Prerequesite Problem : <https://leetcode.com/problems/max-number-of-k-sum-pairs/description/>

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;

}

}

Question Link : <https://www.desiqna.in/15068/google-oa-swe-intern-july-2023-solution-by-kumar-k>

Question Summary :

Find the maximum number of pairs whose sum is K but we dono the value of K.

Observation :

* The value of K should be , the pair sum of any of the element 🡪 ( O ( N ^ 2) )
* Run the LC code on each pair sum

import java.util.HashMap;

import java.util.Map;

class Solution {

     public int solution(int [] nums){

        int rose =  -1;

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

        for(int i ; nums){

            for(int j : nums){

                map.put(i + j , 0);

            }

        }

        for( int i : map.keySet()){

            int ans = maxOperations(nums , i);

            rose = Math.max(ans , rose);

        }

     }

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

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

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

        int i = 0;

        int n = nums.length;

        while (i < n) {

            kk.put(nums[i], kk.getOrDefault(nums[i], 0) + 1);

            i = i + 1;

        }

        int answer = 0;

        for (i = 0; i < n; i++) {

            int number = nums[i];

            int partner = k - number;

            if (gg.getOrDefault(number, 0) == 0 && gg.getOrDefault(partner, 0) == 0) {

                if (number == partner) {

                    int yy = kk.get(number);

                    answer = answer + (yy / 2);

                } else {

                    answer = answer + Math.min(kk.getOrDefault(number, 0), kk.getOrDefault(partner, 0));

                }

            }

            gg.put(number, 1);

            gg.put(partner, 1);

        }

        return answer;

    }

}

Tc : O ( N ^ 3)