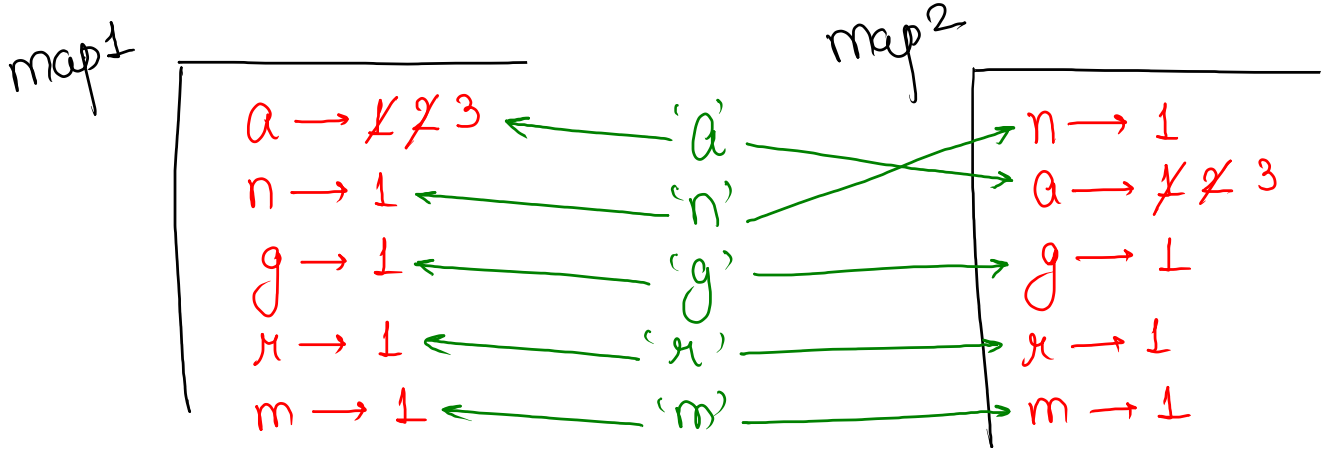
Valid Anagram 5

→ **Anagram :-** when 2 strings are have same character, appearing same no. of time

idea

s = "anagram"

t = "nagaram"



psudo code

1) declare map1
2) traverse in first string
   2.1) check if curr char is present in map
        then find oldfreq
        and put curr element back with oldfreq + 1
   2.2) else
        put curr element with freq 1

3) declare map2
4) traverse in second string
   4.1) check if curr char is present in map
        then find oldfreq
        and put curr element back with oldfreq + 1
   4.2) else
        put curr element with freq 1

5) traverse in map1
   if check if curr key is present in map2
        and freq of both should be same
   else
        return false
6) return true.

Code

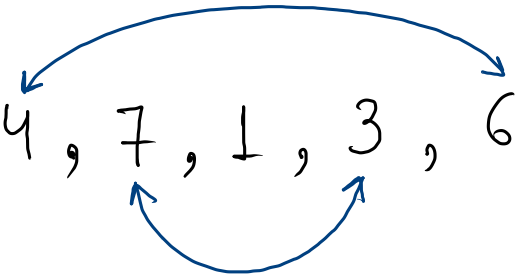
```java
public static boolean checkAnagram(String s, String t) {
    HashMap<Character, Integer> map1 = new HashMap<>();
    for (char c : s.toCharArray()) {
        if ( map1.containsKey(c) ) {
            int oldFreq = map1.get(c);
            map1.put(c, oldFreq + 1);
        } else {
            map1.put(c, 1);
        }
    }

    HashMap<Character, Integer> map2 = new HashMap<>();
    for (char c : t.toCharArray()) {
        if ( map2.containsKey(c) ) {
            int oldFreq = map2.get(c);
            map2.put(c, oldFreq + 1);
        } else {
            map2.put(c, 1);
        }
    }

    for (Map.Entry<Character, Integer> e : map1.entrySet()) {
        char ch = e.getKey();
        int freq = e.getValue();
        if ( !map2.containsKey(ch) ) {
            return false;
        }
        if ( freq != map2.get(ch) ) {
            return false;
        }
    }
    return true;
}
```

T.C = O(n+m)

S.C = O(n+m)

