August 28, 2018 题目：758,734,487,408,533,151,367,377,324,3,292,770,15,518,49

758

1. 做过，我的做法是maintain intervals：

```
class Solution {
public:
    string boldWords(vector<string>& words, string S) {
        int n = S.size();
        map<int, int> B;
        B[-2] = -1;
        B[n+5] = n+6;
        for(string s: words){
            auto i = S.find(s);
            while(i!=string::npos){
                auto it = --B.upper_bound(int(i));
                if(it->second < int(i)) ++it;
                int start = min((int)it->first, (int)i), end =
i + s.size();
                while(it->first <= end){
                    end = max(end, it->second);
                    it = B.erase(it);
                }
                B[start] = end;
                i = S.find(s, i+1);
            }
        }
        int start = 0;
        string ans;
        for(auto it=++B.begin(); it!=--B.end(); ++it){
            ans += S.substr(start, it->first-start) + "<b>" + S.substr(it->first, it->second-it->first) + "</b>";
            start = it->second;
        }
        ans += S.substr(start, n-start);
        return ans;
    }
};
```

734

1. 做过，我在这儿是直接Brute Force，Union Find 可能更快些：

```
class Solution {
public:
    bool areSentencesSimilar(vector<string>& words1, vector<string>& words2, vector<pair<string, string>> pairs) {
        if(words1.size() != words2.size()) return false;
        unordered_map<string, unordered_set<string>> sim;
        for(auto p: pairs) sim[p.first].insert(p.second);
        for(int i=0; i<words1.size(); ++i) if(words1[i] != words2[i]){
            if((!sim.count(words1[i]) || !sim[words1[i]].count(words2[i])) && (!sim.count(words2[i]) || !sim[words2[i]].count(words1[i]))) return false;
        }
        return true;
    }
};
```

487

1. 做过，sliding Window：

```
class Solution {
public:
    int findMaxConsecutiveOnes(vector<int>& nums) {
        int i = -1, j = -1, k = 0, n = nums.size(), ans = 0;
        while(k<n && nums[k]) ++k;
        if(k >= n) return n;
        while(i<n && k<n){
            i = j+1;
            j = k;
            k = j+1;
            while(k<n && nums[k]) ++k;
            ans = max(ans, k-i);
        }
        return ans;
    }
};
```

408

1. 逐个字母比较即可：

```
class Solution {
public:
    bool validWordAbbreviation(string word, string abbr) {
        int i=0, j=0;
        while(i<word.size() && j<abbr.size()){
            if(isdigit(abbr[j])){
                int cnt = stoi(abbr.substr(j));
                if(!cnt) return false;
                i += cnt;
                j += (int)to_string(cnt).size();
            }
            else if(word[i++] != abbr[j++]) return false;
        }
        return (i==word.size() && j==abbr.size());
    }
};
```

533

1. 就是计较烦而已，带删减的BruteForce：

```
class Solution {
public:
    int findBlackPixel(vector<vector<char>>& picture, int N) {
        int n = picture.size(), m = picture[0].size(), ans =
0;
        unordered_set<int> col;
        unordered_map<string, int> group;
        for(int j=0; j<m ; ++j){
            int cnt = 0;
            for(int i=0; i<n; ++i) if(picture[i][j] == 'B') ++
cnt;
            if(cnt == N) col.insert(j);
        }
        for(int i=0; i<n; ++i){
            string s;
            int cnt = 0;
            for(char c: picture[i]){
                s += c;
                if(c == 'B') ++cnt;
            }
            if(cnt == N) ++group[s];
        }
        for(auto p: group) if(p.second == N){
            for(int j=0; j<m; ++j) if(p.first[j] == 'B' && co
l.count(j)) ans += N;
        }
        return ans;
    }
};
```

151

1. PYTHON 就是为这种题设计的，Short is Beauty 系列：

```
class Solution(object):
    def reverseWords(self, s):
        return " ".join(s.strip().split()[::-1])
```

367

1. Short is Beauty 系列：

```
class Solution(object):
    def isPerfectSquare(self, num):
        return int(num ** 0.5) ** 2 == num
```

377

1. DP, 比背包还要简单：

```
class Solution {
public:
    int combinationSum4(vector<int>& nums, int target) {
        vector<int> dp(target+1, 0);
        dp[0] = 1;
        for(int i=0; i<target; ++i) for(int n: nums) if(i+n<=t
arget) dp[i+n] += dp[i];
        return dp[target];
    }
};
```

