July 8, 2018

@Mingze X @Chong T @Zhaorui D @Yu S @Zebo L

486

1. dp:

```
class Solution {
    typedef vector<int> vi;
public:
    bool PredictTheWinner(vector<int>& nums) {
        int n = nums.size();
        vector<vi> dp(n, vi(n, 0)), sm(n, vi(n, 0));
        for(int i=0;i<n;++i) dp[i][i] = sm[i][i] = nums[i];
        for(int l=1;l<n;++l) for(int i=0;i<n-l;++i){
            int j = i+l;
            sm[i][j] = sm[i][j-1] + nums[j];
            dp[i][j] = max(nums[i]+sm[i+1][j]-dp[i+1][j], nums
[j]+sm[i][j-1]-dp[i][j-1]);
        }
        return dp[0][n-1] * 2 >= sm[0][n-1];
    }
};
```

2. dfs, use memo to optimize the time complexity.

```
public class Solution {
    public boolean PredictTheWinner(int[] nums) {
        return winner(nums, 0, nums.length - 1, 1) >= 0;
    }
    public int winner(int[] nums, int s, int e, int turn) {
        if (s == e)
            return turn * nums[s];
```

```
        int a = turn * nums[s] + winner(nums, s + 1, e, -tur
n);
        int b = turn * nums[e] + winner(nums, s, e - 1, -tur
n);
        if(turn == 1)
        return  Math.max(a, b);
        else {
            return Math.min(a,b);
        }
    }
}
```

721

1. 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:
    vector<vector<string>> accountsMerge(vector<vector<string>
>& accounts) {
        unordered_map<string, int> pos;
        int n = accounts.size();
        P.resize(n);
        for(int i=0;i<n;++i) {
            P[i] = i;
            for(int j=1; j<accounts[i].size(); ++j){
                if(pos.count(accounts[i][j])) P[getRoot(i)] =
getRoot(pos[accounts[i][j]]);
```

```
                else pos[accounts[i][j]] = getRoot(i);
            }
        }
        unordered_map<int, set<string>> col;
        for(int i=0;i<n;++i){
            int root = getRoot(i);
            for(int j=1; j<accounts[i].size(); ++j) col[root].
insert(accounts[i][j]);
        }
        vector<vector<string>> ans;
        for(auto p: col){
            ans.push_back(vector<string>{accounts[p.first]
[0]});
            for(auto s: p.second) ans[ans.size()-1].push_back
(s);
        }
        return ans;
    }
};
```

2. typical Union Find.

703:

- Empty

847

1. Memo dp:

```
class Solution {
    int dp[15][5000];
    int target, inf=1E8;
    int getShortestPath(int j, int state, vector<vector<int>>&
G){
        if(state == target) return 0;
```

```
        if(dp[j][state]>=0) return dp[j][state];
        int tmp = state | (1<<j);
        if(tmp == target) return dp[j][state]=1;
        dp[j][state] = inf;
        for(auto i: G[j]) dp[j][state] = min(dp[j][state], 1+getShortestPath(i, tmp, G));
        return dp[j][state];
    }
public:
    int shortestPathLength(vector<vector<int>>& graph) {
        memset(dp, -1, sizeof(dp));
        int ans = inf, n=graph.size();
        target = (1<<n)-1;
        for(int i=0;i<n;++i) ans = min(ans, getShortestPath(i, 0, graph));
        return ans-1;
    }
};
```

2. BFS, can use bits to represent the visited nodes as there are less than 15 nodes.

350

1. 用map计数：follow up 也挺无聊

```
class Solution {
public:
    vector<int> intersect(vector<int>& nums1, vector<int>& nums2) {
        map<int, int> m1, m2;
        vector<int> ans;
        for(int n:nums1) m1[n]++;
        for(int m:nums2) m2[m]++;
```

```
        for(auto p:m1) if(m2.count(p.first)) for(int i=0;i<min
(p.second, m2[p.first]);++i) ans.push_back(p.first);
        return ans;
    }
};
```

80

1. 注意inplace 操作的时候不能同 `nums[i] == nums[i-1]` 来判断重复:

```
class Solution {
public:
    int removeDuplicates(vector<int>& nums) {
        if(nums.empty()) return 0;
        int n = 1;
        for(int i=1,cnt=1,tmp=nums[0]; i<nums.size(); ++i){
            if(nums[i]==tmp) ++cnt;
            else{
                cnt = 1;
                tmp = nums[i];
            }
            if(cnt<=2) {
                nums[n++] = nums[i];
            }
        }
        return n;
    }
};
```

677

1. Tier:

```
class MapSum {
    struct Tier{
```

```
        int sum;
        map<char, Tier*> chl;
        Tier(): sum(0) {}
    };
    Tier *root;
    unordered_map<string, int> M;
public:
    /** Initialize your data structure here. */
    MapSum() {
        root = new Tier();
    }

    void insert(string key, int val) {
        int delta = val - M[key];
        auto p = root;
        for(char c: key){
            p->sum += delta;
            if(!p->chl.count(c)) p->chl[c] = new Tier();
            p = p->chl[c];
        }
        p->sum += delta;
        M[key] = val;
    }

    int sum(string prefix) {
        auto p = root;
        for(auto c: prefix){
            if(!p->chl.count(c)) return 0;
            p = p->chl[c];
        }
```

```
        return p->sum;
    }
};
```

259

1. 确定第一个数，另外两个两端夹一下：

```
class Solution {
public:
    int threeSumSmaller(vector<int>& nums, int target) {
        sort(nums.begin(), nums.end());
        int ans = 0, n = nums.size();
        for(int i=0;i<n;++i){
            int j = i+1, k = n-1;
            while(j<k){
                while(k>j && nums[k] + nums[j]>=target-nums
[i]) --k;
                ans += k-j;
                ++j;
            }
        }
        return ans;
    }
};
```

169

1. One pass vote:

```
class Solution {
public:
    int majorityElement(vector<int>& nums) {
        int tmp = nums[0], cnt=0;
        for(int n: nums){
```

```
            if(n==tmp) ++cnt;
            else{
                if(!cnt){
                    tmp = n;
                    cnt = 1;
                }
                else --cnt;
            }
        }
        return tmp;
    }
};
```

134

1. 简单的判定：

```
class Solution {
public:
    int canCompleteCircuit(vector<int>& gas, vector<int>& cos
t) {
        int idx = 0, low = gas[0]-cost[0], sum = gas[0]-cost
[0];
        for(int i=1; i<cost.size(); ++i){
            sum += gas[i] - cost[i];
            if(sum < low) {
                low = sum;
                idx = i;
            }
        }
        return (sum>=0? (idx+1)%(int)cost.size():-1);
    }
};
```