3 Dynamic Programming

3.1 problem list

- 来源列表：LeetCode – 贪心。

id	title	label
1	p122 买卖股票的最佳时机 II easy	`
2	p860 柠檬水找零 easy	
3	p455 分发饼干 easy	
4	p874 模拟行走机器人 easy	
5	p861 翻转矩阵后的得分 normal	**推荐**
6	p392 判断子序列 normal	
7	p134 加油站 normal	**推荐**
8	p452 用最少数量的箭引爆气球 normal	
9	p435 无重叠区间 normal	

3.2 solution

3.2.1 p122 买卖股票的最佳时机

贪心

$O(N)$。

由于不能同时持有多支股票。

所以相邻两天的差价最多只能被获利一次。

贪心取过去即可。

```
class Solution {
public:
    int maxProfit(vector<int>& prices) {
        int result = 0;
        for (int i = 1; i < prices.size(); i++) {
            result += max(0, prices[i] - prices[i - 1]);
        }
        return result;
    }
};
```

3.2.2 p860 柠檬水找零

贪心

$O(N)$。

优先找大钞。

```
class Solution {
public:
    bool lemonadeChange(vector<int>& bills) {
        int cnt10 = 0, cnt5 = 0;
        for (auto b : bills) {
            cnt5 += b == 5;
            cnt10 += b == 10;
            int need = b - 5;
            if (need >= 10 && cnt10) {
                cnt10--;
                need -= 10;
            }
            if (need >= 5) {
                cnt5 -= need / 5;
            }
            if (cnt5 < 0) return false;
        }
        return true;
    }
};
```

3.2.3 p445 分发饼干

贪心

$O(NlogN)$。

从小到大考虑每块饼干，假设当前大小为 s：

- 把所有 $g <= s$ 的小孩纳入考虑。

- 如果这里面某个小孩没拿到饼干，则分给他这个饼干。

正确性同学们自己努力感受下~

```
class Solution {
public:
    int findContentChildren(vector<int>& g, vector<int>& s) {
        sort(g.begin(), g.end());
        sort(s.begin(), s.end());
        int wait = 0, cur = 0, result = 0;
        for (auto e : s) {
            while (cur < g.size() && g[cur] <= e) {
                cur++;
                wait++;
            }
            if (wait) {
                wait--;
                result++;
            }
        }
        return result;
    }
};
```

3.2.4 p874 模拟行走机器人

贪心

$O(9 * NlogN)$。

暴力模拟，能走就走。

把障碍物扔进 Set 加快碰撞判定。

```
class Solution {

public:
    int robotSim(vector<int>& commands, vector<vector<int>>& obstacles) {
        int x = 0, y = 0, result = 0;
        set<pair<int, int> > obstacles_set;
        for (auto e : obstacles) {
            obstacles_set.insert({e[0], e[1]});
        }
        int g[4][2] = {{0, 1}, {1, 0}, {0, -1}, {-1, 0}}, dir = 0;
        for (auto c : commands) {
            switch (c) {
                case -2:
                    dir = (dir + 3) % 4;
                    break;
                case -1:

                    dir = (dir + 1) % 4;
```

```
                    break;
                default :
                    while (c--) {
                        int nx = x + g[dir][0];
                        int ny = y + g[dir][1];
                        if (obstacles_set.find({nx, ny}) != obstacles_set.end()) {
                            break;
                        }
                        tie(x, y) = {nx, ny};
                    }
                    break;
            }
            result = max(result, x * x + y * y);
        }
        return result;
    }
};
```

