August 4, 2018 题目：854,44,50,521,544,596,708,43,455,797,175,632,497

854

1. BFS

```
class Solution {
    public int kSimilarity(String A, String B) {
        if (A.equals(B)) return 0;
        Queue<String> q = new LinkedList<>();
        Set<String> visited = new HashSet<>();
        q.offer(A);
        int res = 0;
        visited.add(A);
        while (!q.isEmpty()) {
            int cur = q.size();
            for (int k = 0; k < cur; k++) {
                String tmp = q.poll();
                if (tmp.equals(B)) return res;
                int i = 0;
                while (tmp.charAt(i) == B.charAt(i)) i++;
                for (int j = i + 1; j < tmp.length(); j++) {
                    if (tmp.charAt(j) == B.charAt(j) || tmp.ch
arAt(i) != B.charAt(j)) continue;
                    String swap = swap(i, j, tmp.toCharArray
());
                    if (!visited.contains(swap)) {
                        visited.add(swap);
                        q.offer(swap);
```

```
                    }
                }
            }
            res++;
        }
        return res;
    }

    public String swap(int i, int j, char[] s) {
        char tmp = s[i];
        s[i] = s[j];
        s[j] = tmp;
        return new String(s);
    }
}
```

2. ~~以下这个贪心，能过90% 的test case，可以不对：~~

```
class Solution {
    #define CI(c) int((c) - 'a')
    vector<vector<int>> M;
    int res;
    bool inArr(vector<int>& F, int j){
        for(int k: F) if(j==k) return false;
        return true;
    }
    void dfs(int l ,vector<int>& Q, vector<int>&F){
        if(!Q[0]) return;
        if(F.size() == l){
            int q = M[F.back()][F[0]];
            for(auto k: Q) q = min(q, k);
            for(auto &k: Q) k -= q;
```

```
            for(int i=0; i<l; ++i) M[F[i]][F[(i+1)%l]] -= q;
            res += (l-1) * q;
            return;
        }
        for(int k=0; k<6; ++k) if(inArr(F, k) && M[F.back()]
[k]){
            Q.push_back(M[F.back()][k]);
            F.push_back(k);
            dfs(l, Q, F);
            Q.pop_back();
            F.pop_back();
        }
    }
public:
    int kSimilarity(string A, string B) {
        M = vector<vector<int>>(6, vector<int>(6, 0));
        res = 0;
        for(int i=0; i<A.size(); ++i) M[CI(A[i])][CI(B[i])]++;
        for(int l=2; l<=6; ++l){
            for(int i=0; i<6; ++i) for(int j=0; j<6; ++j) if
(i!=j && M[i][j]){
                vector<int> Q{M[i][j]}, F{i, j};
                dfs(l, Q, F);
            }
        }
        return res;
    }
};
```

3. TLE BFS

```
class Solution {
    public int kSimilarity(String A, String B) {
```

```
        if (A.length() == 0) {
            return 0;
        }
        Queue<String> q = new LinkedList<>();
        int res = 0;
        q.offer(A);
        Set<String> visited = new HashSet<>();
        visited.add(A);
        while (!q.isEmpty()) {
            int size = q.size();
            for (int k = 0; k < size; k++) {
                String cur = q.remove();
                if (cur.equals(B)) {
                    return res;
                }
                int i = 0;
                while (i < cur.length()) {
                    while (i < cur.length() && cur.charAt(i) =
= B.charAt(i)) i++;
                    for (int j = i + 1; j < cur.length(); j++)
{
                        String next = swap(cur, i, j);
                        if (!visited.contains(next)) {
                            q.offer(next);
                            visited.add(next);
                        }
                    }
                    i++;
                }
            }
```

```
            res++;
        }
        return res;

    }
    String swap(String str, int l, int r) {
        char[] sc = str.toCharArray();
        StringBuffer sb = new StringBuffer();
        for (int i = 0; i < sc.length; i++) {
            if (i == l) {
                sb.append(sc[r]);
            } else if (i == r) {
                sb.append(sc[l]);
            } else {
                sb.append(sc[i]);
            }

        }
        return sb.toString();
    }
}
```

4. `808 ms` 的BFS：

```
class Solution {
    typedef pair<string, int> si;
public:
    int kSimilarity(string A, string B) {
        map<char, vector<int>> pos;
        int ans = 0;
        for(int i=0; i<B.size(); ++i) pos[B[i]].push_back(i);
```

```
        for(int i=0; i<A.size(); ++i) for(int j=i+1; j<A.size
(); ++j) if(A[i]!=A[j] && A[i]==B[j] && A[j]==B[i]){
            ans++;
            swap(A[i], A[j]);
        }
        unordered_set<string> S;
        queue<si> Q;
        S.insert(A);
        Q.push(si(A, ans));
        while(!Q.empty()){
            si cur = Q.front();
            Q.pop();
            string s = cur.first;
            int step = cur.second;
            if(s == B) return step;
            for(int i=0; i<A.size(); ++i) if(s[i]!=B[i]) for(i
nt j: pos[s[i]]) if(s[i]!=s[j]){
                swap(s[i], s[j]);
                if(!S.count(s)){
                    S.insert(s);
                    Q.push(si(s, step+1));
                }
                swap(s[i], s[j]);
            }
        }
        return -1;
    }
};
```

