

## 901 Online Stock Span

执行用时: **37 ms** , 在所有 Java 提交中击败了 **76.19%** 的用户

内存消耗: **47.9 MB** , 在所有 Java 提交中击败了 **5.03%** 的用户

```
1 public int[] dp;
2 public int[] nums;
3 public int index;
4 public StockSpanner() {
5     dp = new int[110000];
6     Arrays.fill(dp, 1);
7     nums = new int[110000];
8     index = 0;
9 }
10
11 public int next(int price) {
12     nums[index] = price;
13
14     //get res from dp
15     int i = index - 1;
16     while(i >= 0){
17         if(nums[i] <= nums[index]){
18             dp[index] += dp[i];
19             i = i - dp[i];
20         }else{
21             break;
22         }
23     }
24
25     index++;
26     return dp[index - 1];
27 }
28
```

## 905 Sort Array By Parity

```

1 func sortByParity(A []int) []int {
2     left := 0
3     right := len(A) - 1
4
5     for ; left < right; {
6         for ; left < right && A[left] % 2 == 0 ; left++{}
7
8         if left >= right{
9             break
10        }
11
12        for ; left < right && A[right] % 2 != 0 ; right--{}
13
14        if left >= right{
15            break
16        }
17
18        A[left], A[right] = A[right], A[left]
19        left++
20        right--
21    }
22
23    return A
24 }

```

```

1 public int[] sortByParity(int[] A) {
2     int len = A.length;
3     int left = 0, right = len - 1;
4     while(left < right){
5         while(left < len && A[left] % 2 == 0)
6             left++;
7
8         if(left >= right)
9             break;
10
11        while(right >= 0 && A[right] % 2 != 0)
12            right--;
13
14        if(left >= right)
15            break;
16
17        int temp = A[left];
18        A[left] = A[right];
19        A[right] = temp;
20
21        left++;

```

```

22         right--;
23     }
24
25     return A;
26 }

```

## 907 Sum of Subarray Minimum

执行结果： 通过 [显示详情 >](#)



执行用时： **104 ms** ，在所有 C++ 提交中击败了 **30.33%** 的用户

内存消耗： **41.9 MB** ，在所有 C++ 提交中击败了 **16.60%** 的用户

1.6K / 1.6K

```

1  /*
2      author: guoguo
3
4      本质上这道题目， 就是采用单调栈， 拿到 当前位置 i， 往左能扩展到哪里， 往右能扩展到哪里
5      如果采用 on2 遍历 比较费时间
6      因此 这里采用 单调栈， 找到左边最近的 不小于自己的
7                          找到右边最近的， 不小于自己的
8  */
9  class Solution {
10 public:
11     int sumSubarrayMins(vector<int>& arr) {
12         int size = arr.size();
13         long long res = 0;
14         int mod = 1e9 + 7;
15
16         vector<int> leftMin(size, -1);
17         vector<int> rightMin(size, -1);
18         stack<int> monoStack;
19
20         for(int i = 0; i < size; i++){
21             if(i == 0)
22                 monoStack.push(i);
23             else{
24                 if(monoStack.empty() || arr[monoStack.top()] <= arr[i])
25                     monoStack.push(i);
26                 else{
27                     while(!monoStack.empty() && arr[monoStack.top()] > arr[i]){
28                         leftMin[i] = monoStack.top();
29                         if(leftMin[monoStack.top()] != -1)
30                             leftMin[i] = leftMin[monoStack.top()];

