July 6, 2018

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

611

1. Sort the array. Loop through from right to left, and use 2 sum.

```
class Solution {
    public int triangleNumber(int[] nums) {
        if (nums == null || nums.length <= 2) {
            return 0;
        }
        Arrays.sort(nums);
        int res = 0;

        for (int i = nums.length - 1; i >= 0; i--) {
            int tar = nums[i];
            int l = 0;
            int r = i - 1;
            while (l < r) {
                if (nums[l] + nums[r] > tar) {
                    res += r - l;
                    r--;
                } else {
                    l++;
                }
            }
        }

        return res;
    }
}
```

2. GoodSpeed's solution:
    a. lang: c++
    b. idea: 同上：

```
class Solution {
public:
    int triangleNumber(vector<int>& nums) {
        int n = nums.size(), ans = 0;
        if(n < 3) return 0;
        sort(nums.begin(), nums.end());
        for(int i=0; i<n ;++i) for(int j=i+1, k=i+2; j<n; ++j)
{
            k = max(j+1, k);
            int tar = nums[i] + nums[j];
            while(k<n && nums[k]<tar) ++k;
            ans += k-j-1;
        }
        return ans;
    }
};
```

467

1. Actually hard. DP, count the longest string ending at letter, sum them up.

```
class Solution {
    public int findSubstringInWraproundString(String p) {
        if (p == null || p.length() == 0) {
            return 0;
        }

        int[] count = new int[26];
        char[] pc = p.toCharArray();

        int len = 1;
        for (int i = 0; i < pc.length; i++) {
            if (i > 0 && (pc[i] == pc[i - 1] + 1 || pc[i - 1]
== pc[i] + 25)) {
                len++;
            } else {
                len = 1;
            }
            count[pc[i] - 'a'] = Math.max(count[pc[i] - 'a'],
len);
        }

        int res = 0;
        for (int c : count) {
            res += c;
        }
        return res;
    }
}
```

2. GoodSpeed's solution:
    a. lang: c++
    b. idea: 同上，计数问题：

```
class Solution {
    int toNum(char c){
        return int(c - 'a');
    }
public:
    int findSubstringInWraproundString(string p) {
        if(p.empty()) return 0;
        int ans = 0;
        vector<int> cnt(26, 0);
        cnt[toNum(p[p.size()-1])] = 1;
        for(int i=p.size()-2, tmp=1, next=toNum(p[p.size()-
1]); i>=0; --i){
            int k = toNum(p[i]);
            if((k+1)%26 == next) ++tmp;
            else tmp = 1;
            cnt[k] = max(cnt[k], tmp);
            next = k;
        }
        for(int i=0; i<26; ++i) ans += cnt[i];
        return ans;
    }
};
```

692

1. Classic problem, min queue + hashmap

```
class Solution {
    public List<String> topKFrequent(String[] words, int k) {
        List<String> res = new ArrayList<>();
        if (words == null || words.length == 0) {
            return res;
        }

        Map<String, Integer> map = new HashMap<>();

        for (String str : words) {
            if (!map.containsKey(str)) {
                map.put(str, 0);
            }
            map.put(str, map.get(str) + 1);
        }

        PriorityQueue<Map.Entry<String, Integer>> pq = new Pri
orityQueue<>((a,b) -> {
            if (a.getValue() != b.getValue()) {
                return a.getValue() - b.getValue();
            } else {
                return b.getKey().compareTo(a.getKey());
            }
        });

        for (Map.Entry<String, Integer> m : map.entrySet()) {
            if (pq.size() < k) {
                pq.add(m);
            } else {
                Map.Entry<String, Integer> cur = pq.peek();
                if (m.getValue() > cur.getValue()) {
                    pq.poll();
                    pq.add(m);
                }
                if (m.getValue() == cur.getValue() && m.getKey
().compareTo(cur.getKey()) < 0) {
                    pq.poll();
                    pq.add(m);
                }
            }
        }

        for (int i = 0; i < k; i++) {
            res.add(pq.poll().getKey());
        }



        Collections.reverse(res);
        return res;
    }
}
```

2. GoodSpeed's solution:
    a. lang: c++
    b. 计数+排列:

```
class Solution {
    typedef pair<int, string> is;
public:
    vector<string> topKFrequent(vector<string>& words, int k)
{
        set<is> S;
        map<string, int> cnt;
        for(string s: words) ++cnt[s];
        for(auto p: cnt) S.insert(is(-p.second, p.first));
        vector<string> ans;
        for(auto p: S){
            ans.push_back(p.second);
            if(ans.size()==k) break;
        }
        return ans;
    }
};
```

330

1. Greedy

☐ See other people (Leetcode discussion) solution @Zebo L

429 (empty)

1. Can’t find this problem.

200

1. BFS

```
class Solution {
    int di[4]={1,-1,0,0}, dj[4]={0,0,1,-1}, m, n;
public:
    int numIslands(vector<vector<char>>& grid) {
        if(grid.empty() || grid[0].empty()) return 0;
        n = (int)grid.size();
        m = (int)grid[0].size();
        unordered_set<int> S;
        queue<int> Q;
        for(int i=0;i<n;++i) for(int j=0;j<m;++j) if(grid[i][j]=='1') S.insert(i*m+j);
        int ans = 0;
        while(!S.empty()){
            int s = *S.begin();
            ++ans;
            S.erase(s);
            Q.push(s);
            while(!Q.empty()){
                int i = Q.front()/m, j = Q.front()%m;
                Q.pop();
                for(int k=0;k<4;++k){
                    int i1 = i + di[k], j1 = j + dj[k];
                    if(i1>=0 && j1>=0 && i1<n && j1<m && grid
[i1][j1]=='1' && S.count(i1*m + j1)){
                        Q.push(i1*m + j1);
                        S.erase(i1*m + j1);
                    }
                }
            }
        }
        return ans;
    }
};
```

