Hashing-2

Content

→ Pair sum = K

→ Distinct elements in every window of len=K

Question 1

Given N array elements, check if there exists a pair (i,j) such that $a[i] + a[j] = K$ and $i != j$

$\hookrightarrow$ K is given

→ return true/false

eg
$$a[\,] = \begin{array}{cccccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ 8 & 9 & 1 & -2 & 4 & 5 & 11 & -6 & 7 & 5 \end{array}$$

$K=11 \Rightarrow a[4] + a[8] = 4+7 = 11$ True

$K=6 \Rightarrow a[2] + a[5] = 1+5 = 6$

$a[0] + a[3] = 8-2 = 6$ True

$K=22 \Rightarrow \underbrace{a[6] + a[6]}_{i==j\ \times} = 11+11 = 22$ False

Idea 1

Check all pairs sum = K

```
for ( i=0;  i<n;  ++i ) {
          a=a[i]      a+b=K      b=K-a= K-a[i]
    for ( j= i+1,  j<n;  ++j ) {
        if ( a[i] + a[j] ==K )  ⇒  if ( a[j]==b )
             return true
    }
}
return false
```

TC: $O(N^2)$

SC: $O(1)$

Idea2 : Use Hashset

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

K=11 ⇒ a+b = 11

a	b = K-a	check b is present in hashset or not ?
8	3	NO
9	2	NO
1	10	NO
-2	13	NO
4	7	YES { return true }

a+b = 5

a	b	present or not ?
8	-3	NO
9	-4	NO
1	4	YES { return true }

a + b = -4

a	b	present or not ?
8	-12	NO
9	-13	NO
1	-5	NO
-2	-2	YES { return true } X wrong

Basically if a = b then occurence of a should be more than 1.

Note: frequency of elements are important to know.

Idea3 : Hashmap

a[] = 8 9 -2 4 5 11 -6 7 5

hm = { <8,1> <9,1> <7,1> <-2,1> <4,1> <5,2> <11,1> <-6,1> }

a+b = 10

a	b	present or NOT ?
8	2	NO
9	1	NO
-2	12	NO
4	6	NO
5	5	if(a==b & freq(a) >1) { return true }

Code

```
bool pairSum ( a[] , K) {
    Hashmap< int, int> hm
    insert a[]  → hm        // TODO →O(N)

    for (i=0; i<n; ++i) {
        a= a[i] ,   b = K-a[i]
        if ( hm.search (b) == true ) {
            if ( a != b)
                return true
            if (hm[b] > 1)
                return true

        }
    }
    return false
}
```

TC: O(N)

SC: O (N)

Idea 4 : Use Hashset again

at ith index , hs will contain all elements from [0, i-1]

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

K = 22

a	b	hs	present or NOT ?
8	14	{}	NO
9	13	{8}	NO
5	17	{8,9}	NO
-2	24	{8,9,5}	NO
11	11	{8,9,5,-2}	NO → now code is working for a=b & freq(a)=1
		{8,9,5,-2,11}	

K=10

a	b	hs	present or NOT
8	2	{}	NO
9	1	{8}	NO
5	5	{8,9}	NO
-2	12	{8,9,5}	NO
11	-1	{8,9,5,-2}	NO
5	5	{8,9,5,-2,11}	YES { return true }

Code

```
bool pairSum ( a[] , K ) {
    Hashset<int> hs
    for ( i=0; i<n; ++i ) {
        a = a[i] ,  b = K - a[i]
        if ( hs.search(b) == true )
            return true
        hs.insert( a )
    }
    return false
}
```

TC: O(N)

SC: O(N)

Question 2

Given N elements, calculate no. of distinct elements in every subarray of size K.

eg a[10] =

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

K=4

subarrays	distinct elements
[0,3]	4
[1,4]	3

[2,5]	3
[3,6]	4
[4,7]	3
[5,8]	2
[6,9]	3

⇒ Ans

Idea1: for every subarray of len =K insert into hash set & find size.

