July 20, 2018 题目：685,689,312,378,643,673,621,587,232,125,580,533

685

1. 直接暴力能过，不过非常慢。

```
class Solution {
    vector<unordered_set<int>> E;
    bool isTree(int root, int n){
        queue<int> Q;
        unordered_set<int> pass;
        Q.push(root);
        pass.insert(root);
        while(!Q.empty()){
            int j = Q.front();
            Q.pop();
            for(int k: E[j]) if(!pass.count(k)){
                pass.insert(k);
                Q.push(k);
            }
        }
        return (int)pass.size() == n;
    }
public:
    vector<int> findRedundantDirectedConnection(vector<vector<int>>& edges) {
        int n = edges.size(), root = 0;
        E.resize(n+1);
        vector<bool> notRoot(n+1, false);
```

```
        for(auto vec: edges) {
            E[vec[0]].insert(vec[1]);
            notRoot[vec[1]] = true;
        }
        for(int i=1; i<=n; ++i) if(!notRoot[i]) root = i;
        for(int i=n-1; i>=0; --i){
            E[edges[i][0]].erase(edges[i][1]);
            int rt = (root? root: edges[i][1]);
            if(isTree(rt, n)) return edges[i];
            E[edges[i][0]].insert(edges[i][1]);
        }
        return edges[0];
    }
};
```

- [ ] 想想如何改进 @Zebo L

2. UnionFind

```
class Solution {
    Map<Integer, Integer> parents = new HashMap<>();
    public int[] findRedundantDirectedConnection(int[][] edge
s) {

        for (int[] edge : edges) {
            if (!parents.containsKey(edge[0])) {
                parents.put(edge[0], edge[0]);
            }

            if (!parents.containsKey(edge[1])) {
                parents.put(edge[1], edge[1]);
            }
        }
```

```
        int[] can1 = null;
        int[] can2 = null;
        for (int[] edge : edges) {
            int i = find(edge[0]), j = find(edge[1]);
            if (i != j) {
                if (j != edge[1]) can1 = edge;
                else parents.put(edge[1], edge[0]);
            } else {
                can2 = edge;
            }
        }
        if (can1 == null) return can2;
        if (can2 == null) return can1;

        for (int[] edge : edges) {
            if (edge[1] == can1[1]) return edge;
        }
        return new int[]{};
    }

    public int find(int k) {
        int parent = k;
        while (parent != parents.get(parent)) {
            parent = parents.get(parent);
        }
        return parent;
    }
}
```

1. 跟股票那题相似，难点在于要back track， 因为output 是 `vector<int>` ， 很容易出错。

```
class Solution {
public:
    vector<int> maxSumOfThreeSubarrays(vector<int>& nums, int k) {
        int n = nums.size();
        vector<int> P(n+1, 0), ans;
        vector<vector<int>> dp(4, vector<int>(n+1, 0));
        for(int i=1; i<=n; ++i) P[i] = P[i-1] + nums[i-1];
        for(int r=1; r<=3; ++r){
            for(int i=r*k, m = dp[r-1][i-k] - P[i-k]; i<n+1; ++i){
                m = min(m, dp[r-1][i-k] - P[i-k]);
                dp[r][i] = max(dp[r][i-1], P[i] + m);
            }
        }
        for(int r=3, idx=n; r>0; --r){
            int j = idx;
            while(j>0 && dp[r][j] == dp[r][j-1]) --j;
            ans.push_back(max(0, j-k));
            idx = j-k;
        }
        reverse(ans.begin(), ans.end());
        return ans;
    }
};
```

- [ ] 看Leetcode discussion @Zebo L

312

1. dp:
   - `dp[i][j]`: The maximum score to burst balloons from `i + 1` to `j - 1`.

- Assume the last bursted balloon is `k`, then `dp[i][j] = nums[k] * nums[i] * nums[j] + dp[i][k] + dp[j][k]`
- So the recursion should be:

`dp[i][j] = max_k{nums[k] * nums[i] * nums[j] + dp[i][k] + dp[j][k]}`

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

2. DP

```
class Solution {
    public int maxCoins(int[] nums) {
        int[] numsUp = new int[nums.length + 2];
        int index = 1;
        for (int n : nums) {
            if (n > 0) numsUp[index++] = n;
        }
        numsUp[0] = numsUp[index++] = 1;
        int[][] dp = new int[index][index];
        for (int gap = 2; gap < index; gap++) {
```

```
            for (int i = 0; i < index - gap; i++) {
                int j = i + gap;
                for (int k = i + 1; k < j; k++) {
                    dp[i][j] = Math.max(dp[i][j], numsUp[i] *
numsUp[k] * numsUp[j] + dp[i][k] + dp[k][j]);
                }
            }
        }
        return dp[0][index - 1];
    }
}
```

