September 8, 2018 题目：526,780,652,855,228,373,37,8,68,432,604,355,395,404,618

526

1. Memo + dfs + 一些必要的剪枝：

```
typedef vector<int> vi;
vector<vi> P;
vi res;
bool cmp(int i, int j){
    return (int)P[i].size() < (int)P[j].size();
}
class Solution {
    vector<int> init(int N){
        P = vector<vi>(N + 1);
        res.clear();
        for(int i=1; i<=N; ++i){
            res.push_back(i);
            for(int j=1; j<i; ++j) if(!(i%j)) P[i].push_back
(j);
            for(int j=i; j<=N; j+=i) P[i].push_back(j);
        }
        sort(res.begin(), res.end(), cmp);
        return res;
    }
    int dp[16][40000];
    int dfs(int i, int s, const int&N){
        if(i>=N) return 1;
        if(dp[i][s] >= 0) return dp[i][s];
        dp[i][s] = 0;
        for(int j: P[res[i]]) if(!(1&(s>>(j-1)))) dp[i][s] += 
dfs(i+1, s | 1<<(j-1), N);
        return dp[i][s];
    }
public:
    int countArrangement(int N) {
        init(N);
        memset(dp, -1, sizeof(dp));
        return dfs(0, 0, N);
    }
};
```

780

1. 数学题，简单的analysis即可：

```
class Solution {
public:
    bool reachingPoints(int sx, int sy, int tx, int ty) {
        while(tx>=sx && ty>=sy){
            if(tx > ty){
                if(ty == sy && (tx-sx)%sy == 0) return true;
                tx %= ty;
            }
            else{
                if(tx == sx && (ty-sy)%sx == 0) return true;
                ty %= tx;
            }
        }
        return false;
    }
};
```

652

1. 就是简单做下hash即可：

```
class Solution {
    unordered_map<string, vector<TreeNode *>> res;
    string dfs(TreeNode *root){
        if(!root) return "#";
        string s = to_string(root->val) + "(" + dfs(root->left) + ")";
        s += "(" + dfs(root->right) + ")";
        res[s].push_back(root);
        return s;
    }
public:
    vector<TreeNode*> findDuplicateSubtrees(TreeNode* root) {
        dfs(root);
        vector<TreeNode*> ans;
        for(auto p: res) if(p.second.size() > 1) ans.push_back(p.second.back());
        return ans;
    }
};
```

- [ ] 新式对Tree 做hash 的方法：

```
https://leetcode.com/problems/find-duplicate-subtrees/discuss/112442/C++-15ms-(less-99.76)855
```

228

1. 维护一下区间即可

```
class Solution {
public:
    vector<string> summaryRanges(vector<int>& nums) {
        vector<string> ans;
        for(int i=1, cnt=1; i<=nums.size(); ++i){
            if(i<nums.size() && nums[i] == nums[i-1] + 1) ++cn
t;
            else{
                if(cnt == 1) ans.push_back(to_string(nums[i-
1]));
                else ans.push_back(to_string(nums[i-1]-cnt+1)
+ "->" + to_string(nums[i-1]));
                cnt = 1;
            }
        }
        return ans;
    }
};
```

373

1. 这搓B方法都能过：

```
class Solution {
    typedef pair<int, int> ii;
public:
    vector<pair<int, int>> kSmallestPairs(vector<int>& nums1,
vector<int>& nums2, int k) {
        vector<pair<int, int>> ans;
        if(nums1.empty() || nums2.empty()) return ans;
        set<ii> Q;
        int n = nums1.size(), m = nums2.size();
        Q.insert(ii(nums1[0] + nums2[0], 0));
        while(k && !Q.empty()){
            int i = Q.begin()->second/m, j = Q.begin()->second
%m;
            ans.push_back(ii(nums1[i], nums2[j]));
            Q.erase(Q.begin());
            if(i+1 < n) Q.insert(ii(nums1[i+1]+nums2[j], (i+1)
*m + j));
            if(j+1 < m) Q.insert(ii(nums1[i]+nums2[j+1], i*m +
j + 1));
            --k;
        }
        return ans;
    }
};
```