Code

```
for (i=0; i<n-K; ++i) {          → (n-K+1) iterations
    Hashset<int> hs
    for (j=i; j<i+K; ++j) {      → K iterations
        hs.insert(a[j])
    }
    print(hs.size)
}
```

TC: $O((n-K+1)*K)$ ⇒ $O(N^2)$ SC: $O(K)$

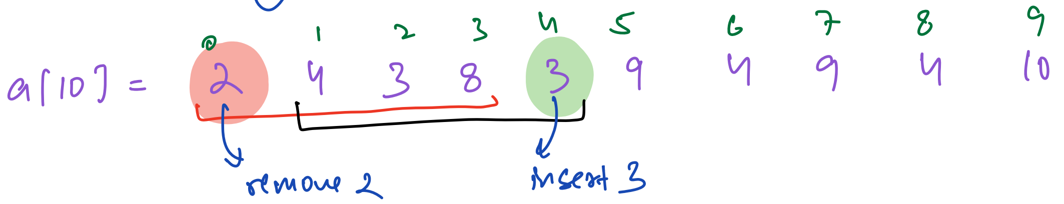
1. $K=n$ → TC: $O((n-n+1)*n) = O(N)$

2. $K=1$ → TC: $O((n-1+1)*1) = O(N)$

3. $K=\frac{N}{2}$ → TC: $O\left((n-\frac{n}{2}+1)*\frac{n}{2}\right) = O\left(\frac{N^2}{4}\right)$

$= O(N^2)$

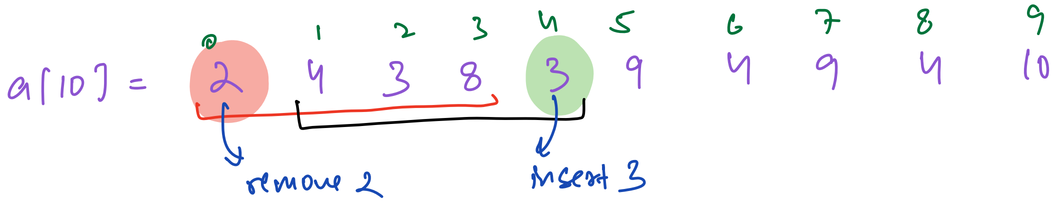
Idea2 : Sliding Window using Hashset



		Hashset	hs.size
[0,3]	→	{2,4,3,8}	4
[1,4]	remove a[0] → insert a[4]	{4,3,8}	3
[2,5]	remove a[1] → insert a[5]	{3,8,9}	3
[3,6]	remove a[2] → insert a[6]	{8,9,4}	3 ✗

In hashset we don't know the frequency of inserted elements. If we insert 2 times and remove 1 time, there should be 1 element in the hashset but we can't do it.

Idea 3 : Sliding window using Hashmap



s e		Hashmap	Size
[0, 3]	→	{<2,1> <4,1> <3,1> <8,1>}	4
(1,4)	$\xrightarrow[\text{add } a[4]]{\text{remove } a[0]}$	{ <4,1> <3,2> <8,1> }	3
[2,5]	$\xrightarrow[\text{add } a[5]]{\text{remow } a[1]}$	{ <3,2> <8,1> <9,1> }	3
[3,6]	$\xrightarrow[\text{add } a[6]]{\text{remove } a[2]}$	{ <3,1> <8,1> <9,1> <4,1> }	4

Code

```
def distinctCount( a[] , K ) {
    n= a.length
    Hashmap <int ,int> hm
    for (i=0; i<K ; ++i) {
        if( hm. search( a[i]) == true )
            hm[a[i]] ++
        else
            hm.insert ( {a[i],1} )
    }
    print (hm.size) → for first subarray [0,K-1]

    s=1, e=K
while (e<n) {
    // subarray : [s,e]  ->  remove a[s-1] , add a[e]
    hm[a[s-1]] --
    if(hm[a[s-1]] == 0)                  -> remove
        hm.remove (a[s-1])
    if( hm.search(a[e]) == true )
        hm[a[e]] ++                      -> insert
    else
        hm.insert ({a[e], 1})
    print(hm.size)
    s++, e++
  }
}
```

TC : O(N)

SC : O(K)