

Q18. Count pairs with given sum.

i/b + {1,5,7,-13, k=6 0/b - 2

Brute force Considerale the possible paires by running 2 nested loops and if we find a pair having sum equal to the given sum, then simply increase the count of the count Pairs. At the end simply return the count Pairs.

Time complexity = 0(n2)
Space complexity = 0(1)

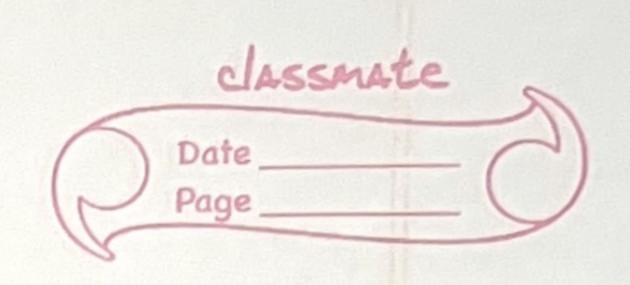
Optimal solution + Dry run 1) Create a map in which the frequency of elements will be stored.

> 5%

7:

2) Traverse the away now and find whether k-awr[i] is present in the away & this we can get to know about from the map.

V/ index = 0 m[6-1] = m[5] = 170 and hence count ++. Here count = 1.



index = 1

m[6-5]=m[I]=1>0 and hence again count ++.

Here count = 2.

// index = 2

m [6-7] = m [-1] = 1 > 0 and hence again Count ++.

Here count = 3.

index = 3

m[6-(-1)] = m[7] = 1 >0 & hence again Count ++.

Here count = 4.

But here we have considered the Bair twice.

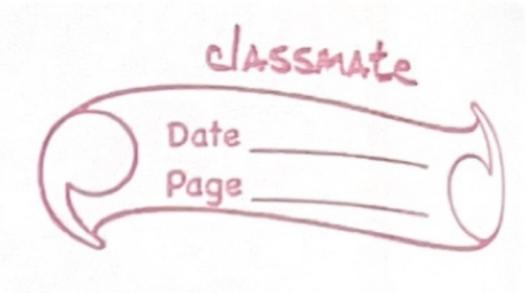
(1,5) and (5,1) => 2 pairs but it should be considered only once. Hence count/2 will be the answer.

Note -> Here we have to make sure that if k-aur[i] == aux [i], then decrement count as we don't have to consider (avr[i], avr[i]) pair.

int get Paires (int avr. [], int n, int k) {

// Create map to store frequency unordered_map <int, int) mi

// Store the frequency of elements in mas for (int i = 0 ; i < n; i++) { m [ovu((i)) ++;



```
// Initially count Pairs = 0
     int count Paire = 0 j
     for (int i=0; l'an; l'++) {//Element present
           if (m [K-avr[i]] !=0){
              count Pairs + = m [K-arr [i]];
         // arr(i), avr(i) not to be considered
         if (k-avr (i) = = avr [i]) {
                count Pairs -- i// Decrement
  3 Count bairs
     int ans = count Paires /2 i // 2 times pair
                      have been considered
     return ansi
    Alt distribution avoid and aread the
Time complexity = O(n)

Space complexity = O(n)
```