July 4, 2018

@Mingze X @Chong T @Zhaorui D @Yu S

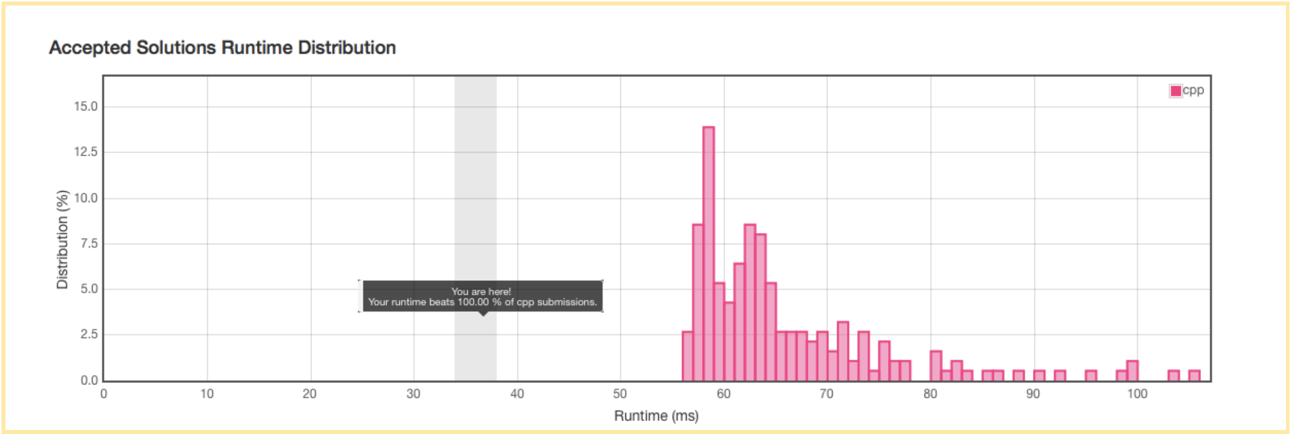
531

1. Mingze's solution:
   lang: JS
   code:

```
var findLonelyPixel = function(picture) {
    let ans = 0;
    let isLonely = arr => {
        let first = arr.indexOf('B');
        return first !== -1 && arr.indexOf('B', first+1) === -
1;
    };
    for (let row of picture) {
        if (isLonely(row)) {
            let rowIndex = row.indexOf('B');
            let col = picture.map(r => r[rowIndex]);
            if (isLonely(col)) ans++;
        }
    }
    return ans;
};
```

2. Goodspeed's solution:
   a. lang: c++
   b. 先全部扫一遍，记录每行的 `B` 的纵坐标跟每列 `B` 的纵坐标。如果多余一个或没有，记录N 或 -1. 最后统计一遍：

```
class Solution {
public:
    int findLonelyPixel(vector<vector<char>>& picture) {
        if(picture.empty() || picture[0].empty()) return 0;
        int n = picture.size(), m = picture[0].size(), ans = 0;
        vector<int> hori(n, -1), vert(m, -1);
        for(int i=0;i<n;++i) for(int j=0;j<m;++j) if(picture[i][j] == 'B'){
            if(hori[i] == -1) hori[i] = j;
            else hori[i] = m;
            if(vert[j] == -1) vert[j] = i;
            else vert[j] = n;
        }
        for(int i=0; i<n; ++i) if(hori[i]>=0 && hori[i]<m){
            if(vert[hori[i]] == i) ++ans;
        }
        return ans;
    }
};
```



上面代码的running time ~ ~

3. by Chong
   O(m * n) time is the least, O(1) space is the least.
   Quite straightforward

```
class Solution {
    public int findLonelyPixel(char[][] picture) {
        if (picture == null || picture.length == 0 || picture
[0] == null || picture[0].length == 0) {
            return 0;
        }

        int num_row_B = 0;
        int first_row_B = -1;
        int res = 0;
        for (int i = 0; i < picture.length; i++) {
            for (int j = 0; j < picture[0].length; j++) {
                if (picture[i][j] == 'B') {
                    num_row_B++;
                    if (num_row_B > 1) {
                        break;
                    }
                    first_row_B = j;
                }
            }
            if (num_row_B == 1) {
                int count_col_B = 0;
                for (int k = 0; k < picture.length; k++) {
                    if (picture[k][first_row_B] == 'B') {
                        count_col_B++;
                        if (count_col_B > 1) {
                            break;
                        }
                    }
                }
                if (count_col_B == 1) {
                    res++;
                }
            }
            num_row_B = 0;
            first_row_B = -1;
        }
        return res;
    }
}
```

722 (tl; dr)