324

1. 神奇的算法：(My Own quick selection is so slow!)
    - https://leetcode.com/problems/wiggle-sort-ii/discuss/77682/Step-by-step-explanation-of-index-mapping-in-Java
    - Put lager one into odd slots, and fill from beginning.
    - Put smaller one into even slots, and fill from end.
    - 最终要让上面两个fill 相交

```
class Solution {
    void kthElement(vector<int>& A, int l, int r, int k){
        if(l>=r) return;
        int s = A[l];
        int i = l, j = r;
        while(j>i){
            while(i<j && A[j]>=s) --j;
            if(i<j) A[i++] = A[j];
            while(i<j && A[i]<=s) ++i;
            if(i<j) A[j--] = A[i];
        }
        A[i] = s;
        if(i - l==k) return;
        else if(i-l > k) kthElement(A, l, i-1, k);
        else kthElement(A, i+1, r, k - (i-l) - 1);
    }
public:
    void wiggleSort(vector<int>& nums) {
        int n = nums.size(), k = (nums.size()-1)/2;
        nth_element(nums.begin(), nums.begin()+k, nums.end());
        int o = nums[k], odd = 1, even = ((n-1)%2? n-2:n-1), i
= 0;
        while(i<n){
            if(nums[i]<o && (i%2 || i < even)) {
                swap(nums[i], nums[even]);
                while(even>=0 && nums[even]<o) even-=2;
            }
            else if(nums[i]>o && (i%2==0 || i>odd)){
                swap(nums[i], nums[odd]);
                while(odd<n && nums[odd]>o) odd+=2;
            }
            else ++i;
        }
    }
};
```

3

1. 典型sliding window：

```
class Solution {
public:
    int lengthOfLongestSubstring(string s) {
        map<char, int> cnt;
        int i=0, j=-1, n=s.size(), ans = 0;
        while(i<n && j<n){
            ++j;
            while(j<n){
                ++cnt[s[j]];
                if(cnt[s[j]] > 1) break;
                ++j;
            }
            ans = max(ans, j-i);
            while(i<j){
                --cnt[s[i]];
                ++i;
                if(cnt[s[i-1]] == 1) break;
            }
        }
        return ans;
    }
};
```
2. 标记序列开始位置：
```
class Solution {
public:
    int lengthOfLongestSubstring(string s) {
        vector<int> pos(300, -1);
        int start=-1, n = s.size(), ans = 0;
        for(int i=0; i<n; ++i){
            if(pos[s[i]] > start){
                start = pos[s[i]];
            }
            pos[s[i]] = i;
            ans = max(ans, i-start);
        }
        return ans;
    }
};
```

292

1. Short is Beauty 系列：
```
class Solution {
public:
    bool canWinNim(int n) {
        return n%4;
    }
};
```

770

1. 又是这道题，第二遍还是没有看懂题

15

1. 没啥技术含量：

```
class Solution {
public:
    vector<vector<int>> threeSum(vector<int>& nums) {
        sort(nums.begin(), nums.end());
        vector<vector<int>> ans;
        int n = nums.size();
        vector<int> next(n, n), prev(n, -1);
        for(int i=n-2; i>=0; --i) next[i] = (nums[i]==nums[i+1]? next[i+1]: i+1);
        for(int i=1; i<n; ++i) prev[i] = (nums[i]==nums[i-1]? prev[i-1]: i-1);
        if(n < 3) return ans;
        for(int i=0; i<n; i=next[i]){
            int j = i+1, k= n-1;
            while(j<k){
                if(nums[j] + nums[k] == -nums[i]) ans.push_back({nums[i], nums[j], nums[k]});
                if(nums[j] + nums[k] > -nums[i]) k = prev[k];
                else j = next[j];
            }
        }
        return ans;
    }
};
```

518

1. DFS + MEMO:

```
class Solution {
    int dp[501][5001];
    int dfs(int i, int a, vector<int>& C){
        if(!a) return 1;
        if(i==C.size() || a < C[i]) return 0;
        if(dp[i][a] >=0 ) return dp[i][a];
        dp[i][a] = 0;
        for(int k=0; k<=a/C[i]; ++k) dp[i][a] += dfs(i+1, a-k*
C[i], C);
        return dp[i][a];
    }
public:
    int change(int amount, vector<int>& coins) {
        sort(coins.begin(), coins.end());
        memset(dp, -1, sizeof(dp));
        return dfs(0, amount, coins);
    }
};
```

2. 前面为了避免`O(n*m)`，复杂度，而绕了一个大圈，其实直接`dp`即可：这题跟上面某个题很类似，但要注意次序

```
class Solution {
public:
    int change(int amount, vector<int>& coins) {
        vector<int> dp(amount+1, 0);
        dp[0] = 1;
        for(int k: coins) for(int i=0; i<=amount-k; ++i) dp[i+
k] += dp[i];
        return dp[amount];
    }
};
```

49

1. 简单的hash：

```
class Solution {
    #define CI(c) int((c) - 'a')
    typedef vector<int> vi;
public:
    vector<vector<string>> groupAnagrams(vector<string>& strs)
{
        map<vi, vector<string>> col;
        vector<vector<string>> ans;
        for(string &s: strs){
            vi cnt(26, 0);
            for(char &c: s) cnt[CI(c)]++;
            col[cnt].push_back(s);
        }
        for(auto p: col) ans.push_back(p.second);
        return ans;
    }
};
```

647

1. 取一点，然后左右延伸即可：

```
class Solution {
public:
    int countSubstrings(string s) {
        int n = s.size(), cnt = 0;
        for(int i=0; i<n; ++i){
            for(int j=0; i-j>=0 && i+j<n && s[i-j]==s[i+j]; ++
j) ++cnt;
            for(int j=0; i-j>=0 && i+j+1<n && s[i-j]==s[i+j+
1]; ++j) ++cnt;
```

```
        }
        return cnt;
    }
};
```