July 24, 2018 题目：246,326,269,726,56,372,717,589,478,815,624,47

246

1.

```
class Solution {
    public boolean isStrobogrammatic(String num) {
        Map<Character, Character> map = new HashMap<>();
        map.put('6', '9');
        map.put('9', '6');
        map.put('8', '8');
        map.put('0', '0');
        map.put('1', '1');

        StringBuilder sb = new StringBuilder();
        for (char c : num.toCharArray()) {
            sb.append(map.get(c));
        }

        return sb.reverse().toString().equals(num);
    }
}
```

2. 同上：

```
class Solution {
    map<char, char> ref;
public:
    bool isStrobogrammatic(string num) {
        ref['0'] = '0';
        ref['1'] = '1';
        ref['6'] = '9';
        ref['8'] = '8';
        ref['9'] = '6';
        int n = num.size();
        for(int i=0; i<(n+1)/2; ++i) if(!ref.count(num[i]) ||
ref[num[i]] != num[n-i-1]) return false;
        return true;
    }
};
```

326

1.

```
class Solution {
    public boolean isPowerOfThree(int n) {
        if (n == 0) return false;
        if (n == 1) return true;
        if (n % 3 != 0) return false;
        return isPowerOfThree(n / 3);
    }
}
```

2. 用对数，`O(1)` complexity:

```
class Solution {
public:
    bool isPowerOfThree(int n) {
        if(!n) return false;
        return abs(roundl(log(n)/logl(3)) - logl(n)/logl(3)) <
1.E-12;
    }
};
```

269

1. 拓扑排序：

```
class Solution {
    vector<set<int>> pre;
public:
    string alienOrder(vector<string>& words) {
        pre.resize(26);
        set<int> rest;
        for(string s: words) for(char c: s) rest.insert(int(c-
'a'));
        for(int i=1; i<words.size(); ++i){
            int j = 0;
            while(j<words[i].size() && j<words[i-1].size() &&
words[i][j]==words[i-1][j]) ++j;
            if(j==words[i].size() && j<words[i-1].size()) retu
rn "";
            if(j<words[i].size() && j<words[i-1].size()) pre[i
nt(words[i][j]-'a')].insert(int(words[i-1][j]-'a'));
        }
        string ans;
        while(!rest.empty()){
            set<int> candidates;
            for(int k: rest) if(pre[k].empty()) {
                candidates.insert(k);
            }
            if(candidates.empty()) return "";
            for(int k: candidates){
                rest.erase(k);
                ans += char(k + 'a');
            }
            for(int j: rest) for(int k:candidates) if(pre[j].c
ount(k)) pre[j].erase(k);
        }
        return ans;
    }
};
```

726

1. 太费劲了。。debug了好久才发现要首字母capital的才算一个元素- -

```
class Solution {
    public String countOfAtoms(String formula) {
        Map<String, Integer> map = new TreeMap<>();
        Stack<Map<String, Integer>> stack = new Stack<>();

        int index = 0;
        while (index < formula.length()) {
            String element = get(formula, index);
            index += element.length();

            if (element.equals("(")) {
                stack.push(map);
                map = new HashMap<>();
            } else if (element.equals(")")) {
                int val = 0;
                String next = get(formula, index);
                if (!Character.isDigit(next.charAt(0))) val =
1;
                else val = Integer.valueOf(next);
                if (!stack.isEmpty()) {
                    Map<String, Integer> tmp = map;
                    map = stack.pop();
                    for (String key : tmp.keySet()) {
                        map.put(key, map.getOrDefault(key, 0)
+ tmp.get(key) * val);
                    }
                }
            } else {
                if (!Character.isDigit(element.charAt(0))) {
                    int val = 0;
                    String next = get(formula, index);
                    if (!Character.isDigit(next.charAt(0))) va
l = 1;
                    else val = Integer.valueOf(next);
                    map.put(element, map.getOrDefault(element,
0) + val);
                }

            }
        }

        StringBuilder sb = new StringBuilder();

        for (String k : map.keySet()) {
            sb.append(k);

            if (map.get(k) != 1) {
                sb.append(map.get(k));
            }
        }
        return sb.toString();
    }
    private String get(String formula, int pos) {
        if (pos >= formula.length()) return " ";
        int i = pos;
        if (Character.isDigit(formula.charAt(i))) {
            while (i < formula.length() && Character.isDigit(f
ormula.charAt(i))) {
                i++;
            }
            return formula.substring(pos, i);
        }

        if (Character.isLetter(formula.charAt(i))) {
            i++;
            while (i < formula.length() && Character.isLowerCa
se(formula.charAt(i))) {
                i++;
            }
            return formula.substring(pos, i);
        }
        return formula.substring(pos, pos + 1);
    }
}
```

