September 6, 2018 题目：17,361,542,490,5,243,502,301,219,237,843,311,728,61,171

17

1. 已知总数，求所有可能的问题：

```
class Solution {
    vector<string> ref = {"abc", "def", "ghi", "jkl", "mno", "pqrs", "tuv", "wxyz"};
public:
    vector<string> letterCombinations(string digits) {
        int prod = 1, n = digits.size();
        vector<string> candidates, ans;
        if(!n) return ans;
        for(char c: digits) {
            candidates.push_back(ref[int(c-'2')]);
            prod *= (int)ref[int(c-'2')].size();
        }
        for(int k=0; k<prod; ++k){
            int m = k;
            string res;
            for(string s: candidates){
                res += s[m % (int)s.size()];
                m /= (int)s.size();
            }
            ans.push_back(res);
        }
        return ans;
    }
```

```
};
```

361

1. DP:

```
class Solution {
    typedef vector<int> vi;
public:
    int maxKilledEnemies(vector<vector<char>>& G) {
        if(G.empty() || G[0].empty()) return 0;
        int n = G.size(), m = G[0].size(), ans = 0;
        vector<vi> H(n, vi(m, 0));
        for(int i=0; i<n; ++i){
            vi pos;
            for(int j=0, cnt=0; j<=m; ++j){
                if(j<m && G[i][j]=='0') pos.push_back(j);
                else if(j<m && G[i][j] == 'E') ++cnt;
                else{
                    for(int k: pos) H[i][k] = cnt;
                    pos.clear();
                    cnt = 0;
                }
            }
        }
        for(int j=0; j<m; ++j){
            vi pos;
            for(int i=0, cnt=0; i<=n; ++i){
                if(i<n && G[i][j]=='0') pos.push_back(i);
                else if(i<n && G[i][j]=='E') ++cnt;
                else{
                    for(int k: pos) ans = max(ans, cnt + H[k]
[j]);
```

```
                    pos.clear();
                    cnt = 0;
                }
            }
        }
        return ans;
    }
};
```

843

542

1. BFS 走一遍即可：

```
class Solution {
    int di[4]={1,-1,0,0}, dj[4]={0,0,1,-1};
    typedef pair<int, int> ii;
public:
    vector<vector<int>> updateMatrix(vector<vector<int>>& matrix) {
        int n = matrix.size(), m = matrix[0].size();
        vector<vector<int>> ans(n, vector<int>(m, -1));
        queue<ii> Q;
        for(int i=0;i<n;++i) for(int j=0;j<m;++j) if(matrix[i][j]==0){
            ans[i][j] = 0;
            Q.push(ii(i, j));
        }
        while(!Q.empty()){
            int i = Q.front().first, j = Q.front().second;
            Q.pop();
            for(int k=0;k<4;++k){
```

```
                int i1 = i+di[k], j1= j+dj[k];
                if(i1>=0 && i1<n && j1>=0 && j1<m && ans[i1][j
1]<0){
                    ans[i1][j1] = ans[i][j] + 1;
                    Q.push(ii(i1, j1));
                }
            }
        }
        return ans;
    }
};
```

490

1. 一直看错题：用map 做hash应该更快

```
class Solution {
    typedef pair<int, int> ii;
    int dr[4] = {1, -1, 0, 0};
public:
    bool hasPath(vector<vector<int>>& M, vector<int>& S, vecto
r<int>& D) {
        assert(!M.empty() && !M[0].empty());
        int n = M.size(), m = M[0].size();
        int init = S[0]*m + S[1];
        set<ii> P;
        queue<ii> Q;
        for(int k=0; k<4; ++k){
            P.insert(ii(init, k));
            Q.push(ii(init, k));
        }
        while(!Q.empty()){
```

```
            int i = Q.front().first/m, j = Q.front().first%m,
dir = Q.front().second;
            if(D[0]==i && D[1]==j) return true;
            Q.pop();
            while(i>=0 && i<n && j>=0 && j<m && !M[i][j]){
                i += dr[dir];
                j += dr[3-dir];
            }
            i -= dr[dir];
            j -= dr[3-dir];
            for(int k=0; k<4; ++k) if(k!=dir && !P.count(ii(i*
m + j, k))){
                P.insert(ii(i*m + j, k));
                Q.push(ii(i*m + j, k));
            }
        }
        return false;
    }
};
```

5

1. DP

```
class Solution {
    public String longestPalindrome(String s) {
        boolean[][] dp = new boolean[s.length()][s.length()];
        String res = "";
        for (int i = s.length() - 1; i >= 0; i--) {
            for (int j = i; j < s.length(); j++) {
                if (s.charAt(i) == s.charAt(j) && (j - i < 3 |
| dp[i + 1][j - 1])) {
                    dp[i][j] = true;
```

```
                    if (res == "" || res.length() < j - i + 1)
{
                        res = s.substring(i, j + 1);
                    }
                }
            }
        }
        return res;
    }
}
```