37

1. Standard DFS:

```
class Solution {
    #define CI(c) int((c) - '1')
    #define IC(i) int((i) + '1')
    typedef vector<bool> vb;
    vector<vb> stat;
    vector<int> res;
    bool solve(int k, vector<vector<char>>& B){
        if(k >= res.size()) return true;
        int i = res[k]/9, j = res[k]%9;
        for(int r=0; r<9; ++r) if(!stat[i][r] && !stat[9+j][r]
&& !stat[18 + (i/3)*3 + j/3][r]){
            stat[i][r] = stat[9+j][r] = stat[18 + (i/3)*3 + j/
3][r] = true;
            B[i][j] = IC(r);
            if(solve(k+1, B)) return true;
            stat[i][r] = stat[9+j][r] = stat[18 + (i/3)*3 + j/
3][r] = false;
        }
        return false;
    }
public:
    void solveSudoku(vector<vector<char>>& board) {
        stat = vector<vb>(27, vb(9, false));
        for(int i=0; i<9; ++i) for(int j=0; j<9; ++j){
            if(board[i][j] == '.') res.push_back(i * 9 + j);
            else stat[i][CI(board[i][j])] = stat[9+j][CI(board[i][j])] = stat[18 + (i/3)*3 + j/3][CI(board[i][j])] = true;
        }
        solve(0, board);
    }
};
```

8

1. 注意允许leading zero：

```
class Solution {
    int opt(char c){
        if(c==' ') return 0;
        if(c=='+' || c=='-') return 1;
        if(c>='0' && c<='9') return 2;
        return 3;
    }
    int tr[3][4] = {
        {0, 1, 2, 3},
        {3, 3, 2, 3},
        {3, 3, 2, 3}
    };

public:
    int myAtoi(string str) {
        long val = 0, sign = 1, state = 0;
        str += ' ';
        for(char c: str){
            state = tr[state][opt(c)];
            if(state==3) return val * sign;
            if(state==1 && c=='-') sign = -1;
            if(state==2) val = val*10 + int((c)-'0');
            if(val*sign > long(INT_MAX)) return INT_MAX;
            if(val*sign < long(INT_MIN)) return INT_MIN;
        }
        return val * sign;
    }
};
```

68

1. 烦一点，没什么大不了的：

```
class Solution {
public:
    vector<string> fullJustify(vector<string>& words, int W) {
        vector<string> ans;
        int n = words.size();
        int i=0;
        while(i<n){
            int j = i+1, current = (int)words[i].size();
            while(j<n && current + (int)words[j].size() + 1<=
W) current += (int)words[j++].size() + 1;
            int k = j - i - 1, m = W - current;
            if(!k) ans.push_back(words[i] + string(m, ' '));
            else if(j<n){
                string tmp = words[i];
                for(int l=0; l<m%k; ++l) tmp += string((m+k-1)/k + 1, ' ') + words[i+l+1];
                for(int l=m%k; l<k; ++l) tmp += string(m/k + 1, ' ') + words[i+l+1];
                ans.push_back(tmp);
            }
            else{
                string tmp = words[i];
                for(int l=0; l<k; ++l) tmp += " " + words[i+l+1];
                tmp += string(m, ' ');
                ans.push_back(tmp);
            }
            i = j;
        }
        return ans;
    }
};
```

432

1. 烦得一B系列：用linked list 记录value 出现次数

```
class AllOne {
    typedef pair<int, int> ii;
    list<ii> cnt;
    unordered_map<string, list<ii>::iterator> pos;
    unordered_map<int, unordered_set<string>> keys;
public:
    AllOne() {}
    void inc(string key) {
        if(!pos.count(key)){
            if(!cnt.empty() && cnt.begin()->first==1) cnt.begi
n()->second++;
            else cnt.push_front(ii(1, 1));
            pos[key] = cnt.begin();
            keys[1].insert(key);
        }
        else{
            auto it = pos[key];
            int val = it->first;
            --it->second;
            if(!it->second) it = cnt.erase(it);
            else ++it;
            if(it!=cnt.end() && it->first==val+1) it->second+
+;
            else it = cnt.insert(it, ii(val+1, 1));
            pos[key] = it;
            keys[val].erase(key);
            keys[val+1].insert(key);
        }
    }
    void dec(string key) {
        if(!pos.count(key)) return;
        auto it = pos[key];
        int val = it->first;
        keys[val].erase(key);
        it->second--;
        if(!it->second) it = cnt.erase(it);
        if(val == 1) pos.erase(key);
        else{
            auto it_ = it;
            --it_;
            if(it!=cnt.begin() && it_->first == val-1){
                --it;
                it->second++;
            }
            else it = cnt.insert(it, ii(val-1, 1));
            pos[key] = it;
            keys[val-1].insert(key);
        }
    }
    string getMaxKey() {
        if(cnt.empty()) return "";
        return *keys[cnt.back().first].begin();
    }
    string getMinKey() {
        if(cnt.empty()) return "";
        return *keys[cnt.front().first].begin();
    }
};
```

