1002 Find Common Characters


```go
func commonChars(words []string) []string {
  alphas := make([][]int, len(words))

  for i := 0; i < len(words); i++{
    word := words[i]

    alphas[i] = make([]int, 26)
    for j := 0; j < len(word); j++{
      alphas[i][word[j] - 'a']++
    }
  }

  res := make([]string, 0)
  for i := 0; i < 26; i++{
    counter := 0
    num := 1000
    for j := 0; j < len(alphas); j++{
      if alphas[j][i] == 0{
        break
      }else{
        counter++
        if num > alphas[j][i]{
          num = alphas[j][i]
        }
      }
    }

    if counter == len(words){
      for k := num; k > 0; k--{
        res = append(res, string(rune('a'+i)))
      }
    }
  }

  return res
}

```

1004 Max Consecutive Ones III

执行结果： 通过 显示详情 > 添加

执行用时： 40 ms , 在所有 C++ 提交中击败了 98.87% 的用户

内存消耗： 54.1 MB , 在所有 C++ 提交中击败了 22.83% 的用户

炫耀一下：

```cpp
class Solution {
public:
    int longestOnes(vector<int>& nums, int k) {
        int res = 0;
        int left = 0, right = 0;
        int countZero = 0;
        int size = nums.size();

        while(right < size) {
            while(right < size && countZero <= k){
                res = max(res, right - left);

                if(nums[right] == 0)
                    countZero++;
                right++;
            }

            if(right == size){
                if(countZero <= k)
                    res = max(res, right - left);
                break;
            }

            while(left < right && countZero > k){
                if(nums[left] == 0)
                    countZero--;
                left++;
            }
        }

        return res;
    }
```

```
};

```

```
//典型双指针
func longestOnes(nums []int, k int) int {
  left, right := 0, 0

  countZero := 0
  res := 0
  for ; right < len(nums); {
    for ; right < len(nums) && countZero <= k ; right++{
      if nums[right] == 0{
        countZero++
      }

      if countZero > k{
        right++
        break
      }

      res = max(res, right - left + 1)
    }

    if right == len(nums){
      break
    }

    for ;left < right && countZero > k; left++{
      if nums[left] == 0{
        countZero--
      }
    }
  }

  return res
}

func max(a int, b int) int{
  if a > b{
    return a
  }

  return b
}
```

1010 Pairs

```java
public int numPairsDivisibleBy60(int[] time) {
    Map<Integer, List<Integer>> map = new HashMap<>();

    for(int i = 0; i < time.length; i++){
        int remains = time[i] % 60;
        map.putIfAbsent(remains, new ArrayList<>());

        map.get(remains).add(i);
    }

    int counter = 0;
    for(int i = 0; i <= 30; i++){
        if(i == 0 || i == 30){
            if(!map.containsKey(i))
                continue;
            int size = map.get(i).size();
            counter += size * (size - 1) / 2;
        }else{
            if(!map.containsKey(i) || !map.containsKey(60 - i))
                continue;
            int size1 = map.get(i).size();
            int size2 = map.get(60 - i).size();

            counter += size1 * size2;
        }
    }

    return counter;
 }
```

1019 Next Greater Node in Linked List

执行用时： 88 ms ，在所有 C++ 提交中击败了 77.11% 的用户

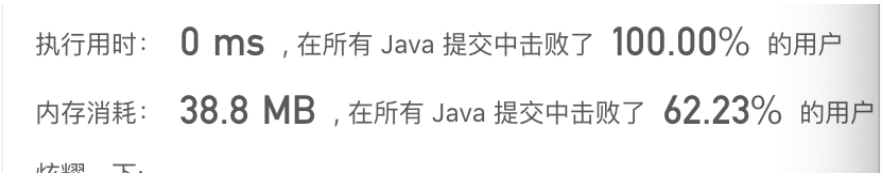
内存消耗： 44.4 MB ，在所有 C++ 提交中击败了 5.88% 的用户

炫耀一下:

```cpp
class Solution {
public:
    vector<int> res;
    stack<int> myStack;
    vector<int> nextLargerNodes(ListNode* head) {
        if(head == nullptr)
            return res;

        getRes(head);
        std::reverse(res.begin(), res.end());
        return res;
    }

    void getRes(ListNode* head){
        if(head == nullptr)
            return;

        getRes(head->next);

        while(!myStack.empty() && myStack.top() <= head->val){
            myStack.pop();
        }

        if(myStack.empty() || myStack.top() <= head->val)
            res.push_back(0);
        else
            res.push_back(myStack.top());



        myStack.push(head->val);
    }
};
```