3.2.5 p861 翻转矩阵后的得分

贪心

$O(N^2)$。

根据贪心思想肯定第一列全是1，所以行变换等于确定了。

在此目标下有第一列是否翻转两种情况，枚举一下即可。

```
class Solution {
private:
    void rot_row(int row, vector<vector<int> > &A) {
        for (int col = 0; col < A[0].size(); col++) {
            A[row][col] ^= 1;
        }
    }
    void rot_col(int col, vector<vector<int> > &A) {
        for (int row = 0; row < A.size(); row++) {
            A[row][col] ^= 1;
        }
    }
public:
    int matrixScore(vector<vector<int> >& A) {
        if (A.empty()) return 0;
        int n = A.size(), m = A[0].size(), result = 0;
        for (int time = 0; time < 2; time++) {
            for (int row = 0; row < n; row++) if (A[row][0] == 1) {
                rot_row(row, A);
            }
            int temp = 0;
            for (int col = 0; col < m; col++) {
                int cnt = 0;
                for (int row = 0; row < n; row++) {
```

```
                    cnt += A[row][col];
                }
                temp += (1 << (m - 1 - col)) * max(cnt, n - cnt);
            }
            result = max(result, temp);
            rot_col(0, A);
        }
        return result;
    }
};
```

暴力

$O(2^N N)$。

枚举列变换的情况。

然后每行检查过去，取翻转/不翻转该行两种情况的最大值。

```
class Solution {
public:
    int matrixScore(vector<vector<int> >& A) {
        if (A.empty()) return 0;
        int n = A.size(), m = A[0].size(), result = 0;
        vector<int> B;
        for (int row = 0; row < n; row++) {
            int temp = 0;
            for (int col = 0; col < m; col++) {
                temp = temp << 1 | A[row][col];
            }
            B.push_back(temp);
        }
        for (int mask_col = 0; mask_col < (1 << m); mask_col++) {
            int temp = 0;
            for (int row = 0; row < n; row++) {
                temp += max(B[row] ^ mask_col, B[row] ^ mask_col ^ ((1 << m) - 1));
            }
            result = max(result, temp);
        }
        return result;
    }
};
```

3.2.6 p392 判断子序列

贪心

$O(N)$。

对 t 进行扫描，能匹配就匹配。

```
class Solution {
public:
    bool isSubsequence(string s, string t) {
        int cur = 0;
        for (auto c : t) {
            if (cur < s.size() && c == s[cur]) {
                cur++;
            }
        }
        return cur >= s.size();
    }
};
```

3.2.7 p134 加油站

贪心

$O(N)$。

假设起点为 $start$ ，枚举走到第 i 个点的时候油不够了；

则测试起点为 $start+1$ ~ i 的这些方案，走到第 i 个点的时候油也会不够。

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

3.2.8 p452 用最少数量的箭引爆气球

贪心

$O(NlogN)$。

对所有的区间排序；

然后每次拿进来一个区间；

保证目前手上的区间存在公共点的时候不断的拿即可。

```
class Solution {
public:
    int findMinArrowShots(vector<pair<int, int>>& points) {
        int result = 0;
        sort(points.begin(), points.end());
        for (int l = 0, r; l < points.size(); l = r) {
            r = l;
            int minr = INT_MAX;
            while (r < points.size() && points[r].first <= min(minr, points[r].second)) {
                minr = min(minr, points[r++].second);
            }
            result++;
        }
        return result;
    }
};
```

3.2.9 p435 无重叠区间

贪心

$O(NlogN)$。

问题等价于找出最多的不重叠区间子集。

排序区间后从小到大考虑。

不冲突则直接拿走，如果冲突（必然只和上次拿的有交集）；

那么看下谁的右端点小留谁（根据贪心思想）；

```
class Solution {
public:
    int eraseOverlapIntervals(vector<Interval>& intervals) {
        int result = 0;
        vector<pair<int, int> > v;
        for (auto e : intervals) {
            v.push_back({e.start, e.end});
        }
        sort(v.begin(), v.end());
        int curR = -INT_MAX;
        for (auto e : v) {
            if (curR <= e.first) {
                curR = e.second;
                result++;
            }
            else {
                curR = min(curR, e.second);
            }
        }

        return intervals.size() - result;
```

```
    }
};
```