friday <- Contest -> Recursion, Maths & Hashing.

<- Pair Sum K ->

Ques

Given arr[N] and K, check if there exists a pair(i, j) such that,

```
arr[i] + arr[j] == K && i != j
```

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

K = 6, arr[2] & arr[5] -> True

K = 22, NO -> False

K = 8, arr[4] & arr[8] -> True

Brute force :- Check all pairs. arr[4]

for i -> 0 to n-1

for j -> i+1 to n-1

. . .

T.C -> O(n²)

S.C -> O(1).

	0	1	2	3
0	(0,0)	(0,1)	(0,2)	(0,3)
1	(1,0)	(1,1)	(1,2)	(1,3)
2	(2,0)	(2,1)	(2,2)	(2,3)
3	(3,0)	(3,1)	(3,2)	(3,3)

idea 2 :- using hash set

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

k = 8,

arr[i],	arr[j] = k - arr[i]
8	0
9	-1
1	7
-2	10
4	4

k = 9,

arr[i],	arr[j]	
8	1	✓

k = 20

arr[i],	arr[j]	
8	12	✗
9	11	✓

8, 9, 1, -2,

4, 5, 11, -6

```
for i -> 0 to n-1
    other = k - arr[i];
    if (hs.contain(other)){
        return True
    }
}
return false;
```

-: Edge Case :-

arr[] = { 5, 7, 9, 2, 3 }

k = 4

arr[i]	arr[j] = k - arr[i]
5	-1
7	-3
9	-5
2	2

$\boxed{5, 7, 9, 2, 3}$

return True; (wrong)

idea: Hashset with optimization:-

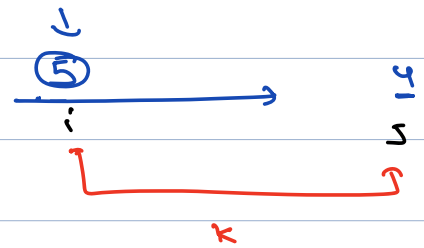
Index	0	1	2	3	4	5	6	7	8
Array	8	9	2	-2	4	5	11	-6	4

k = 9

arr[i]	arr[j] = k - arr[i]
8	1
9	0
2	7
-2	11
4	5
5	4

$\boxed{8, 9, 2, -2, 4}$

→ return True;



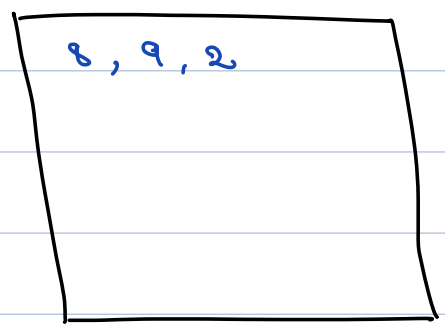
5 4

4 5

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

K = 4

arr[i]	arr[j] = k - arr[i]
8	-4
9	-5
2	2
-2	6
2	2




```
Hashset <int> hs;

for (i=0; i<n; i++) {
    other = k - arr[i];
    if (hs.check (other)) {
        return True;
    }
    hs.add (arr[i]);
}
```

Counting all pairs

0	1	2	3	4	5

arr [3, 5, 1, 2, 1, 2] , k = 3 Ans → 4

Given an arr[n], count number of pairs such that

```
arr[i] + arr[j] = K && i != j
```

k = 10,

	0	1	2	3	4	5	6	7
arr[8] =	2	5	2	5	8	5	2	8

Pairs Ans → 9

(0, 4) (1, 3) (2, 4) (3, 5) (6, 7)

(0, 7) (1, 5) (2, 7) (4, 6)

arr[i]	comp
2	8 → 3.

	0	1	2	3	4	5	6	7
arr[8] =	2	5	2	5	8	5	2	8
	+0	+0	+0	+1	+2	+2	+1	+3

K = 10,

arr[i],	K - arr[i]
2	8
5	5
2	8
5	5
8	2
5	5
2	8
8	2

2 → ~~1~~ ~~2~~ 3
5 → ~~1~~ ~~2~~ 3
8 → ~~1~~ 2

```
int ans=0;
hashmap <int,int> hm;
for (i=0; i<n; i++) {
    other= k-arr[i];
    if (hm.check(other)) {
        ans+= hm.get(other);
    }
    if (hm.check(arr[i])==false) {
        hm(arr[i], 1);
    }
    else {
        hm[arr[i]]+=1
    }
}
```

T.C → O(n), S.C → O(n).

