August 11, 2018 题目：482,792,506,158,391,659,759,471,538,76,559,462,319

482

1. Short is beauty:

```
class Solution:
    def licenseKeyFormatting(self, S, K):
        S = S.replace("-", "").upper()[::-1]
        return '-'.join([S[i*K: i*K+K] for i in range(int((len(S)+K-1)/K))])[::-1]
```

792

1. Semi Brute Force: 复杂度 `O(len(S) + log(len(S))*len(words)*len(words[i]))`

```
class Solution {
public:
    int numMatchingSubseq(string S, vector<string>& words) {
        unordered_map<char, set<int>> pos;
        for(int i=0; i<S.size(); ++i) pos[S[i]].insert(i);
        int ans = 0;
        for(string s: words){
            int ok = 1;
            for(int i=0, idx=-1; i<s.size()&&ok; ++i){
                auto it = pos[s[i]].upper_bound(idx);
                if(it == pos[s[i]].end()) ok = 0;
                else idx = *it;
            }
            ans += ok;
        }
        return ans;
    }
};
```

2. 这个解法用 c++ 会超空间 ：https://leetcode.com/problems/number-of-matching-subsequences/discuss/146922/Java-O(S.length-+-word.length)-time-and-O(S.length)-space
3. 非常brute force了

```
class Solution {
    public int numMatchingSubseq(String S, String[] words) {
        int res = 0;
        Map<Character, List<Integer>> map = new HashMap<>();
        for (int i = 0; i < S.length(); i++) {
            char c = S.charAt(i);
            if (!map.containsKey(c)) map.put(c, new ArrayList<
>());
            map.get(c).add(i);
        }
        for (String w : words) {
            int start = -1;
            int i = 0;
            for (; i < w.length(); i++) {
                if (!map.containsKey(w.charAt(i))) break;
                List<Integer> index = map.get(w.charAt(i));
                int cur = start;
                for (int j : index) {
                    if (j > cur) {
                        cur = j;
                        break;
                    }
                }
                if (cur <= start) {
                    break;
                } else {
                    start = cur;
                }
            }
            if (i == w.length()) {
                res++;
            }
        }
        return res;
    }
}
```

506

1. Short is beauty:

```
class Solution:
    def findRelativeRanks(self, nums):
        medal = ["Gold Medal", "Silver Medal", "Bronze Medal"]
        ranks = dict([(s, i) for i, s in enumerate(sorted(num
s, key=lambda x: -x))])
        return [medal[ranks[s]] if ranks[s] < 3 else str(ranks
[s]+1) for s in nums]
```

2. 想short但是short不起来

```
class Solution {
    public String[] findRelativeRanks(int[] nums) {
        String[] res = new String[nums.length];
        Map<Integer, Integer> map = new HashMap<>();
        for (int i = 0; i < nums.length; i++) {
            map.put(nums[i], i);
        }
        PriorityQueue<Map.Entry<Integer, Integer>> pq = new PriorityQueue<>((a, b) -> b.getKey() - a.getKey());
        pq.addAll(map.entrySet());
        int i = 0;
        while (!pq.isEmpty()) {
            Map.Entry<Integer, Integer> cur = pq.poll();
            if (i == 0) {
                res[cur.getValue()] = "Gold Medal";
            } else if (i == 1) {
                res[cur.getValue()] = "Silver Medal";
            } else if (i == 2) {
                res[cur.getValue()] = "Bronze Medal";
            } else {
                res[cur.getValue()] = String.valueOf(i + 1);
            }
            i++;
        }
        return res;
    }
}
```

158

1. Use stack:

```
int read4(char *buf);
class Solution {
    stack<char> S;
public:
    int read(char *buf, int n) {
        int i = 0, dif=4;
        while(i<n && !S.empty()){
            buf[i++] = S.top();
            S.pop();
        }
        while(i<n && dif==4) i += (dif=read4(buf+i));
        while(i>n) S.push(buf[--i]);
        return i;
    }
};
```