378

1.

```
class Solution {
    class Point{
        int x;
        int y;
        int val;
        public Point(int x, int y, int val) {
            this.x = x;
            this.y = y;
            this.val = val;
        }
    }
    public int kthSmallest(int[][] matrix, int k) {
        PriorityQueue<Point> pq = new PriorityQueue<>(1, new C
omparator<Point>(){
        @Override
        public int compare(Point a, Point b) {
            return a.val - b.val;
```

```
        }

        });
        int n = matrix.length;
        for (int i = 0; i < n; i++) {
            pq.offer(new Point(i, 0, matrix[i][0]));
        }

        for (int i = 0; i < k - 1; i++) {
            Point p = pq.poll();
            if (p.y == n - 1) continue;
            pq.offer(new Point(p.x, p.y + 1, matrix[p.x][p.y +
1]));
        }
        return pq.poll().val;
    }
}
```

2. 二分，但是注意 `l + r` 容易 overflow

```
class Solution {
public:
    int kthSmallest(vector<vector<int>>& matrix, int k) {
        int n = matrix.size();
        //if(k==1) return matrix[0][0];
        long l = matrix[0][0]-1, r = matrix[n-1][n-1] + 1;
        while(l < r-1){
            int c = (l + r)/2;
            int cnt = 0;
            for(int i=n-1, j=0; i>=0; --i){
                for(; j<n && matrix[i][j] < c; ++j);
                cnt += j;
```

```
            }
            if(cnt <= k-1) l = c;
            else r = c;
        }
        return l;
    }
};
```

643

1.

```
class Solution {
    public double findMaxAverage(int[] nums, int k) {
        if (k > nums.length) return 0;
        int sum = 0;

        for (int i = 0; i < k; i++) {
            sum += nums[i];
        }
        double average = (double) sum / k;

        for (int i = k; i < nums.length; i++) {
            sum = sum - nums[i - k] + nums[i];
            average = Math.max(average, (double) sum / k);
        }
        return average;

    }
}
```

2. Sliding window:

```
class Solution {
public:
```

```
    double findMaxAverage(vector<int>& nums, int k) {
        int ans = 0;
        for(int i=0; i<k ; ++i) ans += nums[i];
        for(int i=k, tmp=ans; i<nums.size(); ++i){
            tmp += nums[i] - nums[i-k];
            ans = max(ans, tmp);
        }
        return double(ans)/k;
    }
};
```

673

1. 很有意思的一道题，记录LIS header的同时，track 各个长度以不同数结尾的 LIS的个数：

```
class Solution {
public:
    int findNumberOfLIS(vector<int>& nums) {
        if(nums.empty()) return 0;
        vector<int> ans;
        vector<map<int, int>> cnt;
        for(int n: nums){
            if(ans.empty() || n > ans.back()) {
                ans.push_back(n);
                cnt.push_back(map<int, int>());
            }
            int l = -1, r = ans.size() - 1;
            while(l < r-1){
                int c = (r+l)/2;
                if(ans[c] < n) l = c;
                else r = c;
            }
```

```
            ans[r] = n;
            if(r==0) cnt[0][n]++;
            else{
                for(auto p: cnt[r-1]) if(p.first<n) cnt[r][n]
+= p.second;
            }
        }
        int res = 0;
        for(auto p: cnt[ans.size()-1]) res += p.second;
        return res;
    }
};
```

621

1. 就是计数问题

```
class Solution {
public:
    int leastInterval(vector<char>& tasks, int n) {
        int max_cnt = 0, top = 0;
        unordered_map<char, int> cnt;
        for(char c: tasks) {
            cnt[c]++;
            max_cnt = max(max_cnt, cnt[c]);
        }
        for(auto p: cnt) if(p.second == max_cnt) ++top;
        return max((int)tasks.size(), (max_cnt-1)*(n+1) + to
p);
    }
};
```

