July 17, 2018 题目 352,508,839,27,805,170,396,789,91,236

352

1. MAP 实现:

```
class SummaryRanges {
    map<int, int> R;
public:
    /** Initialize your data structure here. */
    SummaryRanges() {
        R[-10] = -10;
    }

    void addNum(int val) {
        if(R.count(val)) return;
        auto it = --R.lower_bound(val);
        if(val <= it->second) return;
        int beg = val, end = val;
        if(it->second == val-1) beg = it->first;
        if(R.count(val+1)){
            end = R[val + 1];
            R.erase(val + 1);
        }
        R[beg] = end;
    }

    vector<Interval> getIntervals() {
        vector<Interval> ans;
```

```
        for(auto p: R) if(p.first != -10) ans.push_back(Interv
al(p.first, p.second));
        return ans;
    }
};
```

2. Same as above.

508

1. 简单 `dfs`

```
class Solution {
    unordered_map<int, int> cnt;
    int res = 0;
    int dfs(TreeNode* root){
        if(!root) return 0;
        int ans = root->val + dfs(root->left) + dfs(root->righ
t);
        ++cnt[ans];
        res = max(res, (cnt[ans]));
        return ans;
    }
public:
    vector<int> findFrequentTreeSum(TreeNode* root) {
        vector<int> ans;
        if(!root) return ans;
        dfs(root);
        for(auto p: cnt) if(p.second == res) ans.push_back(p.f
irst);
        return ans;
    }
};
```

2. DFS + hashmap

839

1. BFS O(N!*len) worst?

```
public int numSimilarGroups(String[] A) {
        int res = 0;
        boolean[] visited = new boolean[A.length];
        Queue<String> q = new LinkedList<>();
        for (int i = 0; i < A.length; i++) {
            if (visited[i]) continue;
            q.offer(A[i]);
            visited[i] = true;
            res++;

            while (!q.isEmpty()) {
                String cur = q.poll();
                for (int j = 0; j < A.length; j++) {
                    if (visited[j]) continue;

                    int diff = 0;
                    for (int k = 0; k < cur.length(); k++) {
                        if (cur.charAt(k) != A[j].charAt(k)) d
iff++;
                    }
                    if (diff == 2 || cur.equals(A[j])) {
                        q.offer(A[j]);
                        visited[j] = true;
                    }
                }
            }
        }
```

```
        return res;
        }
```

2. Union Find

```
class Solution {
    class UnionFind{
        int[] father;
        int count;
        UnionFind(int size) {
            father = new int[size];
            count = size;
            for (int i = 0; i < size; i++) {
                father[i] = i;
            }
        }
        int find(int x) {
            while(x != father[x]) {
                father[x] = father[father[x]];

                x = father[x];
            }
            return x;
        }

        void union(int x, int y) {
            int root_x = find(x);
            int root_y = find(y);
            if (root_x == root_y) {
                return;
            }
            father[root_x] = root_y;
```

```
            count--;
        }

    }

    public int numSimilarGroups(String[] A) {
        UnionFind uf = new UnionFind(A.length);
        for (int i = 0; i < A.length; i++) {
            for (int j = i + 1; j < A.length; j++) {
                if (similar(A[i], A[j])) {
                    uf.union(i, j);
                }
            }
        }
        return uf.count;
    }
    boolean similar(String s1, String s2) {
        int i = 0;
        int count = 0;
        while (i < s1.length()) {
            if (s1.charAt(i) != s2.charAt(i)) {
                count++;
                if (count > 2) {
                    return false;
                }
            }
            i++;
        }
        return true;
    }
```

```
}
```

3. 同上，中文名：并查集：

```
class Solution {
    vector<int> P;
    int getRoot(int i){
        if(P[i] == i) return i;
        return P[i] = getRoot(P[i]);
    }
public:
    int numSimilarGroups(vector<string>& A) {
        P.resize(A.size());
        for(int i=0; i<P.size(); ++i) P[i] = i;
        for(int i=1; i<A.size(); ++i) for(int j=0; j<i; ++j){
            bool ok = true;
            for(int k=0, cnt=0; k<A[j].size()&&ok; ++k) {
                if(A[j][k] != A[i][k]) ++cnt;
                if(cnt>2) ok=false;
            }
            if(ok) P[getRoot(i)] = getRoot(j);
        }
        int ans = 0;
        for(int i=0;i<P.size();++i) if(P[i] == i) ++ans;
        return ans;
    }
};
```