2. Union Find

```
class Solution {
    vector<int> P;
    int getRoot(int i){
        if(P[i]==i) return i;
        return P[i]=getRoot(P[i]);
    }
public:
    int numIslands(vector<vector<char>>& grid) {
        if(grid.empty() || grid[0].empty()) return 0;
        int n = grid.size(), m = grid[0].size(), ans = 0;
        P.resize(m*n);
        for(int k=0;k<m*n;++k) if(grid[k/m][k%m] == '1'){
            P[k] = k;
            if(k/m && grid[k/m-1][k%m]=='1') if(getRoot(k)!=ge
tRoot(k-m)) P[getRoot(k)] = getRoot(k-m);
            if(k%m && grid[k/m][k%m-1]=='1') if(getRoot(k)!=ge
tRoot(k-1)) P[getRoot(k)] = getRoot(k-1);
        }
        for(int i=0;i<n*m;++i) if(P[i]==i && grid[i/m][i%m]=
='1') ++ans;
        return ans;
    }
};
```

256

1. typical dp. Greedy would be wrong. Need to have a n * 3 2d array

```
class Solution {
    public int minCost(int[][] costs) {
        if(costs == null || costs.length == 0 || costs[0] == n
ull || costs[0].length == 0) {
            return 0;
        }
        int n = costs.length;

        int[][] dp = new int[n][3];
        dp[0][0] = costs[0][0];
        dp[0][1] = costs[0][1];
        dp[0][2] = costs[0][2];

        for (int i = 1; i < n; i++) {
            dp[i][0] = costs[i][0] + Math.min(dp[i - 1][1], dp
[i - 1][2]);
            dp[i][1] = costs[i][1] + Math.min(dp[i - 1][0], dp
[i - 1][2]);
            dp[i][2] = costs[i][2] + Math.min(dp[i - 1][0], dp
[i - 1][1]);
        }

        return Math.min(Math.min(dp[n - 1][0], dp[n - 1][1]),
dp[n - 1][2]);
```

```
    }
}
```

2. GoodSpeed’s solution:
    a. lang: c++
    b. idea: 同上but `O(1)` space:

```
class Solution {
public:
    int minCost(vector<vector<int>>& costs) {
        if(costs.empty() || costs[0].empty()) return 0;
        int n = costs.size();
        vector<int> C = costs[0];
        for(int i=1;i<n;++i){
            vector<int> tmp(C);
            C[0] = costs[i][0] + min(tmp[1], tmp[2]);
            C[1] = costs[i][1] + min(tmp[0], tmp[2]);
            C[2] = costs[i][2] + min(tmp[0], tmp[1]);
        }
        return min(C[0], min(C[1], C[2]));
    }
};
```

321

1. GoodSpeed’s solution:
    a. lang: c++:
    b. dp, but 最后一个case始终过不了：

```
class Solution {
    map<int, map<int, map<int, string>>> dp;
    string getMax(int i, int j, int k, vector<int>& nums1, vector<int>& nums2){
        if(!k) return "";
        if(dp.count(i) && dp[i].count(j) && dp[i][j].count(k)) return dp[i][j][k];
        int n1=nums1.size(), n2=nums2.size(), start_1 = i, start_2 = j, cand_1=-1, cand_2=-1;
        if(i<n1){
            for(int r=i+1;r<=n1+n2-k-j && r<n1;++r) if(nums1[r]>nums1[start_1]) start_1=r;
            cand_1 = nums1[start_1];
        }
        if(j<n2){
            for(int r=j+1;r<=n1+n2-k-i && r<n2;++r) if(nums2[r]>nums2[start_2]) start_2=r;
            cand_2 = nums2[start_2];
        }
        if(cand_1 > cand_2) return dp[i][j][k] = to_string(cand_1) + getMax(start_1+1, j, k-1, nums1, nums2);
        if(cand_1 < cand_2) return dp[i][j][k] = to_string(cand_2) + getMax(i, start_2+1, k-1, nums1, nums2);
        return dp[i][j][k] = to_string(cand_1) + max(getMax(start_1+1, j, k-1, nums1, nums2), getMax(i, start_2+1, k-1, nums1, nums2));
    }
public:
    vector<int> maxNumber(vector<int>& nums1, vector<int>& nums2, int k) {
        string s = getMax(0, 0, k, nums1, nums2);
        vector<int> ans;
        for(char c: s) ans.push_back(int(c-'0'));
        return ans;
    }
};
```

- [ ] Look at Leetcode discussion @Zebo L

77

1. GoodSpeed's solution
    a. lang: c++
    b. idea: Easy recursion

```
class Solution {
public:
    vector<vector<int>> combine(int n, int k) {
        vector<vector<int>> ans;
        if(k>n) return ans;
        if(!k){
            ans.push_back(vector<int>());
            return ans;
        }
        if(k==n){
            vector<int> tmp;
            for(int i=1;i<=k;++i) tmp.push_back(i);
            ans.push_back(tmp);
            return ans;
        }
        ans = combine(n-1, k);
        auto pre = combine(n-1, k-1);
        for(auto tmp: pre){
            tmp.push_back(n);
            ans.push_back(tmp);
        }
        return ans;
    }
};
```

409:

1. GoodSpeed's solution:
    a. lang: c++

b. idea: 计数问题

```
class Solution {
public:
    int longestPalindrome(string s) {
        map<char, int> cnt;
        for(auto c: s) ++cnt[c];
        int add = 0, ans = 0;
        for(auto p: cnt){
            ans += (p.second/2)*2;
            if(p.second%2) add = 1;
        }
        return ans + add;
    }
};
```