1025 Divisor Game

1026Maximum Difference Between Node and Ancestor




```java
    int res = 0;
    public int maxAncestorDiff(TreeNode root) {
        helper(root, root.val, root.val);
        return res;
    }

    private void helper(TreeNode root, int max, int min){
        if(root == null)
            return;

        res = Math.max(res, Math.abs(root.val - max));
        res = Math.max(res, Math.abs(root.val - min));

        helper(root.left, Math.max(max, root.val), Math.min(min, root.val));
        helper(root.right, Math.max(max, root.val), Math.min(min, root.val));
    }
```

1031 Maximum Sum of Two Non Overlapping Subarrays

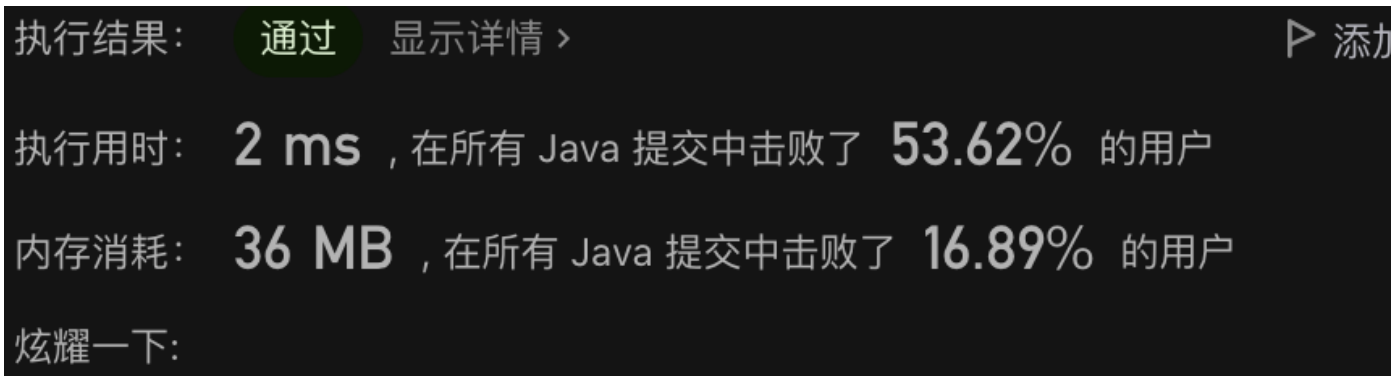
```java
/*
  思路来源 花花

  因为本身 问题规模 在 1000左右
  那么实际上可以用搜索来做

```

```
  先定位 firstLen,  再定位 secondLen
  采用前缀和 减少问题规模

  时间复杂度 ON2
*/
class Solution {
public:
    int maxSumTwoNoOverlap(vector<int>& nums, int firstLen, int secondLen) {
        int size = nums.size();
        vector<int> prefixSum(size + 1, 0);

        for(int i = 0; i < size; i++)
            prefixSum[i + 1] += prefixSum[i] + nums[i];

        int res = 0;
        for(int i = 0; i <= size - firstLen; i++){   //check the status
            int firstSum = prefixSum[i + firstLen] - prefixSum[i];
            int secondSum = std::max(maxSum(prefixSum, 0, i - 1, secondLen),
                                     maxSum(prefixSum, i + firstLen + 1, size,
secondLen));

            res = max(res, firstSum + secondSum);
        }

        return res;
    }

    int maxSum(const vector<int>& prefixSum, int left, int right, int secondLen){
        if(right - left + 1 < secondLen)
            return INT_MIN;

        int res = INT_MIN;
        for(int i = left; i + secondLen - 1 <= right && i + secondLen <
prefixSum.size(); i++) {
            res = std::max(res, prefixSum[i + secondLen] - prefixSum[i]);
        }

        return res;
    }
};
```