194

FUCKING BASH

604

1. Easy, 记录index即可

```
class StringIterator {
    string s;
    int i, cnt;
public:
    StringIterator(string compressedString): s(compressedStrin
g), i(0), cnt(stoi(compressedString.substr(1))) {}
```

```
    char next() {
        if(i<s.size()){
            char c = s[i];
            --cnt;
            if(!cnt){
                ++i;
                while(i<s.size() && isdigit(s[i])) ++i;
                if(i<s.size()) cnt = stoi(s.substr(i+1));
            }
            return c;
        }
        return ' ';
    }
    bool hasNext() { return i<s.size(); }
};
```

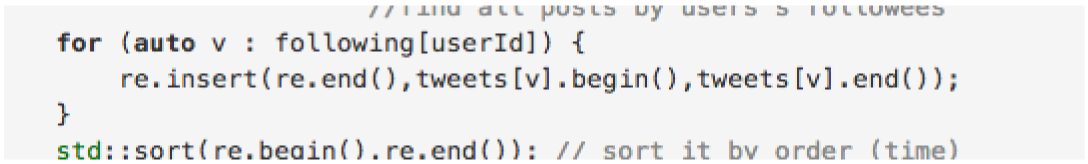
355

1. 本办法，把该显示的feed都记下来：

```
class Twitter {
    typedef pair<int, int> ii;
    int tstamp;
    unordered_map<int, set<ii, greater<ii>>> post, display;
    unordered_map<int, unordered_set<int>> follower;
public:
    Twitter(): tstamp(0) {}
    void postTweet(int userId, int tweetId) {
        post[userId].insert(ii(tstamp, tweetId));
        display[userId].insert(ii(tstamp, tweetId));
        for(int k: follower[userId]) display[k].insert(ii(tsta
mp, tweetId));
        ++tstamp;
    }
    vector<int> getNewsFeed(int userId) {
        auto it = display[userId].begin();
        int k = 10;
        vector<int> ans;
        while(k && it!=display[userId].end()){
            ans.push_back(it->second);
            ++it;
            --k;
        }
        return ans;
    }
    void follow(int followerId, int followeeId) {
        if(followerId == followeeId) return;
        follower[followeeId].insert(followerId);
        for(auto p: post[followeeId]) display[followerId].inse
rt(p);
    }

    /** Follower unfollows a followee. If the operation is inv
alid, it should be a no-op. */
    void unfollow(int followerId, int followeeId) {
        if(followerId == followeeId) return;
        follower[followeeId].erase(followerId);
        for(auto p: post[followeeId]) display[followerId].eras
e(p);
    }
};
```

- 头一次看到， insert还能这么用，长见识了：



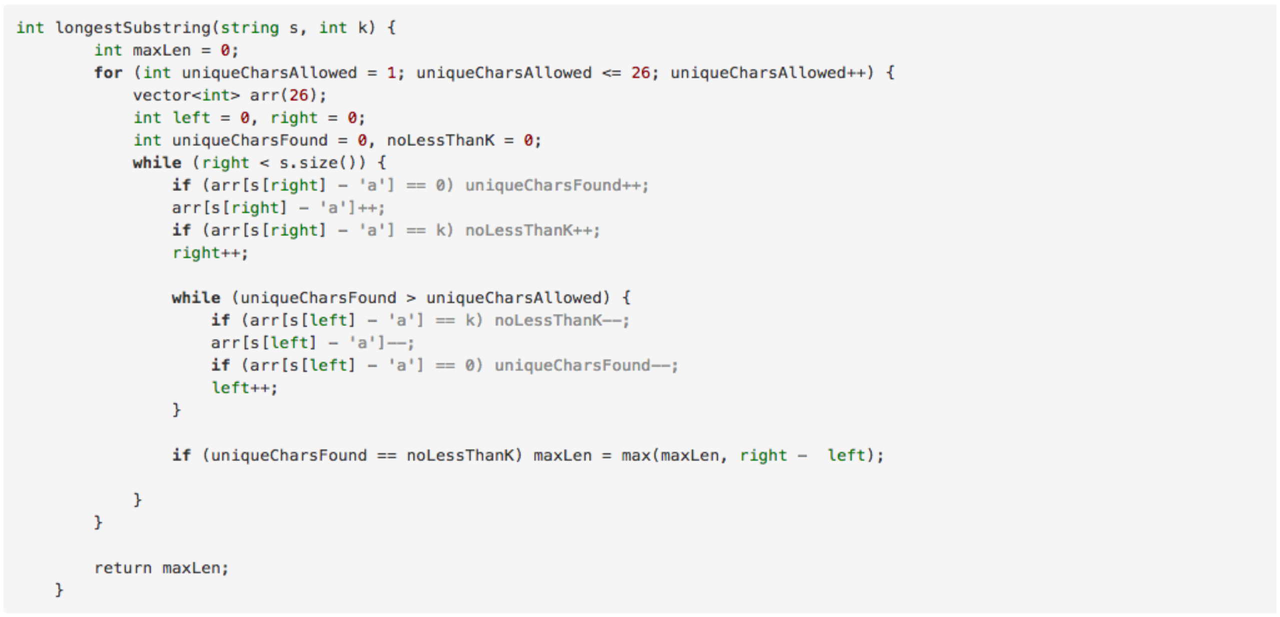
```
        //find all posts by user's followees
    for (auto v : following[userId]) {
        re.insert(re.end(),tweets[v].begin(),tweets[v].end());
    }
    std::sort(re.begin(),re.end()); // sort it by order (time)
```