Subarrays with sum = k.

Ques) Given an array arr[n] check if there exists a subarray with sum = K

Index	0	1	2	3	4	5	6	7	8
arr[7]	2	3	9	-4	1	5	6	2	5

K = 11, (5, 6) or (2, 3, 9, -4, 1)

K = 10, (2, 3, 9, -4)

K = 15, (-4, 1, 5, 6, 2, 5)

-: Brute force :-

check every subarray sum.

$O(n^2)$.

- : Optimized idea :-

	0	1	2	3	4	5	6	7	8
arr[] =	2	3	9	-4	1	5	6	2	5
pf[] =	2	5	14	10	11	16	22	24	29

K = 12

pf[i] = sum (0 to i)

K = 13

a − b = k

a-k

	0	1	2	3	4	5	6	7	8
arr[] =	2	3	9	-4	1	5	6	2	5
pf[] =	2	5	14	10	11	16	22	24	29

2 14

k = 12.

a,	a-k
2	-10
5	-7
14	2

HS

2, 5

pf[R] - pf[L] = k.

L+1 to R = k

	0	1	2	3	4
arr[] =	2,	3,	9,	-4,	1 3
pf = 0,	2	5	14	10	11

11

k=11

a	a-k
2	-9
5	-6
14	3
10	-1
11	0

HS

0, 2, 5, 14, 10, 11

using pf :-

sum(L to R) = pf(R) - pf(L-1) (L != 0)

pf(R) (L == 0)

```
pf = 0;
HashSet<int> hs;

hs.add(0);
for (i=0; i<n; i++) {
    pf = pf + arr[i];
    if (hs.contains(pf-k)) {
        return True;
    } else
    hs.add(pf);
}
```

T.C → O(n)
S.C → O(n)

30,

45, 15

15 → 3

Ques Distinct elements in every window of size k.

arr[] = {1, 2, 1, 3, 4, 2, 3}

k = 4 {3, 4, 4, 3}

① using hashset

arr[] = {1, 2, 1, 3, 4, 2, 3}

k = 4 , {3, 4

1, 3, 4, 2

starting point	endpoint	len
s	n-1	k

$\Rightarrow n - x - s + 1 = k \Rightarrow s = n - k$

```
for (i=0; i<=n-k; i++) {
    Hashset<int> hs;
    for (j=0; j<k; j++) {
        hs.add(arr[i+j]);
    }
    print(hs.size());
}
```

T.C → $O((n-k+1) * k)$

when $k = 1$, $O(n)$

when $k = n$, $O(n)$

when $k = n/2$, $(n - \frac{n}{2} + 1) * \frac{n}{2} \Rightarrow O(n^2)$

idea 2 :- using hash map

arr[] = {1, 2, 1, 3, 4, 2, 3}

k = 4 , {3, 4, 4, 3}

0 1 k-1 e

~~1 → 2 → 0~~
2 → 1
3 → ~~1~~ 2
4 → 1

```
HashMap<int, int> hm;
    // [0, k-1]
    for (i=0; i<k; i++) {
        if (hm.contains(arr[i])) {
            hm[arr[i]]+=1;
        }
        else {
            hm[arr[i]] = 1
        }
    }

    s=1, e=k      // next window
    while (e<n) {
        hm[arr[s-1]] -=1;
        if (hm[arr[s-1]] == 0) {
            hm.remove(arr[s-1])
        }
```

T.C → O(n)
S.C → O(k).

```
if (hm.contains (arr[e])) {
        hm[arr[e]]+ = 1;
}
else {
        hm[arr[e]] = 1
}
        Print (hm.size ());
s++;
e++;
}
```

← Bubble Sort →

∴ unique Elements :-

1 2 ~~2~~ 3 2 4 . . .