```

```

31         monoStack.pop();
32     }
33
34     monoStack.push(i);
35 }
36 }
37 }
38
39 monoStack = stack<int>();
40
41 for(int i = size - 1; i >= 0; i--){
42     if(i == size - 1)
43         monoStack.push(i);
44     else{
45         if(monoStack.empty() || arr[monoStack.top()] < arr[i])
46             monoStack.push(i);
47         else{
48             while(!monoStack.empty() && arr[monoStack.top()] >= arr[i]){
49                 rightMin[i] = monoStack.top();
50                 if(rightMin[monoStack.top()] != -1)
51                     rightMin[i] = rightMin[monoStack.top()];
52                 monoStack.pop();
53             }
54
55             monoStack.push(i);
56         }
57     }
58 }
59
60 for(int i = 0; i < size; i++){
61     int left;
62     if(leftMin[i] == -1)
63         left = 1;
64     else
65         left = i - leftMin[i] + 1;
66     int right;
67
68     if(rightMin[i] == -1)
69         right = 1;
70     else
71         right = rightMin[i] - i + 1;
72
73     res += ((long)left * (long)right) * (long)arr[i];
74 }
75
76 return res % mod;
77 }
78 };

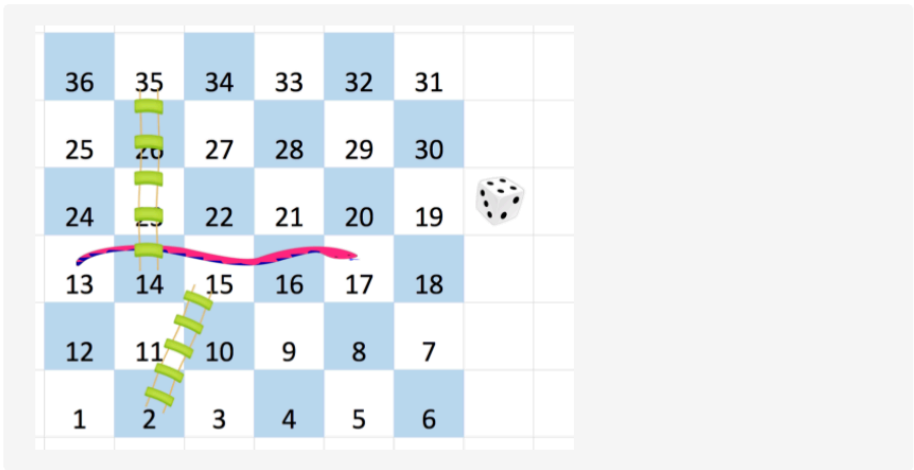
```

# 909 Snakes and Ladders

## 909. Snakes and Ladders

难度 中等 39 收藏 评论 通知 帮助

On an  $N \times N$  board, the numbers from 1 to  $N*N$  are written *boustrophedonically* starting from the bottom left of the board, and alternating direction each row. For example, for a 6 x 6 board, the numbers are written as follows:



You start on square 1 of the board (which is always in the last row and first column). Each move, starting from square  $x$ , consists of the following:

- You choose a destination square  $s$  with number  $x+1$ ,  $x+2$ ,  $x+3$ ,  $x+4$ ,  $x+5$ , or  $x+6$ , provided this number is  $\leq N*N$ .
  - (This choice simulates the result of a standard 6-sided die roll: ie., there are always **at most 6 destinations, regardless of the size of the board.**)
- If  $s$  has a snake or ladder, you move to the destination of that snake or ladder. Otherwise, you move to  $s$ .

A board square on row  $r$  and column  $c$  has a "snake or ladder" if  $board[r][c] \neq 1$ . The destination of that snake or ladder is

# 918 Maximum Sum Circular Subarray

执行用时:

92 ms

在所有 C++ 提交中击败了 16.70% 的用户

内存消耗:

