September 4, 2018 题目：359,128,10,362,708,26,241,291,147,540,1,192,468,423,229

359

1. Hash:

```
class Logger {
    unordered_map<string, int> msgs;
public:
    Logger() {}
    bool shouldPrintMessage(int timestamp, string message) {
        if(msgs.count(message) && timestamp < msgs[message] +
10) return false;
        msgs[message] = timestamp;
        return true;
    }
};
```

128

1. Unordered_set, 然后左右搜：

```
class Solution {
public:
    int longestConsecutive(vector<int>& nums) {
        unordered_set<int> S(nums.begin(), nums.end());
        int ans = 0;
        while(!S.empty()){
            int k = *S.begin();
            S.erase(S.begin());
            int l = k - 1, r = k + 1;
            while(S.count(l)) S.erase(l--);
            while(S.count(r)) S.erase(r++);
            ans = max(ans, r-l-1);
        }
        return ans;
    }
};
```

10

1. 这题每次做都费九牛二虎之力：

```
class Solution {
    typedef vector<bool> vb;
public:
    bool isMatch(string s, string p) {
        string tmp;
        for(char c: p) {
            if(!tmp.empty() && tmp.back()=='*' && c=='*') cont
inue;
            tmp.push_back(c);
        }
        swap(p, tmp);
        int n = s.size(), m = p.size();
        vector<vb> dp(n+1, vb(m+1, false));
        dp[n][m] = true;
        for(int j=m-2; j>=0; --j) dp[n][j] = p[j]!='*' && p[j+
1] =='*' && dp[n][j+2];
        for(int i=n-1; i>=0; --i) for(int j=m-1; j>=0; --j){
            if(p[j] == '*') dp[i][j] = false;
            else if(j<m-1 && p[j+1]=='*'){
                dp[i][j] = dp[i][j+2] || ((p[j]==s[i] || p[j]=
='.') && dp[i+1][j]);
```

```
            }
            else dp[i][j] = dp[i+1][j+1] && (p[j]==s[i] || p
[j]=='.');
        }
        return dp[0][0];
    }
};
```

362

1. 这题明显比359还简单：

```
class HitCounter {
    queue<int> Q;
public:
    HitCounter() {}
    void hit(int timestamp) {
        Q.push(timestamp);
        while(Q.front() <= timestamp-300) Q.pop();
    }
    int getHits(int timestamp) {
        while(!Q.empty() && Q.front() <= timestamp-300) Q.pop
();
        return Q.size();
    }
};
```

708

1. 一直看错题，以为是return 那个插入的节点：

```
class Solution {
public:
    Node* insert(Node* head, int v) {
        if(!head){
            head = new Node(v, NULL);
            head->next = head;
            return head;
        }
        auto p0 = head, p1 = head->next, p2=head;
        do{
            if(p0->val <= v && p1->val >= v){
                Node *p = new Node(v, p1);
                p0->next = p;
                return head;
            }
            if(p1->val > p2->val) p2 = p1;
            p0 = p1;
            p1 = p1->next;
        }while(p0!=head);
        Node *p = new Node(v, p2->next);
        p2->next = p;
        return head;
    }
};
```

26

1.

```
public int removeDuplicates(int[] nums) {
        int k = 1;
        for (int i = 1; i < nums.length; i++) {
            while (i < nums.length && nums[i] == nums[i - 1])
{
                i++;
            }
            if (i == nums.length) {
                break;
            }
            nums[k] = nums[i];
            k++;
        }
        return k;
    }

```

2. Semi Short is Beauty 系列：

```
class Solution {
public:
    int removeDuplicates(vector<int>& nums) {
        if(nums.empty()) return 0;
        int l = 1;
        for(int j=1; j<nums.size(); ++j) if(nums[j] != nums[l-
1]) nums[l++] = nums[j];
        return l;
    }
};
```

241

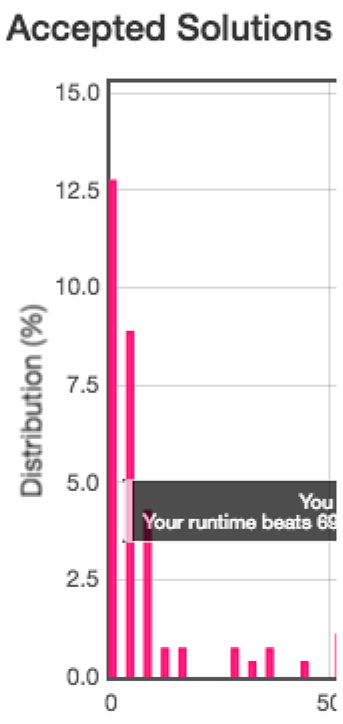
1. 无聊吧，重复的也算：

```
class Solution {
    typedef vector<int> vi;
    int getVal(int x, int y, char o){
        if(o=='+') return x+y;
        else if(o=='-') return x-y;
        else return x*y;
    }
public:
    vector<int> diffWaysToCompute(string s) {
        auto i = s.find_first_of("+-*");
        if(i==string::npos) return {stoi(s)};
        vi res;
        while(i!=string::npos){
            vi pre = diffWaysToCompute(s.substr(0, i)), pos = 
diffWaysToCompute(s.substr(i+1));
            for(auto x: pre) for(auto y: pos) res.push_back(ge
tVal(x, y, s[i]));
            i = s.find_first_of("+-*", i+1);
        }
        return res;
    }
};
```

