Today's Content

- → Pair Sum = K
- → Pair Difference = K
- → Distinct elements in every window of size = K
- → Subarray with sum = K

Q1: Given A[N], check if there exists a pair (i, j) s.t. $\underbrace{A[i] + A[j]}_{a + b} = \underbrace{K}_{K}$, i != j

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

K = 11 : ar[4] + ar[8] ⇒ 4 + 7 = 11 ⇒ YES

K = 6 : ar[2] + ar[5] ⇒ 1 + 5 } = 6 ⇒ YES
ar[0] + ar[3] ⇒ 8 - 2 }

K = 22 : ar[6] + ar[6] = 11 + 11 = yes × NO
↳ NO! i != j

idea: Check all pair sum = K , TC: O(N^2), SC: O(1)

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

idea2: Optimise using Hashset

* insert all elements into hashset

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

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

a+b = 11 (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	Yes {return true}

a+b = 5 (K=5)

a	b (K-a)	b is present in HS
8	-3	No
9	-4	No
1	4	Yes

a+b = -4 (K=-4)

a	b (K-a)	is b present in HS
8	-12	No
9	-13	No
1	-5	No
-2	-2	YES ↳ Error!

idea 3: Use hashmap

* insert all elements into hashmap & store their frequency.

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

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

a + b = 5

a	b (k-a)	is b present in map
8	-3	No
9	-4	No
1	4	4 is present, ⇒ return true. (↳ (a != b && b is present in map))

a + b = 10

a	b (k-a)	is b present in map
8	2	X
9	1	Yes → true.

ar[] = { 8 9 4 5 11 -6 7 5 -2 4 }

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

a + b = 10

a	b (K-a)	is b present in map
8	2	NO
9	1	NO
4	6	NO
5	5	YES

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

Pseudocode:

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

```
bool pairSum(int a[], int K) {
    Hashmap<int, int> hm
    Insert a[] into hm // Todo
    for(i=0; i<n; i++) {
        a = a[i], b = K-a
        if(a!=b && hm.search(b) == true) {
            return true
        }
        if(a==b && hm[a] > 1) {
            return true
        }
    }
    return false
}
```

idea 4 : Why does HS not work? ⇒ can contain duplicates.

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

K = 10

a	b (k-a)	HS	b is present in HS
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.	

Pseudo code

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

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

Q2: Given A[N], check if there exists a pair (i, j) s.t. $A[i] - A[j] = K$, $i != j$

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

$a[] = \{1, 3, 7, 9, 2, 5, 6\}$

$K = 2$

$a[1] - a[0]$ } 2
$a[3] - a[2]$ }

Prev Qn: $A[i] + A[j] = K$ $(i != j)$

or

$A[j] + A[i] = K$

fix A[i]: $a = A[i]$: $b = K - A[j]$

fix A[i]: $a = A[i]$: $b = K - A[j]$

In this Qn:

$A[i] - A[j] = K$

fix A[i]

$a = A[i]$, $\Rightarrow$ $b = A[j] + K$

$A[j] - A[i] = K$

fix A[i]

$a = A[i]$ $c = A[j] - K$

idea: Search for both b & c in idea 4 above

↳ if any is present

↳ return true.

(Todo)

Break: 8:08 am → 8:18 am

Q3: Given A[N]. Calculate & print the no. of distinct elements in subarray of size = K

	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 1: For every subarray of len = K Get No. of distinct elements.

TC: $O(N-K+1) \times K$

$K = N/2 \approx \left(N - \frac{N}{2} + 1\right)\left(\frac{N}{2}\right)$

$\approx O(N^2)$

SC: $O(K)$

idea 2: Optimise using Hashset

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

	remove	add	Hashset	HS.size
[0 3]			{2, 4, 3, 8}	4
[1 4]	a[0]	a[4]	{~~2~~, 4, 3, 8}	3
[2 5]	a[1]	a[5]	{~~4~~, 3, 8, 9}	3
[3 6]	a[2]	a[6]	{~~3~~, 8, 9, 4}	3 ↓ Error!

idea 3: Use hashmap

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

K = 4.