51.3 MB

在所有 C++ 提交中击败了 5.05% 的用户

```
1
2 class Solution {
3 public:
4     int maxSubarraySumCircular(vector<int>& nums) {
5         int size= nums.size();
6         int doubleSize = 2 * size;
7         int temp = nums[0];
8
9         vector<int> prefixSum(doubleSize, 0);
10        for(int i = 0; i < doubleSize; i++){
11            int pos = i % size;
12            if(i == 0)
13                prefixSum[i] = nums[i];
14            else
15                prefixSum[i] += prefixSum[i - 1] + nums[pos];
16
17            temp = nums[pos] > temp ? nums[pos] : temp;
18        }
19
20        int index = doubleSize - 1;
21        deque<pair<int, int>> maxStack;
22        vector<int> lenMax(size + 1, 0);
23        while(index >= 0){
24            if(index >= size){
25                while(!maxStack.empty() && maxStack.back().first <
prefixSum[index])
26                    maxStack.pop_back();
27
28                if(maxStack.empty() || maxStack.back().first >= prefixSum[index])
29                    maxStack.emplace_back(prefixSum[index], index);
30
31                index--;
32            }else{
33                if(!maxStack.empty() && maxStack.front().second == index + size)
34                    maxStack.pop_front();
35
36                int curMax = maxStack.front().first;
37                lenMax[index] = curMax;
38
39                while(!maxStack.empty() && maxStack.back().first <
prefixSum[index])
```

```

40         maxStack.pop_back();
41
42         maxStack.emplace_back(prefixSum[index], index);
43
44         index--;
45     }
46 }
47
48 int res = prefixSum[size - 1];
49 for(int i = 0; i < size; i++){
50     if(res < lenMax[i] - prefixSum[i])
51         res = lenMax[i] - prefixSum[i];
52 }
53
54
55
56 return res > temp ? res : temp;
57 }
58 };

```

## 922 Sort Array By Parity II

执行用时： **28 ms** ，在所有 Go 提交中击败了 **52.94%** 的用户

内存消耗： **6.2 MB** ，在所有 Go 提交中击败了 **80.88%** 的用户

```

1 func sortArrayByParityII(nums []int) []int {
2     even, odd := 0, 1
3
4     for ;odd < len(nums) && even < len(nums);{
5         for;odd < len(nums) && nums[odd] % 2 != 0;{
6             odd += 2
7         }
8
9         for ;even < len(nums) && nums[even] % 2 == 0;{
10             even += 2
11         }
12
13         if odd >= len(nums) || even >= len(nums){
14             break

```

```

15     }
16
17     nums[odd],nums[even] = nums[even], nums[odd]
18     odd += 2
19     even += 2
20 }
21
22 return nums
23 }

```

## 937 Reorder Data in Log Files

```

1 public String[] reorderLogFiles(String[] logs) {
2     List<String> nums = new ArrayList<>();
3     List<String> alpha = new ArrayList<>();
4
5     for(int i = 0 ; i < logs.length; i++){
6         String log = logs[i];
7
8         int pos = log.indexOf(" ");
9         char ch = log.charAt(pos + 1);
10        if('0' <= ch && ch <= '9')
11            nums.add(log);
12        else
13            alpha.add(log);
14    }
15
16    alpha.sort(new Comparator<String>() {
17        @Override
18        public int compare(String o1, String o2) {
19            int pos1 = o1.indexOf(" ");
20            int pos2 = o2.indexOf(" ");
21
22            String sub1 = o1.substring(pos1 + 1);
23            String sub2 = o2.substring(pos2 + 1);
24
25            if (sub1.compareTo(sub2) == 0) {
26                return o1.substring(0, pos1).compareTo(o2.substring(0, pos2));
27            } else {
28                return sub1.compareTo(sub2);

```



```

29         }
30     }
31 });
32
33 String[] res = new String[logs.length];
34 for(int i = 0; i < logs.length; i++){
35     if(i < alpha.size())
36         res[i] = alpha.get(i);
37     else
38         res[i] = nums.get(i - alpha.size());
39 }
40
41 return res;
42 }

```

## 934 Shortest Bridge

执行用时: **236 ms** , 在所有 C++ 提交中击败了 **5.32%** 的用户

内存消耗: **29.6 MB** , 在所有 C++ 提交中击败了 **8.56%** 的用户

```

1  class Solution {
2  public:
3      vector<vector<int>>> dir{{1, 0}, {0, 1}, {-1, 0}, {0, -1}};
4      int row, col;
5      int shortestBridge(vector<vector<int>>& grid) {
6          row = grid.size();
7          col = grid[0].size();
8
9          queue<pair<int, int>> myQueue1;
10         queue<pair<int, int>> myQueue2;
11
12         unordered_set<string> set1;
13         unordered_set<string> set2;
14         bool jump = false;
15
16         for(int i = 0; i < row && !jump; i++){
17             for(int j = 0; j < col && !jump; j++){
18                 if(grid[i][j] == 1){
19                     dfs(grid, i, j);
20                     jump = !jump;
21                 }
22             }
23         }

