2342. Max Sum of a Pair With Equal Sum of Digits

You are given a **0-indexed** array `nums` consisting of **positive** integers. You can choose two indices `i` and `j`, such that `i != j`, and the sum of digits of the number `nums[i]` is equal to that of `nums[j]`.

Return *the* ***maximum*** *value of* `nums[i] + nums[j]` *that you can obtain over all possible indices* `i` *and* `j` *that satisfy the conditions.*

Example 1:

```
Input: nums = [18,43,36,13,7]
Output: 54
Explanation: The pairs (i, j) that satisfy the conditions are:
- (0, 2), both numbers have a sum of digits equal to 9, and their sum is 18 + 36 = 54.
- (1, 4), both numbers have a sum of digits equal to 7, and their sum is 43 + 7 = 50.
So the maximum sum that we can obtain is 54.
```

Example 2:

```
Input: nums = [10,12,19,14]
Output: -1
Explanation: There are no two numbers that satisfy the conditions, so we return -1.
```

Constraints:

- `1 <= nums.length <=` 10^5
- `1 <= nums[i] <=` 10^9

Overview

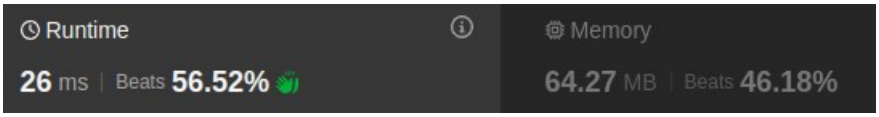
We are given an array `nums` of positive integers. Our goal is to find the largest possible sum of two distinct elements, `nums[i]` and `nums[j]`, where both numbers have the same digit sum. If no such pair exists, we return `-1`.

Observation

Observe that we can divide the numbers into groups, where all numbers with the same digit sum belong to the same group. The two largest numbers in each group will always form the pair with the greatest sum for that group.

2342. Max Sum of a Pair With Equal Sum of Digits

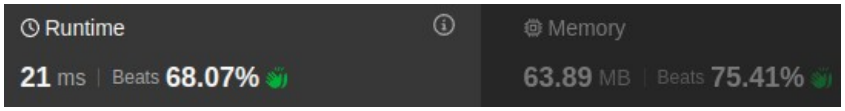
```
/*
   Hash map + Min heap (keep at most the top two largest integers)
   Time complexity: O(n(logm+log2+n)=O(nlogm+n)=O(nlogm)
   Space complexity: O(2logm)
   m: max number in the given array
*/
```




```
typedef std::vector<int> vi;
typedef std::pair<int,int> ii;
class Solution {
   public:
      // Function to compute the sum of digits of a positive intrger x
      // Time complexity: O(logx)
      int sum_digits(int x){
         int s=0;
         while(x!=0){
            s+=x%10;
            x/=10;
         }
         return s;
      }
```

```
int maximumSum(vi& nums){
        int n=nums.size();

        // Hash map to map the sum of digits with at most the top two largest numbers given that sum
        // Min heap to track the top two largest numbers
        std::unordered_map<int,std::priority_queue<int,vi,std::greater<int>>> sum_numbers;

        for(int i=0;i<n;++i){
           // Compute the sum of digits of nums[i]
           int s=sum_digits(nums[i]);

           // Map nums[i] with its sum of digit by pushing it into the min heap
           sum_numbers[s].push(nums[i]);

           // To keep only at most the two largest elements giving the sum of digits s
           while(sum_numbers[s].size()>2) sum_numbers[s].pop();
        }

        int ans=-1; // -1, if there are no two numbers that satisfy the conditions
        // Iterate over each sum of digits and get its min heap
        for(auto& [sum,min_heap]: sum_numbers){
           // If there is one element in the min heap, ignore that sum
           if(min_heap.size()<2) continue;

           // Otherwise, we are sue that there are two numbers that satisfy the conditions
           int s1=min_heap.top();
           min_heap.pop();
           int s2=min_heap.top();
           min_heap.pop();

           // Maximize the answer
           ans=std::max(ans,s1+s2);
        }

        return ans;
    }
};
```

2342. Max Sum of a Pair With Equal Sum of Digits




```
/*
   Hash map: Track the max number given a sum of digit
   Time complexity: O(nlogm)
   Space complexity: O(logm)
   m: max number in the given array
*/
typedef std::vector<int> vi;
typedef std::pair<int,int> ii;
class Solution {
   public:
      int sum_digits(int x){
         int s=0;
         while(x!=0){
            s+=x%10;
            x/=10;
         }
         return s;
      }

      int maximumSum(vi& nums){
         int n=nums.size();

         std::unordered_map<int,int> sum_numbers;

         int ans=-1;
         for(int i=0;i<n;++i){
            int s=sum_digits(nums[i]);
            // If the sum of digits is already computed
            // add the number store in the map to the current one and maximize the answer
            if(sum_numbers.find(s)!=sum_numbers.end()) ans=std::max(ans,sum_numbers[s]+nums[i]);

            // Store only the max number given that sum
            sum_numbers[s]=std::max(sum_numbers[s],nums[i]);
         }

         return ans;
      }
};
```

2342. Max Sum of a Pair With Equal Sum of Digits

```
/*
   Array: Track the max number given a sum of digit
   Time complexity: O(nlogm)=O(9n)=O(n)
   Space complexity: O(82)=O(1)
   m=999999999
*/
typedef std::vector<int> vi;
typedef std::pair<int,int> ii;
class Solution {
   public:
      int sum_digits(int x){
         int s=0;
         while(x!=0){
            s+=x%10;
            x/=10;
         }
         return s;
      }

      int maximumSum(vi& nums){
         int n=nums.size();

         vi sum_numbers(82,-1);

         int ans=-1;
         for(int i=0;i<n;++i){
            int s=sum_digits(nums[i]);
            // If the sum of digits is already computed
            // add the number store in the map to the current one and maximize the answer
            if(sum_numbers[s]!=-1) ans=std::max(ans,sum_numbers[s]+nums[i]);

            // Store only the max number given that sum
            sum_numbers[s]=std::max(sum_numbers[s],nums[i]);
         }

         return ans;
      }
};
```