August 3, 2018 题目: 389,852,31,336,425,738,239,758,599,597,360,164,283

389

1. 计出现次数

```
class Solution {
    #define CI(c) int((c) - 'a')
public:
    char findTheDifference(string s, string t) {
        vector<int> cnt(26, 0);
        for(char c: s) ++cnt[CI(c)];
        for(char c: t){
            --cnt[CI(c)];
            if(cnt[CI(c)]<0) return c;
        }
        return '0';
    }
};
```

852

1. 幼儿园题

```
class Solution {
public:
    int peakIndexInMountainArray(vector<int>& A) {
        for(int i=0; i<A.size()-1; ++i) if(A[i]>A[i+1]) return
i;
        return A.size()-1;
    }
```

```
};
```

31

1. 刷过题的人应该都见过吧

```
class Solution {
public:
    void nextPermutation(vector<int>& nums) {
        int j = nums.size()-1;
        while(j && nums[j]<=nums[j-1]) --j;
        if(j){
            int k = j;
            while(k<nums.size()-1 && nums[k+1]>nums[j-1]) ++k;
            swap(nums[j-1], nums[k]);
        }
        reverse(nums.begin()+j, nums.end());
    }
};
```

336(做过)

1. 单纯用map，用了两个hash map，为了写起来容易些。

```
class Solution {
public:
    vector<vector<int>> palindromePairs(vector<string>& words)
{
        map<string, int> pos, rev;
        vector<vector<int>> ans;
        for(int i=0; i<words.size(); ++i){
            pos[words[i]] = i;
            reverse(words[i].begin(), words[i].end());
            rev[words[i]] = i;
        }
```

```
        for(auto p: pos){
            string s = p.first;
            for(auto it=rev.lower_bound(s); it!=rev.end()&&it->first.size()>=s.size()&&it->first.substr(0, s.size())==s; ++it){
                if(it->second == p.second) continue;
                string t = it->first;
                bool ok = true;
                for(int i=0; i<(t.size()-s.size())/2&&ok; ++i) if(t[i+s.size()]!=t[t.size()-i-1]) ok=false;
                if(ok) ans.push_back(vector<int>{p.second, it->second});
            }
        }
        for(auto p: rev){
            string s = p.first;
            for(auto it=pos.upper_bound(s); it!=pos.end()&&it->first.size()>s.size()&&it->first.substr(0, s.size())==s; ++it){
                string t = it->first;
                bool ok = true;
                for(int i=0; i<(t.size()-s.size())/2&&ok; ++i) if(t[i+s.size()]!=t[t.size()-i-1]) ok=false;
                if(ok) ans.push_back(vector<int>{it->second, p.second});
            }
        }
        return ans;
    }
};
```

425（做过）

1. 一下是别人Tier solution。
    - [ ] @Zebo L 想想这个

```
class Solution {
    class TrieNode {
        char val;
        TrieNode[] child;
        List<String> prefix;
        TrieNode(char val) {
            this.val = val;
            this.child = new TrieNode[26];
            this.prefix = new ArrayList<>();
        }
    }
    public List<List<String>> wordSquares(String[] words) {
        TrieNode root = new TrieNode(' ');
        for (String word : words) {
            TrieNode curr = root;
            for (int i = 0; i < word.length(); i ++) {
                if (curr.child[word.charAt(i) - 'a'] == null)
{
                    curr.child[word.charAt(i) - 'a'] = new Tri
eNode(word.charAt(i));
                }
                curr = curr.child[word.charAt(i) - 'a'];
                curr.prefix.add(word);
            }
        }

        List<List<String>> result = new ArrayList<>();
        for (String word : words) {
            List<String> path = new ArrayList<>();
```

```
            path.add(word);
            traverse(path, result, root);
        }
        return result;
    }

    private void traverse(List<String> path, List<List<String>
> result, TrieNode root) {
        int i = path.size();
        if (i == path.get(0).length()) {
            result.add(new ArrayList<>(path));
            return;
        }
        TrieNode curr = root;
        int k = 0;
        while (k < i) {
            if (curr.child[path.get(k).charAt(i) - 'a'] != nul
l) {
                curr = curr.child[path.get(k).charAt(i) -
'a'];
            } else {
                return;
            }
            k ++;
        }
        for (String next : curr.prefix) {
            path.add(next);
            traverse(path, result, root);
            path.remove(path.size() - 1);
        }
    }
```

```
}
```

738

1. 不难，但要注意读题（monotonically increasing 居然指不减），注意细节

```
class Solution {
public:
    int monotoneIncreasingDigits(int N) {
        vector<int> D;
        while(N){
            D.push_back(N%10);
            N /= 10;
        }
        reverse(D.begin(), D.end());
        int i = 0, n = D.size(), ans = 0;
        while(i<n-1 && D[i]<=D[i+1]) ++i;
        if(i<n-1){
            while(i && D[i]==D[i-1]) --i;
            --D[i];
        }
        for(int j=0; j<=i; ++j) ans = ans*10 + D[j];
        for(int j=i+1; j<n; ++j) ans = ans*10 + 9;
        return ans;
    }
};
```