587

1. 几何题，就是比较烦，细心点别搞错

```
struct Rot{
    const double delta = 1E-7;
    int x, y;
    Rot(int x_, int y_) : x(x_), y(y_) {}
    void set(int x_, int y_){
        x = x_;
        y = y_;
    }
    bool compare(int x1, int y1, int x2, int y2){
        double cos1 = (x1 * x + y1 * y) / sqrt(x1*x1 + y1*y1)
/ sqrt(x*x + y*y);
        double cos2 = (x2 * x + y2 * y) / sqrt(x2*x2 + y2*y2)
/ sqrt(x*x + y*y);
        if(fabs(cos1 - cos2) < this->delta) return x1*x1 + y1*
y1 < x2*x2 + y2*y2;
        else return cos1 > cos2;
    }
};

bool pointCmp(const Point&p1, const Point&p2){
    if(p1.y == p2.y) return p1.x < p2.x;
    return p1.y < p2.y;
}

bool equ(const Point&p1, const Point&p2){
    return p1.x==p2.x && p1.y==p2.y;
}

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

```
public:
    vector<Point> outerTrees(vector<Point>& points) {
        vector<Point> ans;
        vector<vector<bool>> used(105, vector<bool>(105, fals
e));
        int n = points.size();
        if(n<=2) return points;
        Point start = points[0];
        for(int i=1; i<n; ++i) if(pointCmp(points[i], start))
start = points[i];
        ans.push_back(start);
        Rot rot(1, 0);
        while(true){
            Point pivot = ans.back(), next=Point(200, 200);
            for(auto p: points) if(!equ(p, pivot) && !used[p.
x][p.y]){
                if(next.x == 200) next = p;
                else if(rot.compare(p.x-pivot.x, p.y-pivot.y,
next.x-pivot.x, next.y-pivot.y)){
                    next = p;
                }
            }
            if(equ(next, ans[0])) return ans;
            ans.push_back(next);
            used[next.x][next.y] = true;
            rot.set(next.x-pivot.x, next.y-pivot.y);
        }

        return ans;
    }
};
```

- [ ] 看看Leetcode discussion 中的solution @Zebo L

232

1. 两个stack 来回倒

```
class MyQueue {
public:
    /** Initialize your data structure here. */
    stack<int> Q1, Q2;
    int front;
    MyQueue() {

    }

    /** Push element x to the back of queue. */
    void push(int x) {
        if(Q1.empty()) front = x;
        Q1.push(x);
    }

    /** Removes the element from in front of queue and returns
that element. */
    int pop() {
        while(!Q1.empty()){
            Q2.push(Q1.top());
            Q1.pop();
        }
        int x = Q2.top();
        Q2.pop();
        if(!Q2.empty()) front = Q2.top();
        while(!Q2.empty()){
            Q1.push(Q2.top());
```

```
            Q2.pop();
        }
        return x;
    }

    /** Get the front element. */
    int peek() {
        return front;
    }

    /** Returns whether the queue is empty. */
    bool empty() {
        return Q1.empty();
    }
};
```

125

1. 注意下数字即可：

```
class Solution {
    int toNum(char c){
        if(c>='a' && c<='z') return (int)(c - 'a');
        else if(c>='A' && c<='Z') return (int)(c - 'A');
        else if(c>='0' && c<='9') return 100 + (int)(c - '0');
        else return -1;
    }
public:
    bool isPalindrome(string s) {
        int i = 0, j = s.size()-1;
        while(i < j){
            while(i<j && toNum(s[i])==-1) ++i;
```

```
            while(j>i && toNum(s[j])==-1) --j;
            if(toNum(s[i]) != toNum(s[j])) return false;
            ++i;
            --j;
        }
        return true;
    }
};
```

580

SQL

533

1. 计数问题

```
class Solution {
public:
    int findBlackPixel(vector<vector<char>>& picture, int N) {
        int n = picture.size(), m = picture[0].size(), ans =
0;
        unordered_set<int> col;
        unordered_map<string, int> group;
        for(int j=0; j<m ; ++j){
            int cnt = 0;
            for(int i=0; i<n; ++i) if(picture[i][j] == 'B') ++
cnt;
            if(cnt == N) col.insert(j);
        }
        for(int i=0; i<n; ++i){
            string s;
            int cnt = 0;
            for(char c: picture[i]){
```

```
                s += c;
                if(c == 'B') ++cnt;
            }
            if(cnt == N) ++group[s];
        }
        for(auto p: group) if(p.second == N){
            for(int j=0; j<m; ++j) if(p.first[j] == 'B' && co
l.count(j)) ans += N;
        }
        return ans;
    }
};
```