1313 Decompress Run-Length Encoded List

```go
func decompressRLElist(nums []int) []int {
  res := make([]int, 0)
  for i := 0; i < len(nums); i = i + 2{
    freq := nums[i]
    val  := nums[i + 1]

    for j := 0; j < freq; j++{
      res = append(res, val)
    }
  }

  return res
}
```

1329 Sort the Matrix Diagonally

```go
/*
  不错的思路，主要利用了 i - j 对于左对角线相同的原理
*/
func diagonalSort(mat [][]int) [][]int {
  row := len(mat)
  col := len(mat[0])

  lookup := make(map[int][]int)

  for i := 0; i < row; i++{
    for j := 0; j < col; j++{
      lookup[i - j] = append(lookup[i-j], mat[i][j])
    }
  }

```

```
  for key, _ := range lookup {
    sort.Ints(lookup[key])
  }

  for i := 0; i < row; i++{
    for j := 0; j < col; j++{
      temp := lookup[i - j][0]
      lookup[i - j] = lookup[i - j][1:]

      mat[i][j] = temp
    }
  }

  return mat
}

```

1379 Find a Corresponding of a Binary Tree in a

```
方法二
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode(int x) : val(x), left(NULL), right(NULL) {}
 * };
 */

class Solution {
public:
    TreeNode* res = nullptr;
    TreeNode* getTargetCopy(TreeNode* original, TreeNode* cloned, TreeNode* target)
{
        inorder(original, cloned, target);
        return res;
    }
```

```
19
20      void inorder(TreeNode* original, TreeNode* cloned, TreeNode* target){
21          if(original == nullptr || cloned == nullptr)
22              return;
23          if(res != nullptr)
24              return;
25
26          if(original == target){
27              res = cloned;
28              return;
29          }
30
31          inorder(original->left, cloned->left, target);
32          inorder(original->right, cloned->right, target);
33
34      }
35  };
```

执行用时： 440 ms ，在所有 C++ 提交中击败了 99.82% 的用户

内存消耗： 160 MB ，在所有 C++ 提交中击败了 88.57% 的用户

炫耀一下:

```
16
17
18
19
20
```

```
1   /**
2    * Definition for a binary tree node.
3    * struct TreeNode {
4    *     int val;
5    *     TreeNode *left;
6    *     TreeNode *right;
7    *     TreeNode(int x) : val(x), left(NULL), right(NULL) {}
8    * };
9    */
10
11  class Solution {
12  public:
13      TreeNode* res = nullptr;
14      TreeNode* getTargetCopy(TreeNode* original, TreeNode* cloned, TreeNode* target)
    {
15          inorder(original, cloned, target);
16          return res;
17      }
```

```
18
19      void inorder(TreeNode* original, TreeNode* cloned, TreeNode* target){
20          if(res != nullptr)
21              return;
22
23          if(original == nullptr || cloned == nullptr)
24              return;
25
26          if(target->val == cloned->val){
27              res = cloned;
28              return;
29          }
30
31          inorder(original->left, cloned->left, target);
32          inorder(original->right, cloned->right, target);
33
34      }
35  };
```

1342 Number of Steps to Reduce a Number to Zero

```
1   func numberOfSteps(num int) int {
2       res := 0
3       for ; num != 0; {
4           if num % 2 == 0{
5               num /= 2
6           }else{
7               num -= 1
8           }
9
10          res = res + 1
11      }
12
13      return res
14  }
```

1383 Maximum Performance of a Tream

1383. Maximum Performance of a Team

难度 困难 91

You are given two integers `n` and `k` and two integer arrays `speed` and `efficiency` both of length `n`. There are `n` engineers numbered from `1` to `n`. `speed[i]` and `efficiency[i]` represe the speed and efficiency of the i^{th} engineer respectively.

Choose **at most** `k` different engineers out of the `n` engineers to fo a team with the maximum **performance**.