239

1. 用 `deque`

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

```
        vector<int> ans;
        deque<int> Q;
        for(int i=0; i<nums.size(); ++i){
            while(!Q.empty() && nums[i]>=nums[Q.back()]) Q.pop
_back();
            Q.push_back(i);
            if(Q.front()<=i-k) Q.pop_front();
            if(i>=k-1) ans.push_back(nums[Q.front()]);
        }
        return ans;
    }
};
```

758

1. Merging intervals approach:

```
class Solution {
public:
    string boldWords(vector<string>& words, string S) {
        int n = S.size();
        map<int, int> B;
        B[-2] = -1;
        B[n+5] = n+6;
        for(string s: words){
            auto i = S.find(s);
            while(i!=string::npos){
                auto it = --B.upper_bound(int(i));
                if(it->second < int(i)) ++it;
                int start = min((int)it->first, (int)i), end =
i + s.size();
                while(it->first <= end){
                    end = max(end, it->second);
```

```
                    it = B.erase(it);
                }
                B[start] = end;
                i = S.find(s, i+1);
            }
        }
        int start = 0;
        string ans;
        for(auto it=++B.begin(); it!=--B.end(); ++it){
            ans += S.substr(start, it->first-start) + "<b>" +
S.substr(it->first, it->second-it->first) + "</b>";
            start = it->second;
        }
        ans += S.substr(start, n-start);
        return ans;
    }
};
```

599

1. One pass + hash

```
class Solution {
public:
    vector<string> findRestaurant(vector<string>& list1, vecto
r<string>& list2) {
        unordered_map<string, int> pos;
        vector<string> ans;
        int cur = 1E8;
        for(int i=0; i<list2.size(); ++i) pos[list2[i]] = i;
        for(int j=0; j<list1.size(); ++j) if(pos.count(list1
[j]) && j+pos[list1[j]]<=cur){
            if(j+pos[list1[j]]<cur) {
```

```
                ans.clear();
                cur = j+pos[list1[j]];
            }
            ans.push_back(list1[j]);
        }
        return ans;
    }
};
```

597

SQL

360

1. 题很简单，就是要注意一下 `a<0` 和 `a==0 && b<0 || b>0` 的情况。

```
class Solution {
public:
    vector<int> sortTransformedArray(vector<int>& nums, int a,
int b, int c) {
        if(!a){
            bool rev = (b < 0);
            for(auto &n: nums) n = b*n + c;
            if(rev) reverse(nums.begin(), nums.end());
            return nums;
        }
        bool rev = (a < 0);
        int l=-1, r=nums.size(), n=nums.size();
        double center = -double(b)/2./a;
        while(l < r-1){
            int c = (l+r)/2;
            if(double(nums[c]) < center) l = c;
            else r = c;
```

```
        }
        vector<int> ans;
        for(int i=0; i<n; ++i){
            if(l<0) {
                ans.push_back(a * nums[r] * nums[r] + b * nums
[r] + c);
                ++r;
            }
            else if(r>=n) {
                ans.push_back(a * nums[l] * nums[l] + b * nums
[l] + c);
                --l;
            }
            else if(nums[r]-center < center-nums[l]){
                ans.push_back(a * nums[r] * nums[r] + b * nums
[r] + c);
                ++r;
            }
            else{
                ans.push_back(a * nums[l] * nums[l] + b * nums
[l] + c);
                --l;
            }
        }
        if(rev) reverse(ans.begin(), ans.end());
        return ans;
    }
};
```

164

1. 典型的Bucked sort （边界条件很烦）

```
class Solution {
    typedef pair<int, int> ii;
public:
    int maximumGap(vector<int>& nums) {
        if(nums.size() <= 1) return 0;
        if(nums.size() == 2) return abs(nums[0] - nums[1]);
        long m = INT_MAX, M = INT_MIN, n = nums.size(), ans =
0L;
        for(int k: nums) {
            m = min(long(k), m);
            M = max(long(k), M);
        }
        if(M==m) return 0;
        long block_size = (M - m + n - 2)/(n-1);
        long num_blocks = (M - m + block_size - 1) / block_siz
e;
        cout << block_size <<' '<<num_blocks<<endl;
        vector<ii> bucket(num_blocks + 1, ii(-1, -1));
        for(int k: nums){
            int idx = (long(k) - m)/block_size;
            if(bucket[idx].first==-1){
                bucket[idx] = ii(k, k);
            }
            else{
                bucket[idx].first = min(bucket[idx].first, k);
                bucket[idx].second = max(bucket[idx].second,
k);
            }
        }
        for(auto p: bucket) if(p.first != -1){
            ans = max(ans, p.first - m);
```

```
            m = p.second;
        }
        return ans;
    }
};
```

- [ ] 以后重温上述方法 @Zebo L

283

1. One pass:

```
class Solution {
public:
    void moveZeroes(vector<int>& nums) {
        int k = 0;
        for(int i=0; i<nums.size(); ++i){
            if(nums[i]) nums[k++] = nums[i];
        }
        while(k < nums.size()) nums[k++] = 0;
    }
};
```