QN Link : <https://leetcode.com/problems/max-number-of-k-sum-pairs/>

Step 1 : Put the all occurrence of the array in the haspMap.

Step 2 : For each step, the req will be req – nums[i].

Step 3 : Chack wheter req and nums[i] is equal because it refers the same element , so we need to subract the count of same element in the map.

Step 4 : If not , check whether , nums[i] & req is present in the array , if so then get the min of both occurrence and add it to the count.

Step 5 : Whenever removing the element from the map , also reduce the count and if it reached zero then remove it.

class Solution {

    public int maxOperations(int[] nums, int k) {

        int n = nums.length;

        int cnt = 0;

        Map<Integer , Integer> map = new HashMap<>();

        for(int i : nums){

            map.put(i , map.getOrDefault(i , 0) + 1);

        }

        for(int i = 0; i < n ; i++){

            int req = k - nums[i];

            if(req != nums[i] && map.containsKey(req) && map.containsKey(nums[i])){

                int x = map.get(req);

                int y = map.get(nums[i]);

                int min = Math.min(x , y);

                cnt += min;

                map.put(nums[i] , map.getOrDefault(nums[i] , 0) - min);

                map.put(req , map.getOrDefault(req , 0) - min);

                if(map.get(req) == 0) map.remove(req);

                if(map.get(nums[i]) == 0) map.remove(nums[i]);

            }else if(req == nums[i] && map.containsKey(req) && map.get(req) >= 2){

                cnt += 1;

                map.put(req , map.getOrDefault(req , 0) - 2);

                if(map.get(req) == 0) map.remove(req);

            }

        }

        return cnt;

    }

}