```

```

24
25 // island 1 and island 2
26 for(int i = 0; i < row; i++){
27     for(int j = 0; j < col; j++){
28         if(grid[i][j] == 1){
29             myQueue1.push({i, j});
30             set1.insert(to_string(i) + "@" + to_string(j));
31         }else if(grid[i][j] == 2){
32             myQueue2.push({i, j});
33             set2.insert(to_string(i) + "@" + to_string(j));
34         }
35     }
36 }
37
38 if(myQueue2.size() == 0 || myQueue1.size() == 0)
39     return 0;
40
41 int count = 0;
42 while(true){
43     int size1 = myQueue1.size();
44     for(int i = 0; i < size1; i++){
45         pair<int, int> curPos1 = myQueue1.front(); myQueue1.pop();
46
47         for(int k = 0; k < 4; k++){
48             int newX = curPos1.first + dir[k][0];
49             int newY = curPos1.second + dir[k][1];
50             string symbol = to_string(newX) + "@" + to_string(newY);
51
52             if(set2.count(symbol) != 0)
53                 return 2 * count;
54
55             if(isInRange(newX, newY) && set1.count(symbol) == 0){
56                 myQueue1.push({newX, newY});
57                 set1.insert(symbol);
58             }
59         }
60     }
61
62     int size2 = myQueue2.size();
63     for(int i = 0; i < size2; i++){
64         pair<int, int> curPos2 = myQueue2.front(); myQueue2.pop();
65
66         for(int k = 0; k < 4; k++){
67             int newX = curPos2.first + dir[k][0];
68             int newY = curPos2.second + dir[k][1];
69             string symbol = to_string(newX) + "@" + to_string(newY);
70
71             if(set1.count(symbol) != 0)
72                 return 2 * count + 1;

```

```

73
74         if(isInRange(newX, newY) && set2.count(symbol) == 0){
75             myQueue2.push({newX, newY});
76             set2.insert(symbol);
77         }
78     }
79 }
80
81     count++;
82 }
83
84     return -1;
85 }
86
87 inline bool isInRange(int i ,int j){
88     return i >= 0 && j >= 0 && i < row && j < col;
89 }
90
91 void dfs(vector<vector<int>>& grid, int i, int j){
92     grid[i][j] = 2;
93
94     for(int k = 0; k < 4; k++){
95         int newX = i + dir[k][0];
96         int newY = j + dir[k][1];
97
98         if(isInRange(newX, newY) && grid[newX][newY] == 1){
99             dfs(grid, newX, newY);
100         }
101     }
102 }
103 };
104

```

## 935 Knight Dialer

```

1  ref : https://leetcode-cn.com/problems/knight-dialer/solution/4zhuang-tai-dong-tai-gui-hua-pythonjie-kong-jian-f/
2  class Solution {
3  public:
4      int mod = 1e9 + 7;
5      int knightDialer(int n) {
6          if(n == 1)
7              return 10;

```

```

8      vector<long> dp(4, 1);
9      for(int i = 2; i <= n; i++){
10         vector<long> newDP(4, 0);
11         newDP[0] = (dp[1] % mod + dp[2] % mod) % mod;
12         newDP[1] = (2 * dp[0]) % mod;
13         newDP[2] = ((2 * dp[0]) % mod + dp[3] % mod) % mod;
14         newDP[3] = (2 * dp[2]) % mod;
15
16         swap(dp, newDP);
17     }
18
19     return (4 * dp[0] + 2 * dp[1] + 2 * dp[2] + dp[3]) % mod;
20 }
21 };