2. 记 prefix :

```
class Solution {
public:
    string longestPalindrome(string s) {
        int n = s.size(), res = 0;
        string ans;
        vector<int> prefix;
        for(int i=0; i<n; ++i){
            vector<int> tmp{i};
            if(i && s[i]==s[i-1]) tmp.push_back(i-1);
            for(int j: prefix) if(j && s[j-1]==s[i]) tmp.push_
back(j-1);
            for(int j: tmp) if(i-j+1 > res){
                res = i-j+1;
                ans = s.substr(j, i-j+1);
            }
            swap(tmp, prefix);
        }
        return ans;
    }
};
```

3. 居然忘了向两头延伸的方法：

```
class Solution {
public:
    string longestPalindrome(string s) {
        int n = s.size(), res = 0, min_start=0;
        if(!n) return "";
        for(int i=0; i<n; ++i){
            int l = i, r = i;
            while(l-1>=0 && r+1<n && s[l-1]==s[r+1]) --l, ++r;
            if(res < r-l+1){
                res = r-l+1;
                min_start = l;
            }
            l = i+1, r = i;
            while(l-1>=0 && r+1<n && s[l-1]==s[r+1]) --l, ++r;
            if(res < r-l+1){
                res = r-l+1;
                min_start = l;
            }
        }
        return s.substr(min_start, res);
    }
};
```

243

1.

```
class Solution {
    public int shortestDistance(String[] words, String word1,
String word2) {
        int w1 = -1;
        int w2 = -1;
```

```
        int min = Integer.MAX_VALUE;
        for (int i = 0; i < words.length; i++) {
            if (word1.equals(words[i])) {
                w1 = i;
            }
            if (word2.equals(words[i])) {
                w2 = i;
            }
            if (w1 != -1 && w2 != -1) {
                min = Math.min(min, Math.abs(w1 - w2));
            }
        }

        return min;
    }
}
```

2. One pass:

```
class Solution {
public:
    int shortestDistance(vector<string>& words, string word1,
string word2) {
        int x1 = -2*(int)words.size(), x2 = -4*(int)words.size
(), n=words.size(), ans = INT_MAX;
        for(int i=0; i<n; ++i){
            if(words[i]==word1){
                x1 = i;
                ans = min(ans, abs(x1 - x2));
            }
            if(words[i]==word2){
                x2 = i;
                ans = min(ans, abs(x1 - x2));
```

```
            }
        }
        return ans;
    }
};
```

502

1.

```
class Solution {
    public int findMaximizedCapital(int k, int W, int[] Profit
s, int[] Capital) {
        PriorityQueue<int[]> pqProfit = new PriorityQueue<>
((a, b) -> a[0] - b[0]);
        PriorityQueue<int[]> pqCapital = new PriorityQueue<>
((a, b) -> b[1] - a[1]);
        for (int i = 0; i < Profits.length; i++) {
            pqProfit.offer(new int[]{Capital[i], Profits[i]});
        }

        while (k-- > 0) {
            while (!pqProfit.isEmpty() && pqProfit.peek()[0] <
= W) {
                pqCapital.offer(pqProfit.poll());
            }

            if (pqCapital.isEmpty()) {
                break;
            }
            W += pqCapital.poll()[1];
        }
        return W;
```

```
    }
}
```

2. `map`: 用固定size的priority queue可能更快些

```
class Solution {
public:
    int findMaximizedCapital(int k, int W, vector<int>& Ps, ve
ctor<int>& Cl) {
        map<int, int, greater<int>> C;
        map<int, vector<int>> R;
        for(int i=0; i<Ps.size(); ++i) R[Cl[i]].push_back(i);
        while(k){
            for(auto it=R.begin(); it!=R.end() && it->first<=
W; it=R.erase(it)){
                for(int j: it->second) ++C[Ps[j]];
            }
            if(C.empty()) return W;
            int m = C.begin()->first;
            W += m;
            --C[m];
            if(!C[m]) C.erase(m);
            --k;
        }
        return W;
    }
};
```