4. 超时BFS, 引以为戒，估算复杂度估算错了，选择了错误的方向:
    - Note: `unordered_set` 存储string 有额外`O(l)`的复杂度

```
class Solution {
public:
    int numSimilarGroups(vector<string>& A) {
```

```
        unordered_set<string> S(A.begin(), A.end());
        int ans = 0;
        while(!S.empty()){
            string s = *S.begin();
            queue<string> Q;
            S.erase(s);
            Q.push(s);
            while(!Q.empty()){
                s = Q.front();
                Q.pop();
                for(int i=0;i<s.size();++i) for(int j=1;j<s.si
ze();++j){
                    swap(s[i], s[j]);
                    if(S.count(s)){
                        Q.push(s);
                        S.erase(s);
                    }
                    swap(s[i], s[j]);
                }
            }
            ++ans;
        }
        return ans;
    }
};
```

5. 同 1 ， c++ 版本的BFS ， 过了 ：

```
class Solution {
public:
    int numSimilarGroups(vector<string>& A) {
        int ans = 0;
```

```
        unordered_set<string> rest(A.begin(), A.end());
        while(!rest.empty()){
            string s = *rest.begin();
            rest.erase(rest.begin());
            queue<string> Q;
            Q.push(s);
            while(!Q.empty()){
                s = Q.front();
                Q.pop();
                for(auto it=rest.begin(); it!=rest.end(); ){
                    bool ok = true;
                    string t = *it;
                    for(int i=0, cnt=0;i<s.size() && ok;++i){
                        if(s[i] != t[i]) ++cnt;
                        if(cnt > 2) ok = false;
                    }
                    if(ok){
                        it = rest.erase(it);
                        Q.push(t);
                    }
                    else ++it;
                }
            }
            ++ans;
        }
        return ans;
    }
};
```

1. 很直接的One Pass

```
class Solution {
public:
    int removeElement(vector<int>& nums, int val) {
        int i = 0, n = nums.size();
        while(i < n){
            if(nums[i] == val) swap(nums[i], nums[--n]);
            else ++i;
        }
        return n;
    }
};
```

2. 同向快慢指针，only swap when not equal to the target value

805

1. Preprocessing + Optimized 背包问题：
    a. Preprocessing: 所有元素都减去该数组的平均值，这样原题转化为数组 `A[:-1]`（拿掉最后一个元素）是否有若干个数之和为 `0`
    b. 背包算法要优化，否则会超时

```
class Solution {
    bool dfs(int i, int tar, bool cnt, vector<int> &A){
        if(!tar && cnt) return true;
        if(!i) return false;
        if(tar>0 && A[i-1]<0) return false;
        for(int j=0; j<i; ++j) if(dfs(j, tar-A[j], true, A)) r
eturn true;
        return false;
    }
public:
    bool splitArraySameAverage(vector<int>& A) {
        int n = A.size(), sum = 0;
        sort(A.begin(), A.end());
```

```
        if(n<2) return false;
        for(int &k: A) {
            sum += k;
            k *= n;
        }
        for(int &k: A) {
            k -= sum;
            if(!k) return true;
        }
        return dfs(n-1, 0, false, A);
    }
};
```

- [ ] See LeetCode discussion @Zebo L

170

```
class TwoSum {

    /** Initialize your data structure here. */
    Map<Integer, Integer> map;
    public TwoSum() {
        map = new HashMap<>();
    }

    /** Add the number to an internal data structure.. */
    public void add(int number) {
        if (map.containsKey(number)) {
            map.put(number, map.get(number) + 1);
        } else {
            map.put(number, 1);
        }
```

```
        return;
    }

    /** Find if there exists any pair of numbers which sum is
equal to the value. */
    public boolean find(int value) {
        for (int k : map.keySet()) {
            int remain = value - k;
            if (remain == k && map.get(k) > 1 || remain != k &
& map.containsKey(remain)) {
                return true;
            }
        }
        return false;
    }
}
```

2. Always choose one of both to optimize

3. Hash 即可

```
class TwoSum {
public:
    unordered_map<int, int> cnt;
    TwoSum() {}
    void add(int number) {
        cnt[number]++;
    }
    bool find(int value) {
        for(auto p: cnt) if(cnt.count(val - p.first)){
            if(p.first + p.first != val) return true;
```

```
            if(p.second > 1) return true;
        }
        return false;
    }
};
```

396

1, tricky math. F[k] - F[k - 1] = sum + size * (the element times 0)

2. Maintain 一个 sum 就行了

```
class Solution {
public:
    int maxRotateFunction(vector<int>& A) {
        int ans = 0, n=A.size(), sum = 0;
        for(int i=0; i<n; ++i) ans += i*A[i], sum += A[i];
        for(int i=n-1, tmp=ans; i>=0; --i){
            tmp += sum;
            tmp -= A[i] * n;
            ans = max(ans, tmp);
        }
        return ans;
    }
};
```