2. 同上，也是直接把map 放到 stack 里面:

```
class Solution {
    #define UP(c) ((c)>='A' && (c)<='Z')
    #define LO(c) ((c)>='a' && (c)<='z')
    #define NU(c) ((c)>='0' && (c)<='9')
public:
    string countOfAtoms(string formula) {
        map<string, int> ans;
        int i = 0;
        string atom;
        stack<map<string, int>> S;
        while(i<formula.size()){
            if(UP(formula[i])){
                string atom;
                atom += formula[i++];
                while(i<formula.size() && LO(formula[i])) atom
+= formula[i++];
                int cnt = 1;
                if(i<formula.size() && NU(formula[i])){
                    cnt = stoi(formula.substr(i));
                    i += to_string(cnt).size();
                }
                ans[atom] += cnt;
            }
            else if(formula[i] == '('){
                S.push(ans);
                ans.clear();
                ++i;
            }
            else{
                assert(formula[i]==')');
                ++i;
                int cnt = 1;
                if(i<formula.size() && NU(formula[i])){
                    cnt = stoi(formula.substr(i));
                    i += to_string(cnt).size();
                }
                auto tmp = S.top();
                S.pop();
                for(auto p: ans){
                    tmp[p.first] += p.second * cnt;
                }
                swap(ans, tmp);
            }
        }
        string res;
        for(auto p: ans) {
            res += p.first;
            if(p.second>1) res += to_string(p.second);
        }
        return res;
    }
};
```

56

1. 之前做过：注意细节就行

```
class Solution {
    static bool cmp(const Interval&i1, const Interval&i2){
        if(i1.start == i2.start) return i1.end<i2.end;
        return i1.start<i2.start;
    }
public:
    vector<Interval> merge(vector<Interval>& intervals) {
        sort(intervals.begin(), intervals.end(), cmp);
        vector<Interval> ans;
        int i = 0;
        while(i<intervals.size()){
            int start=intervals[i].start, end = intervals[i].e
nd;
```

```
            while(i<intervals.size() && intervals[i].start<=en
d){
                end = max(end, intervals[i].end);
                ++i;
            }
            ans.push_back(Interval(start, end));
        }
        return ans;
    }
};
```

372

1. LeetCode 就是喜欢出无聊的越界case

```
class Solution {
    #define M (1337)
    #define ADD(x, y) (((x) + (y))%(M))
    #define MUL(x, y) (((x) * (y))%(M))
    long pwr(long x, long k){
        int ans = 1;
        while(k){
            if(k%2) ans = MUL(ans, x);
            x = MUL(x, x);
            k /= 2;
        }
        return ans;
    }
public:
    int superPow(int aa, vector<int>& b) {
        long ans = 1, a = aa;
        for(int k: b){
            ans = MUL(pwr(ans, 10), pwr(a, long(k)));
        }
        return ans;
    }
};
```

717

1. 就是简单的dp，`O(1)` space 就行：

```
class Solution {
public:
    bool isOneBitCharacter(vector<int>& bits) {
        bool one=(bits[0]==0), two=false, pre=true;
        for(int i=1;i<bits.size();++i){
            bool cur = (one || two);
            one = (cur && bits[i]==0);
            two = (pre && bits[i-1]==1);
            pre = cur;
        }
        return one;
    }
};
```