The performance of a team is the sum of their engineers' speeds multiplied by the minimum efficiency among their engineers.

Return *the maximum performance of this team*. Since the answer ca be a huge number, return it **modulo** $10^9 + 7$.

Example 1:

```
Input: n = 6, speed = [2,10,3,1,5,8], efficiency =
[5,4,3,9,7,2], k = 2
Output: 60
Explanation:
We have the maximum performance of the team by
selecting engineer 2 (with speed=10 and
efficiency=4) and engineer 5 (with speed=5 and
efficiency=7). That is, performance = (10 + 5) *
min(4, 7) = 60.
```

Example 2:

执行用时： 80 ms ，在所有 C++ 提交中击败了 81.03% 的用户

内存消耗： 38.3 MB ，在所有 C++ 提交中击败了 21.55% 的用户

```
/*
  ref:https://www.youtube.com/results?search_query=leetcode+1383
  看到有两个评价指标纬度的题目， 一般的想法都是固定一个纬度， 然后求解另一个维度

  我们通过将效率从大到小进行排序，
  这样可以保证，每次拿到的效率，都是 minEfficiency

  然后对 Sum 进行不断求和， 有些类似前缀和的概念
    数据结构采用优先队列， 这样可以保证效率低的会在堆顶，然后及时 pop
```

```cpp
*/
class Solution {
public:
    int maxPerformance(int n, vector<int>& speed, vector<int>& efficiency, int k) {
        vector<pair<long, long>> persons;
        for(int i = 0; i < n; i++)
            persons.push_back({efficiency[i], speed[i]});

        //
        sort(persons.rbegin(), persons.rend());

        //to store max speed
        long long speedSum = 0;
        long long res = 0;
        priority_queue<int, vector<int>, greater<>> pq;
        for(int i = 0; i < n; i++){
            if(pq.size() > k - 1){
                speedSum -= pq.top();
                pq.pop();
            }

            speedSum += persons[i].second;
            res = std::max(res, speedSum * persons[i].first);

            pq.push(persons[i].second);
        }

        int M = 1e9 + 7;
        return res % M;
    }
};
```

1395 Count Number of Teams

1395. Count Number of Teams

难度 中等 63

There are `n` soldiers standing in a line. Each soldier is assigned a **unique** `rating` value.

You have to form a team of 3 soldiers amongst them under the following rules:

- Choose 3 soldiers with index (`i` , `j` , `k`) with rating (`rating[i]` , `rating[j]` , `rating[k]`).
- A team is valid if: (`rating[i] < rating[j] < rating[k]`) or (`rating[i] > rating[j] > rating[k]`) where (`0 <= i < j < k < n`).

Return the number of teams you can form given the conditions. (soldiers can be part of multiple teams).

Example 1:

```
Input: rating = [2,5,3,4,1]
Output: 3
Explanation: We can form three teams given the conditions. (2,3,4), (5,4,1), (5,3,1).
```

Example 2:

```
Input: rating = [2,1,3]
Output: 0
Explanation: We can't form any team given the conditions.
```

Example 3:

```
Input: rating = [1,2,3,4]
Output: 4
```

Constraints:

- `n == rating.length`
- `3 <= n <= 1000`

执行用时： 24 ms ，在所有 Go 提交中击败了 61.54% 的用户

内存消耗： 2.8 MB ，在所有 Go 提交中击败了 61.54% 的用户

```go
/*
  不错的时间复杂度优化
*/
func numTeams(rating []int) int {

  res := 0
  for j := 1; j < len(rating) - 1; j++{
    iLess, iMore := 0, 0
    kLess, kMore := 0, 0
    for i := 0; i < j; i++{
      if rating[i] < rating[j]{
        iLess++
      }else{
        iMore++
      }
    }

    for k := j + 1; k < len(rating); k++{
      if rating[k] < rating[j]{
        kLess++
      }else{
        kMore++
      }
    }

    res += iLess * kMore + iMore * kLess
  }

  return res
}
```