Today's Quote.

Today's content

→ pair sum = K

→ pair difference = K {idea}

→ Distinct elements in every window of len = K.

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

arr → { 8 9 1 -2 4 5 11 -6 7 5 }
indices: 0 1 2 3 4 5 6 7 8 9

K = 11 : YES (4,8)

K = 6 : YES (0,3) (2,5)

K = 22 : NO.

ideas → Consider all the pairs & check their sum == K.

pseudo-code :

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

T.C → O(N²)
S.C → O(1)

a + b = K.
⇒ b = K - a

arr[i] + arr[j] = K
a + b = K

arr[] → { 8 9 1 -2 4 5 11 -6 7 5 }
0 1 2 3 4 5 6 7 8 9

idea 2. → optimisation using Hashset.

↳ insert all the element in hashset.

hs → { 8, 9, 1, -2, 4, 5, 11, -6, 7 }

K = 11.

a.	b = (K - a)	b is present or not?
8	3	No
9	2	No
1	10	No
-2	13	No
4	7	YES {return true}

K = 5.

a.	b = (K - a)	b is present or not?
8	-3	No
9	-4	No
1	4	YES {return true}

K = -4.

a.	b = (K - a)	b is present or not?
8	-12	No
9	-13	No
1	-5	No
-2	-2	YES {return true}

↳ Wrong ans. # No. of -2 in arr[] = 1., ans should be false.

Idea-3. optimisation using HashMap:

→ Insert all elements in hashmap.

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

hm →

8 → 1	-6 → 1
9 → 1	7 → 1
4 → 2	-2 → 1
5 → 2	
11 → 1	

Key → element
value → frequency

a+b=10

a.	b = (K-a)	b is present in hm or not?
8	2	NO
9	1	NO
4	6	NO
5	5	if (a == b && freq(a) > 1) return 1

a+b=22.

a.	b = (K-a)	b is present in hm or not?
8	14	No
9	13	NO
4	18	NO
5	17	No
11	11	if (a == b && freq(a) > 1)

:

check further

pseudo-code.

```
boolean pairSum( arr, N, K) {
    HashMap < int, int > hm
    // insert all elements in hm [TODO]
    for( i = 0 ; i < N ; i++){
        a = arr[i] , b = K-a
        if ( a == b && hm[a] > 1 ) {return true }
        if ( a != b && hm.search(b) == true ) {return true}
    }
    return false
}
```

T.C → O(N)
S.C → O(N)

idea-4 Optimisation using Hashset

→ At ith index, hs will contain unique elements upto $(i-1)^{th}$ idx.

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

K=22.

a	b = (K-a)	Hashset	is b present ?
8	14	{ - }	NO
9	13	{ 8 }	NO
4	18	{ 8, 9 }	NO
5	17	{ 8, 9, 4 }	NO
[11	11	{ 8, 9, 4, 5 }	NO]

Now, Code is working for this edge Case.

K=10

a.	b.	hs.	is b present ?
8	2	{ }	NO
9	1	{ 8 }	NO
4	6	{ 8, 9 }	NO
5	5	{ 8, 9, 4 }	NO
11	-1	{ 8, 9, 4, 5 }	NO
-6	16	{ 8, 9, 4, 5, 11 }	NO
7	3	{ 8, 9, 4, 5, 11, -6 }	NO
5	5	{ 8, 9, 4, 5, 11, -6, 7 }	YES, return true.

final pseudo-code.

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

T.C → O(N)
S.C → O(N)

[a + b = K]
↳ [a - b = K]
⇒ [a - K = b]

Q) Given N array elements, check if there exists a pair (i,j) such that their difference = K & i != j

arr → { 5, 10, 3, 2, 50, 80 }, K = 78.
(indices: 5 → 0, 10 → 1, 3 → 2, 2 → 3, 50 → 4, 80 → 5)

K = 78.

a.	b. = (a-K)	hs.	is b present?
5	-73	{ }	No
10	-68	{ 5 }	No
3	-75	{ 5, 10 }	No
2	-76	{ 5, 10, 3 }	No
50	-28	{ 5, 10, 3, 2 }	No
80	2	{ 5, 10, 3, 2, 50 }	YES. (return true)

arr → { 5, 10, 3, 80, 50, 2 }, K = 78
(indices: 5 → 0, 10 → 1, 3 → 2, 80 → 3, 50 → 4, 2 → 5)

a.	b. = (a-K) || (a+K)	hs.	is b present?
5	-73 , 83	{ }	NO
10	-68 , 88	{ 5 }	NO
3	-75 , 81	{ 5, 10 }	NO
80	2 , 158	{ 5, 10, 3 }	NO
50	-28 , 128	{ 5, 10, 3, 80 }	NO
2	-76 , 80	{ 5, 10, 3, 80, 50 }	YES (return true)

$$\begin{bmatrix} a+b = K \\ b = K-a \end{bmatrix}$$

Difference of two no.'s is K.

$a - b = K$

$b = a - K$

$b - a = K$

$b = a + K$

pseudo-code.

```
boolean pairDiff(arr, N, K) {
    HashSet<int> hs;
    for(i = 0; i < N; i++){
        a = arr[i], // b = K-a or K+a
        if (hs.search(K-a) || hs.search(K+a)){
            return true
        }
        hs.insert(arr[i])
    }
    return false;
}
```

T.C → O(N)

S.C → O(N)

Q: Given N elements. Calculate no. of distinct elements in every subarray of size K.