789

1. Totally math problem..... Calculate the manhattan distance for ghosts to target and you to target and compare. =-=

```
public boolean escapeGhosts(int[][] ghosts, int[] target) {
        int dest = Math.abs(target[0]) + Math.abs(target[1]);
        for (int[] ghost : ghosts) {
            int dis = Math.abs(target[1] - ghost[1]) + Math.ab
s(target[0] - ghost[0]);
```

```
            if (dis <= dest) return false;
        }
        return true;
    }
```

2. 同上

```
class Solution {
public:
    bool escapeGhosts(vector<vector<int>>& ghosts, vector<int>
& target) {
        int dis = abs(target[0]) + abs(target[1]);
        for(auto p: ghosts) if(abs(p[0] - target[0]) + abs(p
[1]-target[1]) <= dis) return false;
        return true;
    }
};
```

91

1. Use dp to track if two digits are <= 26, if yes, the decode ways for them would be 3 (if either of the digit is not 0). Adding up from tail to head.

```
public int numDecodings(String s) {
        int[] dp = new int[s.length() + 1];
        dp[s.length()] = 1;
        dp[s.length() - 1] = s.charAt(s.length() - 1) == '0' ?
0 : 1;
        for (int i = dp.length - 3; i >= 0; i--) {
            if (s.charAt(i) == '0') continue;
            dp[i] = Integer.parseInt(s.substring(i, i + 2)) <=
26 ? dp[i + 1] + dp[i + 2] : dp[i + 1];
        }
        return dp[0];
    }
```

2. 经典dp:

```
class Solution {
public:
    int numDecodings(string s) {
        if(s.empty()) return 0;
        vector<int> dp(s.size() + 1, 1);
        dp[s.size()-1] = int(s[s.size() - 1] != '0');
        for(int i=s.size()-2; i>=0; --i){
            if(s[i] == '0') dp[i] = 0;
            else dp[i] = dp[i+1] + int(int(s[i]-'0')*10 + int
(s[i+1]-'0') <= 26)*dp[i+2];
        }
        return dp[0];
    }
};
```

3. dp left to right

```
class Solution {
    public int numDecodings(String s) {
        if (s == null || s.length() == 0) {
            return 0;
        }
        int[] dp = new int[s.length() + 1];

        char[] sc = s.toCharArray();
        dp[0] = 1;
        dp[1] = sc[0] == '0' ? 0 : 1;

        for (int i = 2; i <= sc.length; i++) {
            if (sc[i - 1] != '0') {
                dp[i] = dp[i - 1];
            }
```

```
            int second = (sc[i - 2] - '0') * 10 + sc[i - 1] -
'0' ;
            if (second <= 26 && second >= 10) {
                dp[i] += dp[i - 2];
            }
        }
        return dp[sc.length];
    }
}
```

4. o(1) space dp

```
class Solution {
    public int numDecodings(String s) {
        if (s == null || s.length() == 0) {
            return 0;
        }
        int prev = 1;
        int cur = s.charAt(0) == '0' ? 0 : 1;
        for (int i = 1; i < s.length(); i++) {
            int temp = cur;
            if (s.charAt(i) == '0') {
                cur = 0;
            }
            int sec = (s.charAt(i - 1) - '0') * 10 + s.charAt
(i) - '0';
            if (sec >= 10 && sec <= 26) {
                cur += prev;
            }
            prev = temp;
        }
        return cur;
```

```
    }
}
```

236

1.

```
public TreeNode lowestCommonAncestor(TreeNode root, TreeNode
p, TreeNode q) {
        if (root == null) return null;
        if (root == p || root == q) return root;
        TreeNode left = lowestCommonAncestor(root.left, p, q);
        TreeNode right = lowestCommonAncestor(root.right, p,
q);

        if (left != null && right != null) return root;
        return left != null ? left : right;
    }
```

2. 多做了几个hash，不过好像都是多余的。

```
class Solution {
    unordered_map<TreeNode *, TreeNode *> parent;
    unordered_map<TreeNode *, int> level;
    void dfs(TreeNode *root, int l){
        if(!root) return;
        level[root] = l;
        if(root->left) {
            parent[root->left] = root;
            dfs(root->left, l+1);
        }
        if(root->right){
            parent[root->right] = root;
            dfs(root->right, l+1);
```

```
        }
    }
public:
    TreeNode* lowestCommonAncestor(TreeNode* root, TreeNode*
p, TreeNode* q) {
        parent[root] = NULL;
        dfs(root, 0);
        if(level[p] > level[q]) swap(p, q);
        while(level[p] < level[q]) q = parent[q];
        while(p!=q){
            p = parent[p];
            q = parent[q];
        }
        return p;
    }
};
```