[0 3]

HashMap

{<2,1> <3,1>
<4,1> <8,1>}

HM. Size 4

s	e	remove s-1	add e	HashMap	HM. Size
				{<2,1> <3,1> <4,1> <8,1>}	4
[1 4]		a[0]	a[4]	{<2,0> <3,2> <4,1> <8,1>}	3
[2 5]		a[1]	a[5]	{<9,1> <3,2> <4,0> <8,1>}	3
[3 6]		a[2]	a[6]	{<9,1> <3,1> <8,1> <4,1>}	4
[4 7]		a[3]	a[7]	{<9,2> <3,1> <8,0> <4,1>}	3
[5 8]		a[4]	a[8]	{<9,2> <3,0> <4,1>}	2
[6 9]		a[5]	a[9]	{<9,1> <4,2> <10,1>}	3

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

K

N-K

TC: O(N)

SC: O(K)

Qu: Check if there exists a subarray with sum = K

	0	1	2	3	4	5	6	7	8	9
a[] =	2	2	1	-3	4	3	1	-2	-3	2
pf[] =	2	4	5	2	6	9	10	8	5	7

sum([i j]) = K

↓

pf[j] - pf[i-1] = K

↑ fix this

a = pf[i-1] , b = pf[j] - K

pf[i-1] = pf[j] or search for duplicates.

sum = 0 2
K = 0

K = 3

	0	1	2	3	4	5	6	7	8	9
a[] =	2	2	1	-3	4	3	1	-2	-3	2
pf[] =	2	4	5	2	6	9	10	8	5	7

a	b (a-K)
2	-1
4	1
5	2
2	-1
6	3
9	6

HS

{ } → NO

↓

{ 2 } → NO

↓

{ 2, 4 } → YES! pf[2] - pf[0] = 5 - 2 = [1 2]

↓

{ 2, 4, 5 } → NO

↓

{ 2, 4, 5 } → NO

↓

{ 2, 4, 5, 6 } → YES pf[5] - pf[4] = 9 - 6 = 3 = [5 5]

↓

10 7 {2,4,5,6,9} → No

8 5 {2,4,5,6,9,10} → YES

5 2 {2,4,5,6,9,10,8} → YES

7 4 {2,4,5,6,9,10,8} → YES

pf[7] - pf[2]
8 - 5 = 3
[3 7]

pf[8] - pf[3]
5 - 2 = 3
[4 8]

pf[8] - pf[0]
5 - 2 = 3
[1 8]

pf[9] - pf[1]
7 - 4 = 3
[2 9]

pf[i-1] pf[j]

sum[0 i-1] sum[0 j]

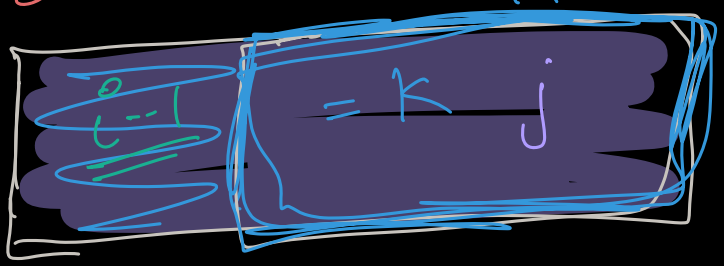
pf[i-1] = pf[j] - k [i..j]

pf[j] = pf[i-1] + k

[0...j] [0...i-1] + k

Rhs

-k



```
bool subarrWithSumK(int a[], int k) {
    Hashset<int> hs;
    int pf[n]
    // Create pf array → Todo
    sum = 0
    for(i=0; i<n; i++) {
        int a = pf[i] // sum = sum+a[i]
        int b = a-k // b = sum-k
        // Edge case
        if(pf[i] == k) { //(sum == k)
            return true
        }
        if(hs.search(b) == true) {
            return true
        }
        hs.insert(a) // hs.insert(sum)
    }
}
```

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

Note: → `pf[i] == k` handles the sum([0...i]) = k case

`hs.search(b) == true` handles the sum([i...j]) case