291

1. 经典dfs：做剪枝可能会更快些

```
class Solution {
    typedef unordered_map<char, string> ucs;
    typedef unordered_set<string> us;
    bool dfs(int i, int j, const string&s, const string&p, ucs
&mmp, us&sst){
        if(i==s.size() || j>=p.size()) return j==p.size() && i
==s.size();
        if(mmp.count(s[i])){
            int l = mmp[s[i]].size();
            if(p.substr(j, l) != mmp[s[i]]) return false;
            else return dfs(i+1, j+l, s, p, mmp, sst);
        }
        for(int l=1; l<=p.size()-j; ++l){
            string tmp = p.substr(j, l);
            if(!sst.count(tmp)){
                mmp[s[i]] = tmp;
                sst.insert(tmp);
                if(dfs(i+1, j+l, s, p, mmp, sst)) return true;
                sst.erase(tmp);
            }
        }
        if(mmp.count(s[i])) mmp.erase(s[i]);
        return false;
    }
public:
    bool wordPatternMatch(string pattern, string str) {
        ucs mmp;
        us sst;
        return dfs(0, 0, pattern, str, mmp, sst);
    }
};
```



这些人是如何做到的

147

1. No way to get better than `O(`n^2`)`, right?

```
class Solution {
    void inse(ListNode *p, ListNode *q){
        while(p->next && p->next->val < q->val) p = p->next;
        q->next = p->next;
        p->next = q;
    }
public:
    ListNode* insertionSortList(ListNode* head) {
        if(!head || !head->next) return head;
        ListNode lead(0), *p=head->next;
        lead.next = head;
        for(head->next=NULL; p; ){
            ListNode *q = p->next;
            inse(&lead, p);
            p = q;
        }
        return lead.next;
    }
};
```

540

1. Short is Beauty 系列：

```
class Solution(object):
    def singleNonDuplicate(self, nums):
        return reduce(lambda x, y: x^y, nums)
```

1

1. `O(n)` : `O(n)` without sort:

```
class Solution {
```

```
public:
    vector<int> twoSum(vector<int>& nums, int target) {
        unordered_map<int, int> A;
        for(int i=0; i<nums.size(); ++i){
            if(A.count(target-nums[i])) return {A[target-nums
[i]], i};
            A[nums[i]] = i;
        }
    }
};
```

192

该死的BASH！！！！

468

1. 想一遍做对是不可能的：

```
class Solution(object):
    def validIPAddress(self, IP):
        if ('.' in IP and ':' in IP) or ('.' not in IP and ':'
not in IP): return "Neither"
        if '.' in IP:
            tokens = IP.split('.')
            if len(tokens) != 4: return 'Neither'
            for token in tokens:
                if (not token.isdigit()) or (len(token)!=1 and
token[0]=='0') or (int(token) > 255):
                    return 'Neither'
            return 'IPv4'
        tokens = IP.split(':')
        if len(tokens) != 8: return 'Neither'
        for token in tokens:
            if len(token) > 4 or (not token):
                return 'Neither'
            for c in token.lower():
                if c not in "0123456789abcdef":
                    return 'Neither'
        return 'IPv6'
```

423

1. 逐次排除法：

```
class Solution {
    const string cref = "wuxzorsvti";
    vector<int> nref = {2, 4, 6, 0, 1, 3, 7, 5, 8, 9};
    vector<string> sref = {"two", "four", "six", "zero", "on
e", "three", "seven", "five", "eight", "nine"};
public:
    string originalDigits(string s) {
        unordered_map<char, int> cnt;
        for(char c: s) ++cnt[c];
        string ans;
        for(int i=0; i<10; ++i){
            int k = cnt[cref[i]];
            if(k){
                for(int j=0; j<k; ++j) ans.push_back(char(nref
[i] + '0'));
                for(char c: sref[i]) cnt[c] -= k;
            }
        }
        sort(ans.begin(), ans.end());
        return ans;
    }
};
```

229

1. 利用C++库函数： `nth_element`, check every possible candidate

```
class Solution {
public:
    vector<int> majorityElement(vector<int>& nums) {
        if(nums.size() <= 1) return nums;
        int n = nums.size(), k = nums.size()/3;
        vector<int> res;
        stack<int> candidates;
        candidates.push(k);
        while(candidates.top() + k+1 < n) candidates.push(candidates.top() + k + 1);
        while(!candidates.empty()){
            int j = candidates.top();
            candidates.pop();
            nth_element(nums.begin(), nums.begin() + j, nums.end());
            if(res.empty() || nums[j] != res.back()){
                int cnt = 0;
                for(int m: nums) if(m==nums[j]) ++cnt;
                if(cnt > k) res.push_back(nums[j]);
            }
        }
        return res;
    }
};
```