Max Number of K-Sum Pairs

arr = [4, 7, 1, 3, 6, 8, 5] , K = 10



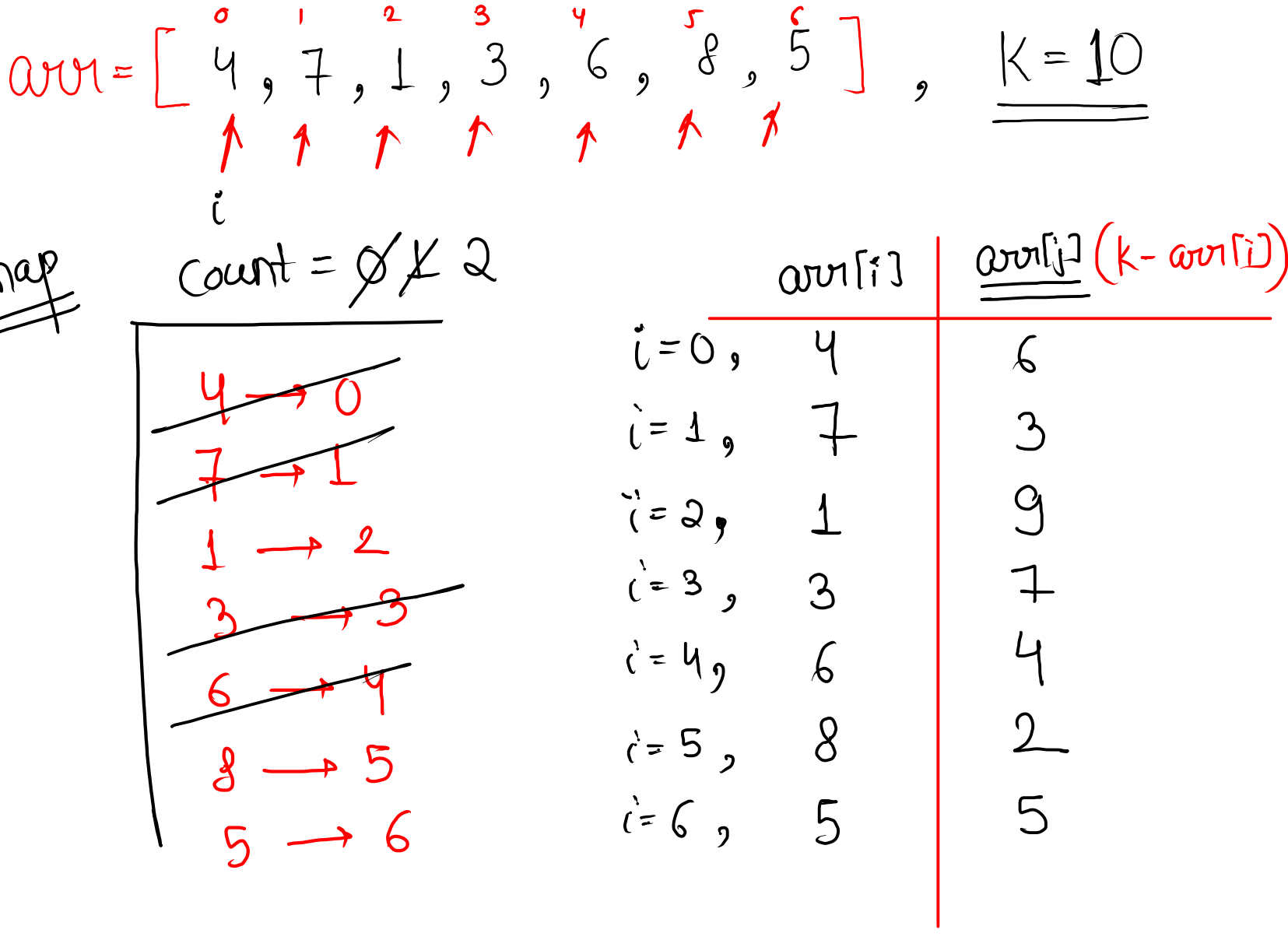
Count = 2

(Note:- no repeated elements)

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

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

Solution:- Consider every element arr as arr[i], and try to find arr[j] using hashmap.


arr = [4 , 7 , 1 , 3 , 6 , 8 , 5] , K = 10
0 1 2 3 4 5 6
i
map
count = ~~0~~ ~~1~~ 2
4 → 0
7 → 1
1 → 2
3 → 3
6 → 4
8 → 5
5 → 6
arr[i] | arr[j] (k - arr[i])
i = 0 , 4 | 6
i = 1 , 7 | 3
i = 2 , 1 | 9
i = 3 , 3 | 7
i = 4 , 6 | 4
i = 5 , 8 | 2
i = 6 , 5 | 5

code

```java
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    int n = scn.nextInt();
    int[] arr = new int[n];
    for (int i = 0; i < n; i++) {
        arr[i] = scn.nextInt();
    }
    int k = scn.nextInt();
    int ans = maxKpairs(arr, n, k);
    System.out.println(ans);
}
public static int maxKpairs(int[] arr, int n, int k) {
    HashMap<Integer, Integer> map = new HashMap<>();
    for (int i = 0; i < n; i++) {
        map.put( arr[i], i );
    }

    int count = 0;
    for (int i = 0; i < n; i++) {
        int num1 = arr[i];
        int num2 = k - num1;
        if ( map.containsKey( num2 ) ) {
            if ( i != map.get(num2) ) {
                count++;
                map.remove(num1);
                map.remove(num2);
            }
        }
    }
    return count;
}
```

T.C = O(n)