```

执行用时: 1144 ms

, 在所有 C++ 提交中击败了 5.03% 的用户

内存消耗: 217.6 MB , 在所有 C++ 提交中击败了 5.03% 的用户

```

1  class Solution {
2  public:
3      int mod = 1000000007;
4      int knightDialer(int n) {
5          unordered_map<int, vector<int>> lookup;
6          lookup[1] = {6, 8};
7          lookup[2] = {7, 9};
8          lookup[3] = {4, 8};
9          lookup[4] = {3, 9, 0};
10         lookup[5] = {};
11         lookup[6] = {0, 1, 7};
12         lookup[7] = {2, 6};
13         lookup[8] = {1, 3};
14         lookup[9] = {2, 4};
15         lookup[0] = {4, 6};
16
17         unordered_map<int, long> map;
18         for(int i = 0; i < 10; i++)
19             map[i] = 1;
20
21         for(int i = 2; i <= n; i++){
22             unordered_map<int, long> temp;
23

```

```

24         for(int j = 0; j < 10; j++){
25             for(int nextDest : lookup[j]){
26 //                 cout << "in j " << j << " next dest" << nextDest << "map ->
" << map[j] << endl;
27                 temp[nextDest] += map[j];
28                 temp[nextDest] %= mod;
29             }
30         }
31
32         map = temp;
33     }
34
35     long res = 0;
36     for(int i = 0; i < 10; i++)
37         res += map[i];
38
39     return (int)(res % mod);
40 }
41 };

```

## 938 Range Sum of BST

```

1     int sum = 0;
2     public int rangeSumBST(TreeNode root, int low, int high) {
3         preorder(root, low, high);
4         return sum;
5     }
6
7     private void preorder(TreeNode root, int low, int high){
8         if(root == null)
9             return;
10
11         if(root.val >= low && root.val <= high)
12             sum += root.val;
13
14         preorder(root.left, low, high);
15         preorder(root.right, low, high);
16     }

```

## 941Valid Mountain Array

执行结果： **通过** [显示详情 >](#)



执行用时： **28 ms** ，在所有 C++ 提交中击败了 **79.66%** 的用户

内存消耗： **21.9 MB** ，在所有 C++ 提交中击败了 **40.52%** 的用户

```
1  class Solution {
2  public:
3      bool validMountainArray(vector<int>& arr) {
4          int len = arr.size();
5          if(len < 3)
6              return false;
7
8          int index = 1;
9          while(index < len - 1 && arr[index] > arr[index - 1])
10             index++;
11
12         if(index == 1){
13             if(arr[index] <= arr[index - 1])
14                 return false;
15         }else if(index == len - 1){
16             return arr[index - 1] > arr[index] ;
17         }
18
19         while(index < len - 1 && arr[index] > arr[index + 1])
20             index++;
21
22         return index == len - 1;
23     }
24 };
```

## 945 Minimum Increment to Make Array Unique

```

1  class Solution {
2  public:
3      int minIncrementForUnique(vector<int>& nums) {
4          int len = nums.size();
5
6          sort(nums.begin(), nums.end());
7          int count = 0;
8          int curNum = -1;
9          for(int i = 0; i < len; i++){
10             if(curNum >= nums[i]){
11                 count += curNum + 1 - nums[i];
12                 curNum += 1;
13             }else{
14                 curNum = nums[i];
15             }
16         }
17
18         return count;
19     }
20 };

```

## 977 Squares of a Sorted Array

执行用时: **36 ms** , 在所有 Go 提交中击败了 **72.20%** 的用户

内存消耗: **6.8 MB** , 在所有 Go 提交中击败了 **17.69%** 的用户

炫耀一下:

```

1  func sortedSquares(nums []int) []int {
2      index := 0
3      for ; index < len(nums); index++){
4          if index > 0 && nums[index] * nums[index - 1] <= 0{
5              break
6          }
7      }
8
9      res := make([]int, 0)
10
11     if index == len(nums){
12         if nums[index - 1] > 0{