- [ ] 学习别人的方法 @Zebo L

1. `dp`

```
class Solution {
public:
    bool isMatch(string s, string p) {
        int n = s.size(), m = p.size();
        if(!n && !m) return true;
        if(!m) return false;
        vector<int> res(m+1, 0);
        for(int j=m-1; j>=0; --j) res[j] = int(p[j]!='*') + re
s[j+1];
        vector<vector<bool>> dp(n+1, vector<bool>(m+1, fals
e));
        dp[n][m] = true;
        for(int j=m-1; j>=0 && p[j]=='*'; --j) dp[n][j] = tru
e;
        for(int j=m-1; j>=0; --j) for(int i=min(n-1, n-res
[j]); i>=0; --i){
            if(p[j] == '*') {
                dp[i][j] = dp[i][j+1] || dp[i+1][j];
            }
            else dp[i][j] = (p[j]=='?' || s[i]==p[j]) && dp[i+
1][j+1];
        }
        return dp[0][0];
    }
};
```

50

1. 没啥好说的

```
class Solution {
    double pwr(double x, int n){
        double ans = 1.;
```

```
        while(n){
            if(n%2) ans *= x;
            x *= x;
            n /= 2;
        }
        return ans;
    }
public:
    double myPow(double x, int n) {
        return (n<0? (1./pwr(x, -n)): pwr(x, n));
    }
};
```

521

1. 毕竟Easy

```
class Solution {
public:
    int findLUSlength(string a, string b) {
        if(a == b) return -1;
        return max((int)a.size(), (int)b.size());
    }
};
```

544

1.

```
class Solution {
    public String findContestMatch(int n) {
        StringBuilder sb = new StringBuilder();

        List<String> list = new ArrayList<>();
        for (int k = 1; k <= n; k++) {
```

```
            list.add(String.valueOf(k));
        }

        while (list.size() != 1) {
            List<String> tmp = new ArrayList<>();
            int i = 0, j = list.size() - 1;
            while (i < j) {
                sb.append("(").append(list.get(i)).append
(",").append(list.get(j)).append(")");
                tmp.add(sb.toString());
                sb = new StringBuilder();
                i++;
                j--;
            }
            list = new ArrayList<>(tmp);
        }
        return list.get(0);
    }
}
```

2. Recursion:

```
class Solution {
public:
    string findContestMatch(int n) {
        if(n==2) return "(1,2)";
        string pre = findContestMatch(n/2), ans;
        int j = 0;
        while(j<pre.size()){
            if(isdigit(pre[j])){
                int k = stoi(pre.substr(j));
                ans += "(" + to_string(k) + "," + to_string(n-
k+1) + ")";
```

```
                j += to_string(k).size();
            }
            else ans+=pre[j++];
        }
        return ans;
    }
};
```

596

SQL

708

1. 记最大值点

```
class Solution {
public:
    Node* insert(Node* head, int insertVal) {
        if(!head) {
            auto p = new Node(insertVal, NULL);
            p->next = p;
            return p;
        }
        Node *p=head->next, *pm=head;
        int m = pm->val;
        for(; p->val!=m && !(p->val<=insertVal && p->next->val
>=insertVal); p=p->next){
            cout<<p->val << ' '<<p->next->val;
            if(p->val > m){
                m = p->val;
                pm = p;
            }
        }
```

```
        Node *q=new Node(insertVal, p->next);
        p->next = q;
        return head;
    }
};
```

43

1. 注意一下多个 Leading 0 的情况

```
class Solution {
    #define CI(c) int((c) - '0')
public:
    string multiply(string num1, string num2) {
        string ans;
        reverse(num1.begin(), num1.end());
        reverse(num2.begin(), num2.end());
        int n = num1.size(), m = num2.size(), cur = 0;
        for(int l=0; l<n+m-1 || cur; ++l){
            if(l<n+m-1){
                for(int i=max(0, l-m+1); i<=min(l, n-1); ++i){
                    cur += CI(num1[i]) * CI(num2[l-i]);
                }
            }
            ans += char(cur%10 + '0');
            cur /= 10;
        }
        while(ans.size()>1 && ans.back()=='0') ans.pop_back();
        reverse(ans.begin(), ans.end());
        return ans;
    }
};
```

