QN Link : <https://www.desiqna.in/13267/microsoft-coding-oa-sde-1-may-3-2023>

Step 1 : Find the sum of each number and and store it in an list in the map

Step 2 : Iterate over each key in the hashMap and find the largest and second largest element from the hashMAp.

Step 3 : if the hashMap of size one , then it is not eligible to perform the opration

Step 4 : Find the maximum sum of those pairs.

Constarins :

1 <= n <= 2 \* 10 ^ 6

1 <= nums[i] <= 10 ^ 9

class Solution {

        private long sumOfDigits(long n){

        long sum = 0;

        while(n != 0){

            sum += (n % 10);

            n/= 10;

        }

        return sum ;

    }

    public int maxPair(long [] nums){

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

        int n = nums.length;

        int ans = -1;

        for(int i = 0 ; i < n ; i++){

            long sum = sumOfDigits(nums[i]);

            if(map.containsKey(sum)){

                long currentSum = nums[i] + map.get(sum);

                //Get the MAX of ans and current pais sum

                ans = Math.max(ans , currentSum);

                //Always put the max number for a sum in the Map --> TO avoid looping

                map.put(sum , Math.max(map.get(sum) , nums[i]));

            }else{

                map.put(sum , nums[i]);

            }

        }

        return ans;

    }

}

Time complexcity : O(N \* 10) ~ O (N) 10 for calculating sum of each digit

Space Complexcity : O (1)

Because , the maximum sum of the inout would be 81

So it will maximum generate 81 keys , so the space complexcity would be 1