Eg: arr[10] : { 2 4 3 8 3 9 4 9 4 10 }
0 1 2 3 4 5 6 7 8 9

K = 4.

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

Idea: for every sub.array of len = K, insert elements into hashset & get no. of distinct elements.

TC : $(N-K+1) \cdot K = O(N^2)$, S.C → O(K)

↳ K=1 → $(N-1+1) \cdot 1 = N$

K=N → $(N-N+1) \cdot N = N$

K = N/2 → $(N - \frac{N}{2} + 1) \cdot \frac{N}{2} \approx \frac{N^2}{4} \rightarrow O(N^2)$

optimisation → Sliding Window + hashset.

arr[ ] → { 2 4 3 8 3 9 4 9 4 10 } , K = 4.
0 1 2 3 4 5 6 7 8 9

hashset.

no. of distinct element = hs.size()

[0-3] → { 2 4 3 8 } 4

	remove	add		
[1-4]	arr[0]	arr[4]	{ ~~4~~ 3 8 }	3
[2-5]	arr[1]	arr[5]	{ ~~3~~ 8 9 }	3
[3-6]	arr[2]	arr[6]	{ 8 9 4 }	3 . {it is wrong}

Note → frequency of every element is also important. {HashMap}

Idea-3. Optimisation using HashMap.

arr → { 2 4 3 8 3 9 4 9 4 10 } , K = 4

0 1 2 3 4 5 6 7 8 9

i: 0 to 9

	remove	add	HashMap	size
0-3.			2→1, 3→1, 4→1, 8→1	4
1-4.	arr[0] (arr[i-1])	arr[4] (arr[i])	4→1, 8→1, 3→2	3
2-5.	arr[1]	arr[5]	3→2, 9→1, 8→1	3
3-6.	arr[2]	arr[6]	3→1, 9→1, 8→1, 4→1	4
4-7	arr[3]	arr[7]	3→1, 4→1, 9→2	3
5-8	arr[4]	arr[8]	9→2, 4→2	2
6-9	arr[5]	arr[9]	9→1, 10→1, 4→2	3

final pseudo code.

```
void distinctElements ( arr, N, K) {
    HashMap < int, int > hm;
    for( i=0 ; i < K ; i++){
        if ( hm.search (arr[i]) == true){
            hm[arr[i]] += 1
        }
        else {
            hm.insert(arr[i], 1)
        }
    }
    print (hm.size())        // no. of distinct element in 1st subarray of size K

    s=1 , e=K
    while ( e < N ){
        // get subarray from s to e.      remove arr[s-1], add arr[e]
        hm [arr[s-1]] -= 1
        if ( hm [arr[s-1]] == 0){
            hm.remove ( arr [s-1])
        }
        if ( hm.search (arr[e]) == true){
            hm ( arr[e] ] += 1
        }
        else {
            hm.insert ( arr[e], 1)
        }
        print ( hm.size() )
        s += 1 , e += 1
    }
}
```

K

N-K

T.C → O(N)
S.C → O(K)

Doubts. →

Christmas tree.

A → { 5 3 8 11 4 9 6 }

B → { 2 3 4 6 5 3 8 }

$p < q < r$ $\quad A_p < A_q < A_r$

cost → $B_p + B_q + B_r$.

B.f. approach int minCost → ∞

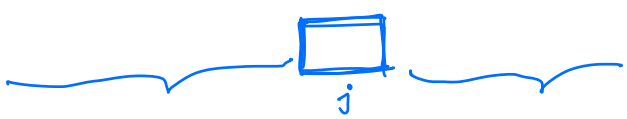
$i < j < k$

$A_i < A_j < A_k$

```
for( i = 0 ; i < N ; i++){
    for( j = i+1; j < N ; j++){
        for( k = j+1 ; k < N ; k++){
            if ( A[i] < A[j] && A[j] < A[k]){
                minCost = min( minCost , B[i]+B[j]+B[k]);
            }
        }
    }
}
```

T.C → $O(N^3)$

optimise $\quad A_i < A_j < A_k$.



observations

→ Consider every element as the middle element

→ A_i will be present on its left's & that should be less than middle element.

→ A_K will be present on its right's & that should be greater than middle element.

A → { 5 (0) 3 (1) 8 (2) 11 (3) 4 (4) 9 (5) 6 (6) }

B → { 2 (0) 3 (1) 4 (2) 6 (3) 5 (4) 3 (5) 8 (6) }

```
for( j = 1 ; j < N-1 ; j++){
    leftcost = ∞
    for( i = 0 ; i < j ; i++){
        if ( arr[i] < arr[j] && leftcost < B[i]){
            leftcost = B[i]
        }
    }
    rightcost = ∞
    for( K = j+1 ; K < N ; K++){
        if (arr[K] > arr[j] && rightcost < B[K])
            rightcost = B[K]
    }
```

```
    minCost = Min(minCost, leftCost + B[j] + rightCost)
}
return minCost;
```

T.C → O(N²)

S.C → O(1)