```

```

13         for index = 0; index < len(nums); index++){
14             res = append(res, nums[index] * nums[index])
15         }
16     }else{
17         for index = len(nums) - 1; index >= 0; index--{
18             res = append(res, nums[index] * nums[index])
19         }
20     }
21
22 }else{
23     left := index - 1
24     right := index
25
26     for ;left >= 0 || right < len(nums);{
27         if left < 0{
28             res = append(res, nums[right] * nums[right])
29             right++
30         }else if right >= len(nums){
31             res = append(res, nums[left] * nums[left])
32             left--
33         }else if nums[left] * nums[left] < nums[right] * nums[right]{
34             res = append(res, nums[left] * nums[left])
35             left--
36         }else{
37             res = append(res, nums[right] * nums[right])
38             right++
39         }
40     }
41
42 }
43
44 return res
45 }
46

```

## 979 Distribute Coins in Binary Tree

```

1  class Solution {
2  public:
3      int count = 0;
4      int distributeCoins(TreeNode* root) {
5          postorder(root);
6          return count;

```



```

7     }
8
9     int postorder(TreeNode* root){
10         if(root == nullptr)
11             return 0;
12
13         int left = postorder(root->left);
14         int right = postorder(root->right);
15
16         int cur = root->val - 1 + left + right;
17
18         count += abs(cur);
19         return cur;
20
21     }
22 };

```

## 984 String Without AAA

```

1  /*
2      简单的思路，就是当 a和b个数相等的时候，最舒服
3      直接 ababababa
4
5      剩下的就是往里面插a & b
6      当然a > b 和 b > a 是不同的情况，需要多加注意
7  */
8
9  class Solution {
10 public:
11     string strWithout3a3b(int a, int b) {
12         int minFreq = std::min(a, b);
13         string res = "";
14         for(int i = 0; i < minFreq; i++) {
15             if(a > b)
16                 res += "ab";
17             else
18                 res += "ba";
19         }
20
21         int count = 0;
22         if(a > b){
23             for(int i = 0; count != a - b && i < res.size();i++){
24                 if(res[i] == 'b') {

```

```

25         res.insert(res.begin() + i, 'a');
26         count++;
27         i += 2;
28     }
29 }
30 }else if(a < b){
31     for(int i = 0; count != b - a && i < res.size(); i++){
32         if(res[i] == 'a') {
33             res.insert(res.begin() + i, 'b');
34             count++;
35             i += 2;
36         }
37     }
38 }
39
40 if(count != abs(a - b)){
41     for(int i = 0; i < abs(a - b) - count; i++){
42         if(a > b)
43             res += "a";
44         else
45             res += "b";
46     }
47 }
48 return res;
49 }
50 };
51

```

```

1  /*
2     更好的 PQ 解法
3     ref https://github.com/wisdompeak/LeetCode
4
5     整体思路是 每次从堆中拿两个元素,
6     比如 a > b
7     那么我就尽量多拿a, 拿2个a 一个b
8     res += aab
9  */
10 class Solution {
11 public:
12     string strWithout3a3b(int a, int b) {
13         priority_queue<pair<int, int>> pq;
14         pq.push({a, 'a'});
15         pq.push({b, 'b'});
16
17         string res = "";

```

```

18         while(!pq.empty()){
19             if(pq.size() == 1){
20                 int freq = pq.top().first;
21                 if(freq > 2)
22                     return "";
23                 else{
24                     for(int i = 0; i < freq; i++)
25                         res.push_back(pq.top().second);
26                     return res;
27                 }
28             }
29
30             auto x = pq.top(); pq.pop();
31             auto y = pq.top(); pq.pop();
32
33             int k = std::min(1 + x.first - y.first, 2);
34             for(int i = 0; i < k; i++)
35                 res.push_back(x.second);
36             res.push_back(y.second);
37
38             x.first -= k;
39             y.first -= 1;
40             if(x.first > 0)    pq.push({x.first, x.second});
41             if(y.first > 0)    pq.push({y.first, y.second});
42
43         }
44
45         return res;
46     }
47 };