301

1. BFS

```
class Solution {
    public List<String> removeInvalidParentheses(String s) {
        List<String> res = new ArrayList<>();
```

```
        Queue<String> q = new LinkedList<>();
        q.offer(s);
        Set<String> visited = new HashSet<>();
        visited.add(s);
        boolean flag = false;
        while (!q.isEmpty()) {
            String cur = q.poll();
            if (isValid(cur)) {
                res.add(cur);
                flag = true;
            }
            if (flag) {
                continue;
            }

            for (int i = 0; i < cur.length(); i++) {
                if (cur.charAt(i) != '(' && cur.charAt(i) !=
')') {
                    continue;
                }
                String t = cur.substring(0, i) + cur.substring
(i + 1);
                if (!visited.contains(t)) {
                    q.offer(t);
                    visited.add(t);
                }
            }
        }
        return res;
    }
```

```
    public boolean isValid(String s) {
        int count = 0;
        for (int i = 0; i < s.length(); i++) {
            if (s.charAt(i) == '(') {
                count++;
            } else if (s.charAt(i) == ')') {
                count--;
            }
            if (count < 0) return false;
        }
        return count == 0;
    }
}
```

2. 剪枝加BFS：

```
class Solution {
    bool isValid(const string&s){
        int cnt = 0;
        for(char c: s){
            if(c=='(') ++cnt;
            else if(c==')') --cnt;
            if(cnt<0) return false;
        }
        return !cnt;
    }
public:
    vector<string> removeInvalidParentheses(string s) {
        int l = -1;
        unordered_set<int> remove;
```

```
        for(int i=0; i<s.size() && s[i]!='('; ++i) if(s[i]==')') remove.insert(i);
        for(int i=s.size()-1; i>=0 && s[i]!=')'; --i) if(s[i]=='(') remove.insert(i);
        string p;
        for(int i=0; i<s.size(); ++i) if(!remove.count(i)) p.push_back(s[i]);
        swap(s, p);
        vector<string> ans;
        unordered_set<string> S{s};
        queue<string> Q;
        Q.push(s);
        while(!Q.empty()){
            string t = Q.front();
            Q.pop();
            if(l>=0 && t.size()<l) return ans;
            else if(isValid(t)){
                if(l==-1) l = (int)t.size();
                ans.push_back(t);
                continue;
            }
            for(int i=0; i<t.size(); ++i) if(t[i]=='(' || t[i]==')'){
                string tmp = t.substr(0, i) + t.substr(i+1);
                if(!S.count(tmp)){
                    S.insert(tmp);
                    Q.push(tmp);
                }
            }
        }
        return ans;
```

```
    }
};
```

219

1.

```
class Solution {
    public boolean containsNearbyDuplicate(int[] nums, int k)
{
        Map<Integer, Set<Integer>> map = new HashMap<>();

        for (int i = 0; i < nums.length; i++) {
            if (!map.containsKey(nums[i])) {
                map.put(nums[i], new HashSet<>());
            } else {
                for (int j = 0; j <= k; j++) {
                    if (i - j < 0) continue;
                    if (map.get(nums[i]).contains(i - j)) {
                        return true;
                    }
                }
            }
            map.get(nums[i]).add(i);
        }
        return false;
    }
}
```

- [ ] 应该可以optimize，回来再check

2. 真的还可以optimize么？

```
class Solution {
public:
    bool containsNearbyDuplicate(vector<int>& nums, int k) {
```

```
        unordered_map<int, int> pos;
        for(int i=0; i<nums.size(); ++i){
            if(pos.count(nums[i]) && i - pos[nums[i]] <= k) re
turn true;
            pos[nums[i]] = i;
        }
        return false;
    }
};
```

237

1. 就是把value 往前挪：

```
class Solution {
public:
    void deleteNode(ListNode* node) {
        while(node->next) {
            node->val = node->next->val;
            if(node->next->next) node = node->next;
            else{
                delete node->next;
                node->next = NULL;
            }
        }
    }
};
```

843

- [x] ~~Solve IT！！！！@Zebo L~~

1. 贪心法凑巧过了，不过心里知道其实只是因为test case 弱而已：

```
class Solution {
    typedef unordered_set<int> ui;
```

```
    typedef vector<int> vi;
    typedef vector<bool> vb;
    int getMaxSetEffect(ui &cur, vector<ui> &candi, vb &used){
        int idx = -1, eff = 0;
        for(int i=0; i<candi.size(); ++i) if(!used[i]) {
            int tmp = 0;
            for(int k: candi[i]) if(!cur.count(k)) ++tmp;
            if(tmp > eff){
                eff = tmp;
                idx = i;
            }
        }
        if(idx == -1) return -1;
        for(int k: candi[idx]) if(!cur.count(k)) cur.insert
(k);
        return idx;
    }
    int getMatches(const string&s, const string&p){
        int ans = 0;
        for(int i=0; i<6; ++i) ans += int(s[i]==p[i]);
        return ans;
    }

public:
    void findSecretWord(vector<string>& wl, Master& master) {
        int n = wl.size(), cnt = 9, idx;
        ui current;
        vector<ui> candidates(n);
        for(int i=0; i<n; ++i) for(int j=i+1; j<n; ++j) for(in
t k=0; k<n; ++k)
```

```
            if(getMatches(wl[i], wl[k]) != getMatches(wl[j], wl[k])) candidates[k].insert(i*n + j);
        vector<int> choice, ans;
        vb used(n, false);
        while((idx = getMaxSetEffect(current, candidates, used)) != -1 && cnt){
            choice.push_back(idx);
            used[idx] = true;
            --cnt;
        }
        cout << choice.size() << endl;
        //assert(choice.size() <= 9);
        map<vi, int> pos;
        for(int i=0; i<n; ++i){
            vi tmp;
            for(int k: choice) tmp.push_back(getMatches(wl[i], wl[k]));
            pos[tmp] = i;
        }
        for(int k: choice){
            ans.push_back(master.guess(wl[k]));
        }
        master.guess(wl[pos[ans]]);
    }
};
```