1. Goodspeed's solution:
   a. lang: c++
   b. 就是烦了点，细心即可：

```
class Solution {
public:
    vector<string> removeComments(vector<string>& source) {
        vector<string> ans;
        if(source.empty()) return ans;
        string whole = "", ref = "";
        for(string s: source) whole += s + "\n";
        cout<<whole.size()<<endl;
        int start = 0;
        while(start < whole.size()){
            cout << start << endl;
            auto i = whole.find("//", start);
            if(i == string::npos) i = whole.size();
            auto j = whole.find("/*", start);
            if(j == string::npos) j = whole.size();
            if(i<=j){
                ref += whole.substr(start, i-start);
                auto k = whole.find("\n", i);
                if(k == string::npos) k = whole.size();
                start = k;
            }
            else{
                ref += whole.substr(start, j-start);
                auto k = whole.find("*/", j+2);
                if(k==string::npos) k = whole.size();
                start = k + 2;
            }
        }
        auto i = ref.find_first_not_of("\n");
        if(i==string::npos) i = ref.size();
        while(i < ref.size()){
            auto j = ref.find_first_of("\n", i);
            if(j==string::npos) j = ref.size();
            ans.push_back(ref.substr(i, j-i));
            i = ref.find_first_not_of("\n", j);
            if(i==string::npos) return ans;
        }
        return ans;
    }
};
```

204

1. Mingze's solution:
   lang: JS
   code:

```
var countPrimes = function(n) {
    let ps = new Array(n).fill(1);
    ps[0] = 0;
    ps[1] = 0;
    for (let i = 2; i < n; i++) {
        for (let j = 2; i * j < n; j++) {
            ps[i*j] = 0;
        }
    }
    return ps.reduce((p, a) => (a+p), 0);
};
```

2. GoodSpeed's solution:
    a. lang: c++
    b. 筛：

```
class Solution {
public:
    int countPrimes(int n) {
        vector<int> P(n, 0);
        int ans=0;
        for(int i=2;i<n;++i) if(!P[i]){
            ++ans;
            for(int j=2;j*i<n;++j) P[i*j] = 1;
        }
        return ans;
    }
};
```

2. Same idea as above, remove multiples from a 1 d array.

750

1. GoodSpeed's solution:
    a. lang: c++
    b. 对于每一行，记录1的横坐标， 然后计数：

```
class Solution {
public:
    int countCornerRectangles(vector<vector<int>>& grid) {
        if(grid.empty() || grid[0].empty()) return 0;
        int n = grid.size(), m = grid[0].size(), ans = 0;
        vector<unordered_set<int>> G(n);
        for(int i=0;i<n;++i) for(int j=0;j<m;++j) if(grid[i]
[j]) G[i].insert(j);
        for(int i=0;i<n;++i) for(int j=i+1;j<n;++j){
            int cnt = 0;
            for(auto p: G[i]) if(G[j].count(p)) ++cnt;
            ans += cnt * (cnt-1)/2;
        }
        return ans;
    }
};
```

- [ ] Check other people's solution (leetcode discussion) @Zebo L

2. by Chong
   Count number of horizontal lines with the same width, calculate the combination using C2/n.
   Time complexity = O(m * n * n), Space complexity = O(1)

```
class Solution {
    public int countCornerRectangles(int[][] grid) {
        if (grid == null || grid.length == 0 || grid[0] == nul
l || grid[0].length == 0) {
            return 0;
        }

        int res = 0;

        for (int left = 0; left < grid[0].length - 1; left++)
{
            for (int right = left + 1; right < grid[0].length;
right++) {
                int count = 0;
                for (int i = 0; i < grid.length; i++) {
                    if (grid[i][left] == 1 && grid[i][left] ==
grid[i][right]) {
                        count++;
                    }
                }
                res += count * (count - 1) / 2;
            }
        }
        return res;
    }
}
```

645

1. GoodSpeed’s solution:
    a. lang: c++
    b. idea: 初等数学：

```
class Solution {
public:
    vector<int> findErrorNums(vector<int>& nums) {
        int n = nums.size();
        long S = 0, S2 = 0, Sref = long(n)*(n+1)/2, S2ref = lo
ng(2*n+1)*(n+1)*n/6;
        for(int n: nums) {
            S += long(n);
            S2 += long(n) * n;
        }
        long A = S - Sref, B = (S2 - S2ref)/(S - Sref);
        return vector<int>{int(A+B)/2, int(B-A)/2};
    }
};
```

2. Use the negating technique to label the visited numbers

```
class Solution {
    public int[] findErrorNums(int[] nums) {
        int dup = -1;
        int mis = -1;
        //int[] copy = Arrays.copyOf(nums, nums.length);
        for (int n : nums) {
            if (nums[Math.abs(n) - 1] < 0) {
                dup = Math.abs(n);
                continue;
            }
            nums[Math.abs(n) - 1] = -nums[Math.abs(n) - 1];
        }
        for (int i = 0; i < nums.length; i++) {
            if (nums[i] > 0) {
                mis = i + 1;
                break;
            }
        }
        return new int[]{dup, mis};
    }
}
```