```

## 994 Rotting-Orange

执行结果： 通过 [显示详情 >](#)



执行用时： **12 ms** ，在所有 C++ 提交中击败了 **31.73%** 的用户

内存消耗： **12.8 MB** ，在所有 C++ 提交中击败了 **45.55%** 的用户

炫耀一下：

```

1 //进行优化
2 class Solution {
3 public:
4     vector<vector<int>>> dir{{1, 0}, {0, 1}, {-1, 0}, {0, -1}};
5     int row;
6     int col;
7     int orangesRotting(vector<vector<int>>& grid) {
8         row = grid.size();
9         col = grid[0].size();
10
11         queue<pair<int, int>> myQueue;
12         int fresh = 0;
13         for(int i = 0; i < row; i++){
14             for(int j = 0; j < col; j++){
15                 if(grid[i][j] == 2){
16                     myQueue.push({i, j});
17                     grid[i][j] = -1;
18                 }else if(grid[i][j] == 1)
19                     fresh++;
20             }
21         }
22
23         if(fresh == 0)
24             return 0;
25         auto isInRange = [&](int i, int j){return i >= 0 && j >= 0 && i < row && j
< col;};
26
27         int res = 0;
28         int round = 0;
29         while(!myQueue.empty()){
30             int size = myQueue.size();
31             for(int i = 0; i < size; i++){
32                 pair<int, int> curPos = myQueue.front();
33                 myQueue.pop();
34
35                 for(int k = 0; k < dir.size(); k++){
36                     int newX = curPos.first + dir[k][0];
37                     int newY = curPos.second + dir[k][1];
38                     string symbol = to_string(newX) + "@" + to_string(newY);
39
40                     if(isInRange(newX, newY) && grid[newX][newY] == 1){
41                         fresh--;
42
43                         if(fresh == 0)
44                             return round + 1;
45                         grid[newX][newY] = -1;
46                         myQueue.push({newX, newY});
47                     }
48                 }

```

```

49         }
50
51         round++;
52     }
53
54     return fresh == 0 ? round : -1;
55 }
56
57
58 };

```

执行用时: 24 ms 超过 99.99% 的用户

执行用时: **24 ms** , 在所有 C++ 提交中击败了 **9.97%** 的用户

内存消耗: **13.8 MB** , 在所有 C++ 提交中击败了 **5.99%** 的用户

```

1
2 class Solution {
3 public:
4     vector<vector<int>>> dir{{1, 0}, {0, 1}, {-1, 0}, {0, -1}};
5     int row;
6     int col;
7     int orangesRotting(vector<vector<int>>& grid) {
8         row = grid.size();
9         col = grid[0].size();
10
11         queue<pair<int, int>> myQueue;
12         unordered_set<string> set;
13         int fresh = 0;
14         for(int i = 0; i < row; i++){
15             for(int j = 0; j < col; j++){
16                 if(grid[i][j] == 2){
17                     myQueue.push({i, j});
18                     set.emplace(to_string(i) + "@" + to_string(j));
19                 }else if(grid[i][j] == 1)
20                     fresh++;
21             }
22         }
23
24         if(fresh == 0)
25             return 0;
26
27         int res = 0;
28         int round = 0;
29         while(!myQueue.empty()){
30             int size = myQueue.size();

```

```

31         for(int i = 0; i < size; i++){
32             pair<int, int> curPos = myQueue.front();
33             myQueue.pop();
34
35             for(int k = 0; k < dir.size(); k++){
36                 int newX = curPos.first + dir[k][0];
37                 int newY = curPos.second + dir[k][1];
38                 string symbol = to_string(newX) + "@" + to_string(newY);
39
40                 if(isInRange(newX, newY) && set.count(symbol) == 0 &&
grid[newX][newY] != 0){
41                     set.emplace(symbol);
42
43                     if(grid[newX][newY] == 1)
44                         fresh--;
45
46                     if(fresh == 0)
47                         return round + 1;
48                     grid[newX][newY] = 2;
49                     myQueue.push({newX, newY});
50                     set.emplace(symbol);
51                 }
52             }
53         }
54
55         round++;
56     }
57
58     return fresh == 0 ? round : -1;
59 }
60
61 bool isInRange(int i, int j){
62     return i >= 0 && j >= 0 && i < row && j < col;
63 }
64 };