455

1. 排序加数数

```
class Solution {
public:
    int findContentChildren(vector<int>& g, vector<int>& s) {
        sort(g.begin(), g.end(), greater<int>());
        sort(s.begin(), s.end(), greater<int>());
        int ans = 0;
        for(int i=0, j=0; i<s.size() && j<g.size(); ++i){
            while(j < g.size() && g[j] > s[i]) ++j;
            if(j<g.size()) {
                ++j;
                ++ans;
            }
        }
        return ans;
    }
};
```

797

1. DFS

```
class Solution {
    public List<List<Integer>> allPathsSourceTarget(int[][] gr
aph) {
        List<List<Integer>> res = new ArrayList<>();
        List<Integer> list = new ArrayList<>();
        list.add(0);
        helper(res, list, graph, 0);
        return res;
    }
```

```
    public void helper(List<List<Integer>> res, List<Integer>
list, int[][] graph, int k) {
        if (k == graph.length - 1) {
            res.add(new ArrayList<>(list));
            return;
        }
        int[] cur = graph[k];

        for (int c : cur) {
            list.add(c);
            helper(res, list, graph, c);
            list.remove(list.size() - 1);
        }

    }
}
```

2. DFS:

```
class Solution {
    void dfs(vector<vector<int>> &ans, vector<int> cur, vector
<vector<int>>& G){
        if(cur.back() == G.size()-1){
            ans.push_back(cur);
            return;
        }
        for(int j: G[cur.back()]){
            cur.push_back(j);
            dfs(ans, cur, G);
            cur.pop_back();
        }
    }
```

```
public:
    vector<vector<int>> allPathsSourceTarget(vector<vector<int
>>& graph) {
        vector<vector<int>> ans;
        dfs(ans, vector<int>{0}, graph);
        return ans;
    }
};
```

175

SQL

632

1. minHeap

```
class Solution {
    class Tuple {
        int val;
        int index;
        int rowNumber;
        public Tuple(int val, int index, int rowNumber) {
            this.val = val;
            this.index = index;
            this.rowNumber = rowNumber;
        }
    }
    public int[] smallestRange(List<List<Integer>> nums) {
        PriorityQueue<Tuple> pq = new PriorityQueue<>(1, new C
omparator<Tuple>(){
           public int compare(Tuple a, Tuple b) {
               return a.val - b.val;
           }
```

```
        });
        int max = Integer.MIN_VALUE;
        for (int i = 0; i < nums.size(); i++) {
            pq.offer(new Tuple(nums.get(i).get(0), 0, i));
            max = Math.max(max, nums.get(i).get(0));
        }
        int range = Integer.MAX_VALUE;
        int start = -1;
        int end = -1;
        while (pq.size() == nums.size()) {
            Tuple minT = pq.poll();
            if (max - minT.val < range) {
                range = max - minT.val;
                start = minT.val;
                end = max;
            }
            if (minT.index < nums.get(minT.rowNumber).size() -
1) {
                Tuple t = new Tuple(nums.get(minT.rowNumber).g
et(minT.index + 1), minT.index + 1, minT.rowNumber);
                pq.offer(t);
                if (t.val > max) max = t.val;
            }
        }
        return new int[]{start, end};
    }
}
```

2. 用Set :

```
class Solution {
    typedef pair<int, int> ii;
public:
```

```
    vector<int> smallestRange(vector<vector<int>>& nums) {
        vector<int> idx(nums.size(), 1);
        set<ii> Q;
        for(int i=0; i<nums.size(); ++i) Q.insert(ii(nums[i]
[0], i));
        vector<int> ans{Q.begin()->first, (--Q.end())->first};
        while(true){
            auto p = *Q.begin();
            Q.erase(Q.begin());
            if(idx[p.second] >= nums[p.second].size()) break;
            Q.insert(ii(nums[p.second][idx[p.second]++], p.sec
ond));
            if((--Q.end())->first - Q.begin()->first < ans[1]
- ans[0]){
                ans = {Q.begin()->first, (--Q.end())->first};
            }
        }
        return ans;
    }
};
```

497

1. 用面积建一个分布

```
class Solution {
    vector<vector<int>> R;
    vector<int> P;
    int N;
public:
    Solution(vector<vector<int>> rects): R(rects) {
        N = rects.size();
        P = vector<int>(N, 0);
```

```
        for(int i=0; i<N; ++i){
            P[i] = (rects[i][2]-rects[i][0] + 1) * (rects[i]
[3]-rects[i][1] + 1);
            if(i) P[i] += P[i-1];
        }
    }

    vector<int> pick() {
        int l = -1, r = N-1, u = rand()%P[N-1];
        while(l<r-1){
            int c = (l+r)/2;
            if(P[c]<=u) l = c;
            else r = c;
        }
        return vector<int>{R[r][0] + rand()%(R[r][2]-R[r][0]+
1), R[r][1] + rand()%(R[r][3]-R[r][1]+1)};
    }
};
```