395

1. DFS:

```
class Solution {
    int res;
    void dfs(string s, int &k){
        if(s.size() <= res) return;
        unordered_map<char, vector<int>> pos;
        for(int i=0; i<s.size(); ++i) pos[s[i]].push_back(i);
        vector<int> sep;
        bool ok = false;
        for(auto p: pos) {
            if(p.second.size()<k){
                for(int j: p.second) sep.push_back(j);
            }
            else ok = true;
        }
        if(sep.empty()){
            res = (int)s.size();
            return;
        }
        if(ok){
            sep.push_back(-1);
            sep.push_back(s.size());
            sort(sep.begin(), sep.end());
            for(int i=1; i<sep.size(); ++i) dfs(s.substr(sep[i
-1]+1, sep[i]-sep[i-1]-1), k);
        }
    }
public:
    int longestSubstring(string s, int k) {
        res = 0;
        dfs(s, k);
        return res;
    }
};
```

- [ ] 下面这个好方法：@Zebo L



```
int longestSubstring(string s, int k) {
        int maxLen = 0;
        for (int uniqueCharsAllowed = 1; uniqueCharsAllowed <= 26; uniqueCharsAllowed++) {
            vector<int> arr(26);
            int left = 0, right = 0;
            int uniqueCharsFound = 0, noLessThanK = 0;
            while (right < s.size()) {
                if (arr[s[right] - 'a'] == 0) uniqueCharsFound++;
                arr[s[right] - 'a']++;
                if (arr[s[right] - 'a'] == k) noLessThanK++;
                right++;

                while (uniqueCharsFound > uniqueCharsAllowed) {
                    if (arr[s[left] - 'a'] == k) noLessThanK--;
                    arr[s[left] - 'a']--;
                    if (arr[s[left] - 'a'] == 0) uniqueCharsFound--;
                    left++;
                }

                if (uniqueCharsFound == noLessThanK) maxLen = max(maxLen, right -  left);

            }
        }

        return maxLen;
    }
```

404

1. Easy recursion:

```
class Solution {
    int res;
    inline bool isLeaf(TreeNode *r) {return r&&!r->left&&!r->r
ight;}
    void dfs(TreeNode *root, bool left){
        if(!root) return;
        if(isLeaf(root) && left) res+=root->val;
        dfs(root->left, true);
        dfs(root->right, false);
    }
public:
    int sumOfLeftLeaves(TreeNode* root) {
        res = 0;
        dfs(root, false);
        return res;
    }
};
```

618

SQL Hard, 九牛二虎

```
SELECT a.name as America, b.name as Asia, c.name as Europe FROM
(SELECT @arank := @arank + 1 AS Id, s.name AS name FROM
(SELECT name FROM student where continent = 'America' ORDER BY name) AS s, (SELECT @arank := 0) AS r) AS a
LEFT JOIN
(SELECT @brank := @brank + 1 AS Id, s.name AS name FROM
(SELECT name FROM student where continent = 'Asia' ORDER BY name) AS s, (SELECT @brank := 0) AS r) AS b ON a.Id = b.Id
LEFT JOIN
(SELECT @crank := @crank + 1 AS Id, s.name AS name FROM
(SELECT name FROM student where continent = 'Europe' ORDER BY name) AS s, (SELECT @crank := 0) AS r) AS c ON a.Id = c.Id
```

GET INDEX 用：

1. `SELECT @RANK`

```
SELECT @rank := @rank + 1 AS Id, col FROM tablename, (SELECT @rank := 0) AS r
```

2. `ROW_NUMBER()`:

```
SELECT ROW_NUMBER() OVER(ORDER BY YourColumn) AS Rank FROM tablename
```