589

1. 跟正常二叉树 `preorder` 完全一样

```
class Solution {
public:
    vector<int> preorder(Node* root) {
        stack<Node*> S;
        vector<int> ans;
        while(root){
            ans.push_back(root->val);
            for(int i=root->children.size()-1;i>=0;--i) S.push
(root->children[i]);
            if(S.empty()) break;
            root = S.top();
            S.pop();
        }
        return ans;
    }
};
```

478

Empty

815

1. 很直接的BFS，需要注意的是，只mark stop 会超时，同时要mark 已经乘坐过的公交：

```
class Solution {
    typedef pair<int, int> ii;
    unordered_map<int, vector<int>> S;
public:
    int numBusesToDestination(vector<vector<int>>& routes, int
start, int end) {
        vector<unordered_set<int>> R(routes.size());
        for(int i=0;i<routes.size();++i){
            for(int j=0; j<routes[i].size(); ++j){
                S[routes[i][j]].push_back(i);
            }
        }
        unordered_set<int> pass{start};
        unordered_set<int> Bs;
        queue<ii> Q;
        Q.push(ii(start, 0));
        while(!Q.empty()){
            int stop = Q.front().first, nbus = Q.front().secon
d;
            Q.pop();
            if(stop == end) return nbus;
            for(int bus: S[stop]) if(!Bs.count(bus)) {
                Bs.insert(bus);
                for(int k: routes[bus]) if(!pass.count(k)){
                    if(k == end) return nbus+1;
                    Q.push(ii(k, nbus+1));
                    pass.insert(k);
                }
            }
        }
        return -1;
    }
};
```

2.

```
class Solution {
    public int numBusesToDestination(int[][] routes, int S, in
t T) {
        if (S == T) return 0;
        Map<Integer, List<Integer>> map = new HashMap<>();
        for (int i = 0; i < routes.length; i++) {
            for (int j = 0; j < routes[i].length; j++) {
                List<Integer> list = map.getOrDefault(routes
[i][j], new ArrayList<>());
                list.add(i);
                map.put(routes[i][j], list);
            }
        }
        Queue<Integer> q = new LinkedList<>();
        q.offer(S);

        int bus = 0;
        Set<Integer> set = new HashSet<>();

        while (!q.isEmpty()) {
            int size = q.size();
            bus++;
            for (int i = 0; i < size; i++) {
                int cur = q.poll();
                List<Integer> list = map.get(cur);

                for (int k : list) {
                    if (set.contains(k)) continue;
                    set.add(k);
                    for (int j = 0; j < routes[k].length; j++)
{
                        if (routes[k][j] == T) return bus;
                        q.offer(routes[k][j]);
                    }
                }
            }

        }
        return -1;
    }
}
```

1. One pass:

```
class Solution {
public:
    int maxDistance(vector<vector<int>>& arrays) {
        int ans = 0, m=INT_MAX, M=INT_MIN;
        for(int k: arrays[0]) {
            m = min(m, k);
            M = max(M, k);
        }
        for(int i=1; i<arrays.size(); ++i){
            int tmp_m = INT_MAX, tmp_M = INT_MIN;
            for(int k: arrays[i]){
                tmp_m = min(tmp_m, k);
                tmp_M = max(tmp_M, k);
            }
            ans = max(ans, max(abs(tmp_m - M), abs(tmp_M -
m)));
            M = max(M, tmp_M);
            m = min(m, tmp_m);
        }
        return ans;
    }
};
```

47

1. Use Next Permutation，注意边界

```
class Solution {
    typedef vector<int> vi;
    bool nextPerm(vi &A, int n){
        int j = n-1;
        while(j && A[j]<=A[j-1]) --j;
        if(!j) return false;
        int k = j;
        while(k<n-1 && A[k+1]>A[j-1]) ++k;
        swap(A[j-1], A[k]);
        reverse(A.begin()+j, A.end());
        return true;
    }
public:
    vector<vector<int>> permuteUnique(vector<int>& nums) {
        sort(nums.begin(), nums.end());
        vector<vi> ans;
        do{
            ans.push_back(nums);
        }while(nextPerm(nums, nums.size()));
        return ans;
    }
};
```

- [ ] 递归？ @Zebo L