2. Min Max Histogram:

https://leetcode.com/problems/guess-the-word/discuss/134251/Optimal-MinMax-Solution-(+-extra-challenging-test-cases)

3. 贪心加Random，居然过了：

```
class Solution {
    int match(string&a,string&b){
```

```
        int res=0;
        for(int i=0;i<a.size();i++)
            if(a[i]==b[i])
                res++;
        return res;
    }

public:
    void findSecretWord(vector<string>& wordlist, Master& master) {
        vector<string> candidates = wordlist;
        for(int i = 0; i < 10; i++){
            string s=candidates[rand()%candidates.size()];
            int x = master.guess(s);
            if(x==6)
                return;
            //candidates =  neighbors with match x intersect candidates
            vector<string> intersect;
            for(auto w: wordlist)
                if(match(w, s)==x)
                    if(find(candidates.begin(), candidates.end(), w) != candidates.end())
                        intersect.push_back(w);
            candidates = intersect;
        }
    }
};
```

311

1. 非常基本的CSR sparse matrix 表示法：

```
class Solution {
public:
    vector<vector<int>> multiply(vector<vector<int>>& A, vecto
r<vector<int>>& B) {
        int n = A.size(), k = A[0].size(), m = B[0].size();
        vector<unordered_map<int, int>> MA(n), MB(m);
        for(int i=0; i<n; ++i) for(int j=0; j<k; ++j) if(A[i]
[j]) MA[i][j] = A[i][j];
        for(int j=0; j<m; ++j) for(int i=0; i<k; ++i) if(B[i]
[j]) MB[j][i] = B[i][j];
        vector<vector<int>> ans(n, vector<int>(m, 0));
        for(int i=0; i<n; ++i) for(int j=0; j<m; ++j) for(auto
x: MA[i]) if(MB[j].count(x.first)) ans[i][j] += x.second * MB
[j][x.first];
        return ans;
    }
};
```

728

1. Short is Beauty 系列只Beat 9% 😓

```
class Solution(object):
    def selfDividingNumbers(self, left, right):
        return [x for x in range(left, right+1) if reduce(lamb
da a, b: a and b, [int(c) and not x%int(c) for c in str(x)])]
```

61

1. 怎么都得数下Node：

```
class Solution {
public:
    ListNode* rotateRight(ListNode* head, int k) {
        if(!head || !head->next) return head;
        ListNode *p = head;
```

```
        int n = 1;
        while(p->next){
            p = p->next;
            ++n;
        }
        p->next = head;
        k = n - k%n;
        while(k){
            p = p->next;
            --k;
        }
        head = p->next;
        p->next = NULL;
        return head;
    }
};
```

2.

```
class Solution {
    public ListNode rotateRight(ListNode head, int k) {
        if (k == 0 || head == null || head.next == null) retur
n head;
        int len = 0;
        ListNode node = head;
        while (node != null) {
            len++;
            node = node.next;
        }
        k = k % len;
        if (k == 0) return head;
        ListNode end = head;
```

```
        int n = len - k;
        while (n-- > 1) {
            end = end.next;
        }

        ListNode newStart = end.next;
        node = newStart;
        // 1->2->3->4->5 k=2
        while (node.next != null) {
            node = node.next;
        }
        end.next = null;
        node.next = head;
        return newStart;

    }
}
```

171

1. Easy:

```
class Solution {
public:
    int titleToNumber(string s) {
        int ans = 0;
        for(char c: s) ans = ans * 26 + int(c-'A'+1);
        return ans;
    }
};
```

2. Short is Beauty 系列 :

```
class Solution(object):
    def titleToNumber(self, s):
```

```
        return sum([(ord(c) - ord('A') + 1) * 26**i  for i, c
in enumerate(s[::-1])])
```

3.

```
class Solution {
    public int titleToNumber(String s) {
        int res = 0;
        for (char c : s.toCharArray()) {
            res = res * 26 + (c - 'A' + 1);
        }
        return res;
    }
}
```