391

1.

```
class Solution {
    public boolean isRectangleCover(int[][] rectangles) {
        int minX = Integer.MAX_VALUE, minY = Integer.MAX_VALU
E, maxX = Integer.MIN_VALUE, maxY = Integer.MIN_VALUE;
        Set<String> set = new HashSet<>();
        int area = 0;
        for (int[] rect : rectangles) {
            minX = Math.min(minX, rect[0]);
            minY = Math.min(minY, rect[1]);
            maxX = Math.max(maxX, rect[2]);
            maxY = Math.max(maxY, rect[3]);
            area += (rect[3] - rect[1]) * (rect[2] - rect[0]);
            String a = rect[0] + "," + rect[1];
            String b = rect[0] + "," + rect[3];
            String c = rect[2] + "," + rect[1];
            String d = rect[2] + "," + rect[3];
            if (!set.add(a)) set.remove(a);
            if (!set.add(b)) set.remove(b);
            if (!set.add(c)) set.remove(c);
            if (!set.add(d)) set.remove(d);
        }
        if (!set.contains(minX + "," + minY) || !set.contains
(minX + "," + maxY) || !set.contains(maxX + "," + minY) || !se
t.contains(maxX + "," + maxY) || set.size() != 4) return fals
e;
        area -= (maxY - minY) * (maxX - minX);

        return area == 0;
    }
}
```

2. 很有意思的一道题：Index Tree

```
class Solution {
    typedef pair<int, int> ii;
    map<ii, int> cnt;
public:
    bool isRectangleCover(vector<vector<int>>& rectangles) {
        for(auto vec: rectangles){
            cnt[ii(vec[0], vec[1])] += 1;
            cnt[ii(vec[0], vec[3])] -= 1;
            cnt[ii(vec[2], vec[1])] -= 1;
            cnt[ii(vec[2], vec[3])] += 1;
        }
        for(auto it=cnt.begin(); it!=cnt.end(); ){
            if(it->second) ++it;
            else it = cnt.erase(it);
        }
        if(cnt.size() != 4) return false;
        int x1 = cnt.begin()->first.first, y1 = cnt.begin()->f
irst.second;
        int x2 = (--cnt.end())->first.first, y2 = (--cnt.end
())->first.second;
        if(x2 == x1 || y2 == y1) return false;
        return cnt[ii(x1, y1)]==1 && cnt[ii(x1, y2)]==-1 && cn
t[ii(x2, y1)]==-1 && cnt[ii(x2, y2)]==1;
    }
};
```

659

1. Multiset solution，很慢：`O(n log n)`

```
class Solution {
    typedef pair<int, int> ii;
public:
    bool isPossible(vector<int>& nums) {
        multiset<ii> S;
        for(int k: nums){
            auto it = S.lower_bound(ii(k-1, 0));
            int l = 1;
            if(it!=S.end() && it->first==k-1){
                l += it->second;
                S.erase(it);
            }
            S.insert(ii(k, l));
        }
        for(auto p: S) if(p.second<3) return false;
        return true;
    }
};
```

2. 思路跟上面方法类似，但是用更聪明的方法维护每个成型的序列, Complexity: `O(N)`:

```
class Solution {
    typedef pair<int, int> ii;
public:
    bool isPossible(vector<int>& nums) {
        int n = nums.size();
        vector<int> len;
        int i = 0, j = 1, cur=nums[0]-2;
        while(i<n){
            if(nums[i] > cur+1){
                for(int k: len) if(k<3) return false;
                len.clear();
            }
            vector<int> tmp;
            for(j=1 ; i+j<n && nums[i+j]==nums[i]; ++j);
            for(int k=0; k<max(0, (int)len.size() - j); ++k) if(len[k]<3) return false;
            for(int k=max(0, (int)len.size() - j); k<len.size(); ++k) tmp.push_back(len[k] + 1);
            while(tmp.size()<j) tmp.push_back(1);
            swap(len, tmp);
            cur = nums[i];
            i += j;
        }
        for(int k: len) if(k<3) return false;
        return true;
    }
};
```

759

1. Using heap `O(Nlog(K))`, but kind of slow:

```
class Solution {
    typedef pair<Interval, int> Ii;
    struct G{
        bool operator () (const Ii&a, const Ii& b) const{
            if(a.first.start==b.first.start) return a.first.end > b.first.end;
            return a.first.start > b.first.start;
        }
    };
public:
    vector<Interval> employeeFreeTime(vector<vector<Interval>>& S) {
        vector<Interval> ans;
        priority_queue<Ii, vector<Ii>, G> Q;
        int n = S.size(), end=INT_MIN;
        vector<int> index(n, 0);
        for(int i=0; i<n; ++i) if(!S[i].empty()) Q.push(Ii(S[i][index[i]++], i));
        if(S.empty()) return ans;
        while(!Q.empty()){
            Interval x = Q.top().first;
            int j = Q.top().second;
            Q.pop();
            if(index[j] < S[j].size()) Q.push(Ii(S[j][index[j]++], j));
            end = max(end, x.end);
            if(!S.empty() && end<Q.top().first.start) ans.push_back(Interval(end, Q.top().first.start));
        }
        return ans;
    }
};
```