1041 Robot Bounded In Circle

```java
    public boolean isRobotBounded(String instructions) {
        int[][] dir = new int[][]{{-1, 0}, {0, 1}, {1, 0}, {0, -1}};
        int ori = 0;
        int x = 0, y = 0;

        for(char ch : instructions.toCharArray()){
            if(ch == 'L')
                ori = (ori - 1 + 4) % 4;
            else if(ch == 'R')
                ori = (ori + 1) % 4;
            else{
                x += dir[ori][0];
                y += dir[ori][1];
            }

        }

        return (x == 0 && y == 0) || ori != 0;
    }
```

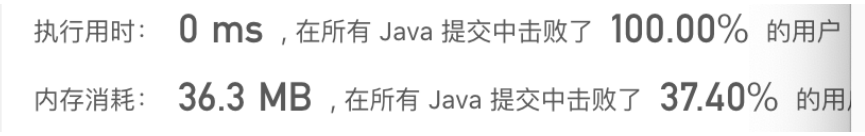
1046 Last Stone Weight




```java
class Solution {
    public int lastStoneWeight(int[] stones) {
        PriorityQueue<Integer> queue = new PriorityQueue<>((o1, o2) -> o2 - o1);
        for(int num : stones){
            queue.add(num);
        }

        while(queue.size() > 1){
```

```
            int left = queue.poll();
            int right = queue.poll();

            if(left != right){
                queue.add(Math.abs(left - right));
            }
        }

        return queue.size() == 0 ? 0 : queue.poll();
    }
}
```

1051 Height Checker




```
    public int heightChecker(int[] heights) {
        int[] arr = new int[101];
        for(int num : heights)
            arr[num]++;

        int counter = 0;
        int index = 0;
        for(int i = 1; i <= 100; i++){
            if(arr[i] == 0)
                continue;

            for(int j = 0; j < arr[i]; j++){
                if(heights[index] != i)
                    counter++;
                index++;
            }
        }

        return counter;
    }
```

执行结果： 通过 显示详情 >

执行用时： 1 ms ，在所有 Java 提交中击败了 67.22% 的用户

内存消耗： 36 MB ，在所有 Java 提交中击败了 84.68% 的用户

炫耀一下:

```java
    public int heightChecker(int[] heights) {
        int[] newHeights = Arrays.copyOf(heights, heights.length);
        Arrays.sort(newHeights);
        int counter = 0;
        for(int i = 0; i < heights.length; i++){
            if(newHeights[i] != heights[i])
                counter++;
        }

        return counter;
    }
```

1054 Distant Barcodes

执行用时： 120 ms ，在所有 C++ 提交中击败了 42.93% 的用户

内存消耗： 44.1 MB ，在所有 C++ 提交中击败了 16.75% 的用户

炫耀一下:

```cpp
class Solution {
public:
    vector<int> rearrangeBarcodes(vector<int>& barcodes) {
        vector<int> res;

        unordered_map<int, int> map;
        for(int barcode : barcodes)
            map[barcode]++;

        auto cmp = [&](pair<int ,int>& p1, pair<int, int>& p2){return p1.second == p2.second ? p1.first > p2.first : p1.second < p2.second;};
        priority_queue<pair<int, int>, vector<pair<int, int>>, decltype(cmp)> pq(cmp);

        for(auto it = map.begin(); it != map.end(); it++)
            pq.push({it->first, it->second});

```

```
        deque<int> myQueue;
        while(!pq.empty()){
            pair<int, int> curPair = pq.top(); pq.pop();
            map[curPair.first]--;
            myQueue.push_back(curPair.first);
            res.push_back(curPair.first);

            if(myQueue.size() == 2){
                int curNum = myQueue.front(); myQueue.pop_front();
                if(map[curNum] != 0){
                    pq.push({curNum, map[curNum]});
                }
            }
        }

        return res;
    }
};
```

1099 Two Sum Less Than K

执行结果： 通过 显示详情 › 添加

执行用时： 1 ms , 在所有 Java 提交中击败了 100.00% 的用户

内存消耗： 37.6 MB , 在所有 Java 提交中击败了 80.20% 的用户

```
class Solution {
    public int twoSumLessThanK(int[] nums, int k) {
        // O(nlgn)

       //1 2 3 4 5 6  7 7 8 8 8 8
        //   |                   |
       //    15

       Arrays.sort(nums);
       int left = 0, right = nums.length - 1;

        int sum = -1;
       while(left < right){
```

```
            int temp = nums[left] + nums[right];
            if(temp < k){
                sum = Math.max(sum, temp);
                left++;
            }else{
                right--;
            }

        }

        return sum;
    }
}
```