S.C = O(n)

arr = [7 , 2 , 3 , -2 , 5 , 12] , k = 10

(indices: 0, 1, 2, 3, 4, 5)

map

~~7 → 0~~

2 → 1

~~3 → 2~~

~~-2 → 3~~

5 → 4

~~12 → 5~~

Count = ~~0~~ ~~1~~ 2

i = 0, (7, 3) ✓

i = 1, (2, 8) ✗

i = 2, (3, 7) ✗

i = 3, (-2, 12) ✓

i = 4, (5, 5) ✗

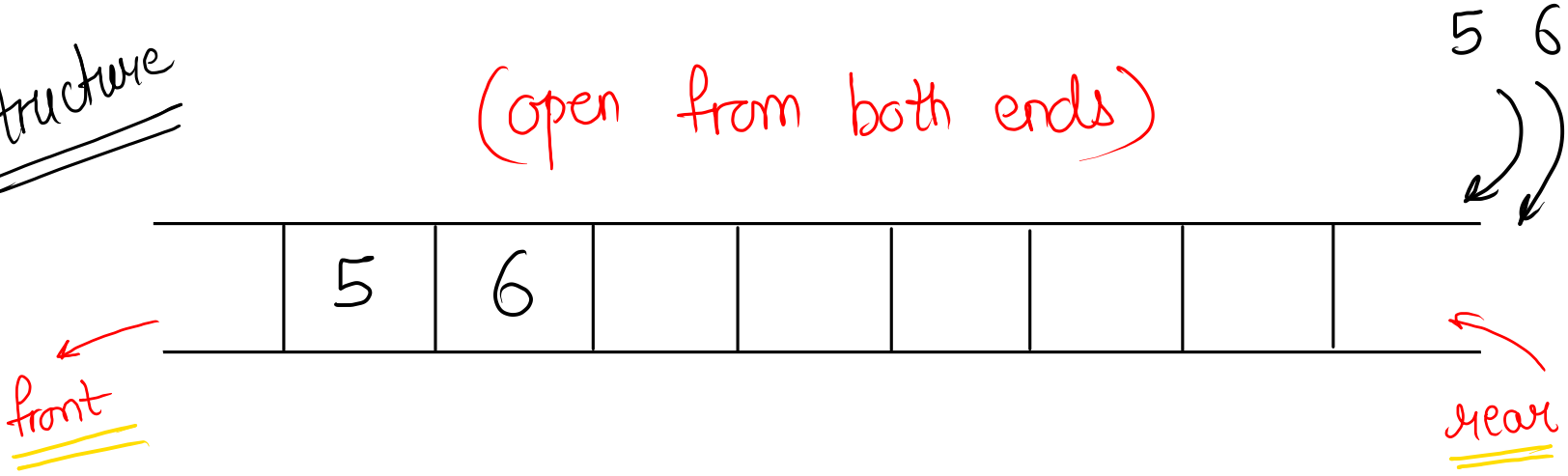
i = 5, (12, -2) ✗

⇒ Queue [FIFO :- first in first out]

(dynamic) → doesn't have any indexing

structure

(open from both ends)



Note:- we can add elements from rear end & remove elements from front end only

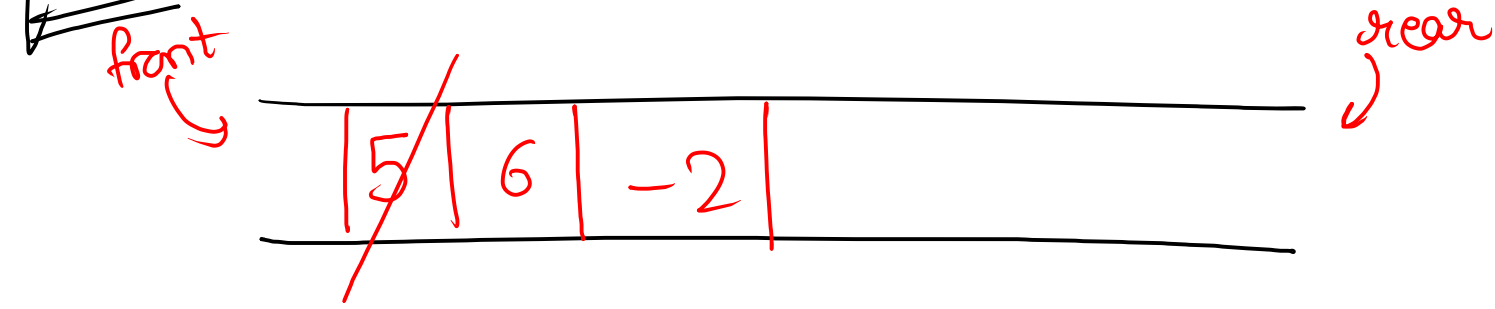
Syntex

```
Queue<DataType> que = new LinkedList<>()
Queue<DataType> que = new ArrayList<>()
```

inbuilt

```
que.add(value); // to insert value from rear end

{ que.remove(); // to delete and return element
  que.poll();   // in front of queue

que.peek();   // return front element from queue without removing it

que.size() / que.isEmpty()   } evergreen
```

Que




```
que.add(5);
que.add(6);
que.add(-2);
que.peek()  ⟶ 5
que.poll()  ⟶ 5
que.poll()  ⟶ 6
que.isEmpty()  ⟶ false.
```