471

1. 超慢dfs 居然过了：

```
class Solution {
    typedef vector<string> vs;
    vector<vs> dp;
    string ref;
    string dfs(int i, int j){
        assert(i<=j);
        if(i > j-5) return ref.substr(i, j-i);
        if(dp[i][j] != "") return dp[i][j];
        int l = j-i, lmin=j-i;
        dp[i][j] = ref.substr(i, l);
        for(int q=1; q<=l; ++q){
            string s = ref.substr(i, q);
            int k = 0;
            while((k+1)*q<=l && ref.substr(i+k*q, q) == s) {
                ++k;
                string tmp = to_string(k) + "[" + dfs(i, i+q) + "]";
                if(tmp.size() >= k * q) tmp = ref.substr(i, k*q);
                tmp += dfs(i+k*q, j);
                if(tmp.size() < lmin){
                    dp[i][j] = tmp;
                    lmin = tmp.size();
                }
            }
        }
        return dp[i][j];
    }
public:
    string encode(string s) {
        int n = s.size();
        dp = vector<vs>(n+1, vs(n+1, ""));
        ref = s;
        return dfs(0, n);
    }
};
```
☐ Look at LeetCode Discussion @Zebo L

538

1. 反向 inorder：

```
class Solution {
public:
    TreeNode* convertBST(TreeNode* r) {
        stack<TreeNode *> S;
        int sum = 0;
        auto root = r;
        while(root || !S.empty()){
            while(root){
                S.push(root);
                root = root->right;
            }
            if(!S.empty()){
                sum += S.top()->val;
                S.top()->val = sum;
                root = S.top()->left;
                S.pop();
            }
        }
        return r;
    }
};
```

76

1. Sliding Window:

```
class Solution {
public:
    string minWindow(string s, string t) {
        map<char, int> cnt, ref;
        for(char c: t) ++ref[c];
        int i=0, j=0, L = s.size()+1, index = 0, n = 0;
        while(i<s.size() && j<s.size()){
            while(j<s.size() && n < t.size()){
                if(ref.count(s[j])) {
                    if(cnt[s[j]] < ref[s[j]]) ++n;
                    cnt[s[j]]++;
                }
                ++j;
            }
            if(n < ref.size()) break;
            while(i<j && n == t.size()){
                if(cnt.count(s[i])){
                    cnt[s[i]]--;
                    if(cnt[s[i]] < ref[s[i]]) --n;
                    if(!cnt[s[i]]) cnt.erase(s[i]);
                }
                ++i;
            }
            if(j - i + 1 < L){
                index = i-1;
                L = j - i + 1;
            }
        }
        if(L==s.size()+1) return "";
        return s.substr(index, L);
    }
};
```

559

1. DFS

```
class Solution {
    public int maxDepth(Node root) {
        if (root == null) return 0;
        return maxHelper(root) + 1;
    }
    public int maxHelper(Node root) {
        int res = 0;
        if (root == null) return res;

        for (Node node : root.children) {
            res = Math.max(res, maxHelper(node) + 1);
        }
        return res;
    }
}
```

2. Easy recursion:

```
class Solution {
public:
    int maxDepth(Node* root) {
        if(!root) return 0;
        int ans = 0;
        for(auto c: root->children) ans = max(ans, maxDepth
(c));
        return ans+1;
    }
};
```

462

1. ....

```
class Solution {
    public int minMoves2(int[] nums) {
        Arrays.sort(nums);
        int res = 0, i = 0, j = nums.length - 1;
        while (i < j) {
            res += nums[j] - nums[i];
            i++;
            j--;
        }
        return res;
    }
}
```

2. 之前用过二分法，后来知道一个结论，就是最小操作数之后的序列中的元素数是，原序列的median. 但还是楼上方法最nb

```
class Solution {
public:
    int minMoves2(vector<int>& nums) {
        sort(nums.begin(), nums.end());
        int n = nums.size(), ans = 0;
        for(int k: nums) ans += abs(nums[n/2] - k);
        return ans;
    }
};
```

319

1. 蛮有意思：

```
class Solution {
```

```
public:
    int bulbSwitch(int n) { return int(sqrt(n)); }
};
```