# 贪心

## 455 分发饼干

class Solution {

public:

    int findContentChildren(vector<int>& g, vector<int>& s) {

        sort(g.begin(), g.end());

        sort(s.begin(), s.end());

        int id\_g = 0, id\_s = 0;

        while(id\_g < g.size() && id\_s < s.size()) {

            if(s[id\_s] >= g[id\_g]) {

                ++id\_s;

                ++id\_g;

            }

            else {

                ++id\_s;

            }

        }

        return id\_g;

    }

};

## 135 分发糖果

class Solution {

public:

    int candy(vector<int>& ratings) {

        int n = ratings.size();

        vector<int> candy(n, 1);

        for(int i=1; i<n; ++i) {

            if(ratings[i] > ratings[i-1]) {

                candy[i] = candy[i-1] + 1;

            }

        }

        for(int i=n-2; i>=0; --i) {

            if(ratings[i] > ratings[i+1]) {

                candy[i] = max(candy[i], candy[i+1] + 1);

            }

        }

        return accumulate(candy.begin(), candy.end(), 0);

    }

};

## 605 种花问题

这个题虽然简单，但是如果思考放花的条件，还是比较麻烦的。这时候想起了在验证二叉搜索树里面的一个小技巧。我们可以考虑问题的反面，什么时候不能放花呢？排除掉不能放的情况，当然就能放啦。代码如下：

class Solution {

public:

    bool canPlaceFlowers(vector<int>& flowerbed, int n) {

        int num = 0;

        for(int i=0; i<flowerbed.size(); ++i) {

            if(flowerbed[i] == 0) {

                if(i-1 >=0 && flowerbed[i-1] == 1) continue;

                if(i+1 < flowerbed.size() && flowerbed[i+1] == 1) continue;

                flowerbed[i] = 1;

                ++num;

            }

        }

        return num >= n;

    }

};

## 763 划分字母区间

这道题类似跳跃游戏，也是双指针类型的贪心问题

class Solution {

public:

    vector<int> partitionLabels(string S) {

        vector<int> res;

        unordered\_map<char, int> char\_to\_last;

        int n = S.size();

        for(int i=0; i<n; ++i) {

            char\_to\_last[S[i]] = i;

        }

        int start = 0, end = 0;

        for(int i=0; i<n; ++i) {

            end = max(end, char\_to\_last[S[i]]);

            if( i == end ) {

                res.push\_back(end-start+1);

                start = end + 1;

                end = end + 1;

            }

        }

        return res;

    }

};