```

## 996 Number of Squareful Array

执行用时: 4 ms 内存消耗: 7.9 MB

执行用时: **4 ms** , 在所有 C++ 提交中击败了 **57.86%** 的用户

内存消耗: **7.9 MB** , 在所有 C++ 提交中击败了 **16.07%** 的用户

```

1  /*
2      如果不能枚举， 就自己构造

```

```

3  */
4  class Solution {
5  public:
6      int count = 0;
7      int numSquarefulPerms(vector<int>& nums) {
8          int size = nums.size();
9          vector<bool> visited(size, false);
10         sort(nums.begin(), nums.end());
11         vector<int> path;
12         backtrack(nums, visited, path, 0);
13
14         return count;
15     }
16
17     void backtrack(vector<int>& nums, vector<bool> visited, vector<int> path, int
cur){
18         if(cur == nums.size()){
19             count++;
20             return;
21         }
22
23         for(int i = 0; i < visited.size(); i++){
24             if(visited[i] || (i > 0 && nums[i] == nums[i - 1] && !visited[i - 1]))
25                 continue;
26
27             if(cur == 0){
28                 path.push_back(nums[i]);
29                 visited[i] = true;
30
31                 backtrack(nums, visited, path, cur + 1);
32
33                 path.pop_back();
34                 visited[i] = false;
35             }else{
36                 int sqr = std::sqrt(nums[i] + path[path.size() - 1]);
37                 if(sqr * sqr == nums[i] + path[path.size() - 1]){
38                     path.push_back(nums[i]);
39                     visited[i] = true;
40
41                     backtrack(nums, visited, path, cur + 1);
42
43                     path.pop_back();
44                     visited[i] = false;
45                 }
46             }
47         }
48     }
49 }
50 };

```

# 997 Find the Town Judge

执行结果: **通过** [显示详情 >](#)

[添加备注](#)

执行用时: **3 ms** , 在所有 Java 提交中击败了 **80.53%** 的用户

内存消耗: **45.9 MB** , 在所有 Java 提交中击败了 **80.30%** 的用户

炫耀一下:



```
1 public int findJudge(int n, int[][] trust) {
2     int[] in = new int[n + 1];
3     int[] out = new int[n + 1];
4
5     for(int i = 0; i < trust.length; i++){
6         int A = trust[i][0];
7         int B = trust[i][1];
8
9         in[B]++;
10        out[A]++;
11    }
12
13    for(int i = 1; i <= n; i++){
14        if(in[i] == n - 1 && out[i] == 0)
15            return i;
16    }
17
18    return -1;
19 }
```

执行结果: **通过** [显示详情 >](#)

[添加备注](#)

执行用时: **23 ms** , 在所有 Java 提交中击败了 **13.21%** 的用户

内存消耗: **46.4 MB** , 在所有 Java 提交中击败了 **16.30%** 的用户

炫耀一下:

```
1 public int findJudge(int n, int[][] trust) {
2     if(n == 1 && trust.length == 0)
3         return 1;
```



```
4
5     Map<Integer, Integer> map = new HashMap<>();
6     Set<Integer> set = new HashSet<>();
7
8     for(int i = 0; i < trust.length; i++) {
9         map.put(trust[i][1], map.getDefault(trust[i][1], 0) + 1);
10        set.add(trust[i][0]);
11    }
12
13    List<Integer> judges = new ArrayList<>();
14    for(Integer potentialJudge : map.keySet()){
15        if(map.get(potentialJudge) == n - 1 && !set.contains(potentialJudge))
16            return potentialJudge;
17    }
18
19    return -1;
20 }
```