357

1. 1d dp. dp[0] = 1, dp[1] = 10, dp[2] = 9 * 9, dp[3] =dp[2] * 8 ... dp[i] = dp[i - 1] * (10-(i-1))

dp[i] means the number of unique value with i digits

```
class Solution {
    public int countNumbersWithUniqueDigits(int n) {
        if (n < 0 || n >= 11) {
            return 0;
        }
        if (n == 0) {
            return 1;
        }
        int res = 10;
        if (n == 1) {
            return res;
        }
        int cur = 9;
        for (int i = 2; i <= n; i++) {
            cur *= 10 - i + 1;
            res += cur;
        }
        return res;
    }
}
```

2. GoodSpeed's solution:
    a. lang: c++
    b. idea: 排列组合

```
class Solution {
    int comb(int n, int m){
        int ans = 1;
        while(m){
            ans *= n;
            --n;
            --m;
        }
        return ans;
    }
    int numOfUnique(int n){
        if(n == 0) return 1;
        return 9*comb(9, n-1);
    }
public:
    int countNumbersWithUniqueDigits(int n) {
        int ans = 0;
        for(int i=0; i<=n; ++i){
            ans += numOfUnique(i);
        }
        return ans;
    }
};
```

12

1. by Chong
   Since the total possible value is limited, we can deduct the values from the value and add the string (do it from large to small).

```
class Solution {
    public String intToRoman(int num) {
        int[] values = {1000, 900, 500, 400, 100, 90, 50, 40,
10, 9, 5, 4, 1};
        String[] str = {"M", "CM", "D", "CD", "C", "XC","L",
"XL", "X", "IX", "V", "IV", "I"};

        StringBuffer sb = new StringBuffer();

        for (int i = 0; i < values.length; i++) {
            while (num >= values[i]) {
                num -= values[i];
                sb.append(str[i]);
            }
        }

        return sb.toString();
    }
}
```

2. GoodSpeed's solution:
    a. lang: c++
    b. 同上：

```
class Solution {
    vector<string> ref = {"M", "CM", "D", "CD", "C", "XC",
"L", "XL", "X", "IX", "V", "IV", "I"};
    vector<int> deci = {1000, 900, 500, 400, 100, 90, 50, 40,
10, 9, 5, 4, 1};
public:
    string intToRoman(int num) {
        string ans = "";
        for(int i=0;i<13;++i) if(num >= deci[i]){
            for(int j=0; j<num/deci[i]; ++j) ans += ref[i];
            num %= deci[i];
        }
        return ans;
    }
};
```

235

1. by Chong
   straightforward

```
class Solution {
    TreeNode res;
    public TreeNode lowestCommonAncestor(TreeNode root, TreeNo
de p, TreeNode q) {
        dfs(root, p, q);
        return res;
    }
    void dfs(TreeNode root, TreeNode p, TreeNode q) {
        if (root == null) {
            return;
        }
        if ((p.val - root.val) * (q.val - root.val) > 0) {
            dfs(root.left, p, q);
            dfs(root.right, p, q);
        } else {
            res = root;
        }
    }
}
```

2. GoodSpeed's solution:
    a. lang: c++
    b. 同上：but iterative

```
class Solution {
public:
    TreeNode* lowestCommonAncestor(TreeNode* root, TreeNode* 
p, TreeNode* q) {
        while(root){
            if(root->val > p->val && root->val > q->val) root=
root->left;
            else if(root->val <p->val && root->val < q->val) r
oot=root->right;
            else return root;
        }
        return root;
    }
};
```

606

1. by Chong

make sure to understand the question

```
class Solution {
    public String tree2str(TreeNode t) {
        if (t == null) {
            return "";
        }
        String left = tree2str(t.left);
        String right = tree2str(t.right);

        if (left == "" && right == "") {
            return t.val + "";
        }
        if (left == "") {
            return   t.val + "()(" + right + ")";
        }
        if (right == "") {
            return  t.val + "(" + left + ")";
        }
        return t.val + "(" + left + ")(" + right + ")";
    }
}
```

2. Goodspeed’s solution:
   a. lang: c++
   b. 同上： recursion：

```
class Solution {
public:
    string tree2str(TreeNode* t) {
        if(!t) return "";
        string ans = to_string(t->val);
        if(t->left || t->right) ans += "(" + tree2str(t->left)
+ ")";
```

```
        if(t->right) ans += "(" + tree2str(t->right) + ")";
        return ans;
    }
};
```

818

BFS or DP
this is actually only BFS