1
        print ( len(hm))
        e = e+1,  s = s+1 ,
    }
}
```

O(K).

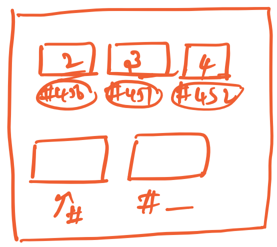
O (N-K+1)

TC: O(n).

SC: O(K)

Doubts.

ar = {2, 3, 4}.



'50'

52 48

[ ][ ]

25 → [11001]

'1' 4 '0'.

9 → [1001]

256 characters
@ $ *!

'A' → 65

s = '65'

53 52.

s[0], s[1].

[ ][ ]

'10'

[1 51 97]

4 → 51

'Aman456@gmail.com'.

456

'4' '5' '6'

0 → 255.

$a = 20.$

$a = [5, 16, 8].$

Aman , $[16, 20, 8, 4, 5]$

AAMN.

Doubt.

→ Ask in groups / whasapp, slack. → hints / videos.

→ Take TA's help.

→ Ask me.