

135-分发糖果

题述

135. 分发糖果

难度 **困难** 756 ☆ 讨论 笔记 题解 1

n 个孩子站成一排。给你一个整数数组 `ratings` 表示每个孩子的评分。

你需要按照以下要求，给这些孩子分发糖果：

- 每个孩子至少分配到 1 个糖果。
- 相邻两个孩子评分更高的孩子会获得更多的糖果。

请你给每个孩子分发糖果，计算并返回需要准备的 **最少糖果数目**。

示例 1:

输入: `ratings = [1,0,2]`

输出: 5

解释: 你可以分别给第一个、第二个、第三个孩子分发 2、1、2 颗糖果。

示例 2:

输入: `ratings = [1,2,2]`

输出: 4

解释: 你可以分别给第一个、第二个、第三个孩子分发 1、2、1 颗糖果。

第三个孩子只得到 1 颗糖果，这满足题面中的两个条件。

提示:

- `n == ratings.length`
- `1 <= n <= 2 * 104`
- `0 <= ratings[i] <= 2 * 104`

思路

这道题目一定是要确定一边之后，再确定另一边，例如比较每一个孩子的左边，然后再比较右边，**如果两边一起考虑一定会顾此失彼。**

先确定右边评分大于左边的情况（也就是从前向后遍历）

此时局部最优：只要右边评分比左边大，右边的孩子就多一个糖果，全局最优：相邻的孩子中，评分高的右孩子获得比左边孩子更多的糖果

局部最优可以推出全局最优。

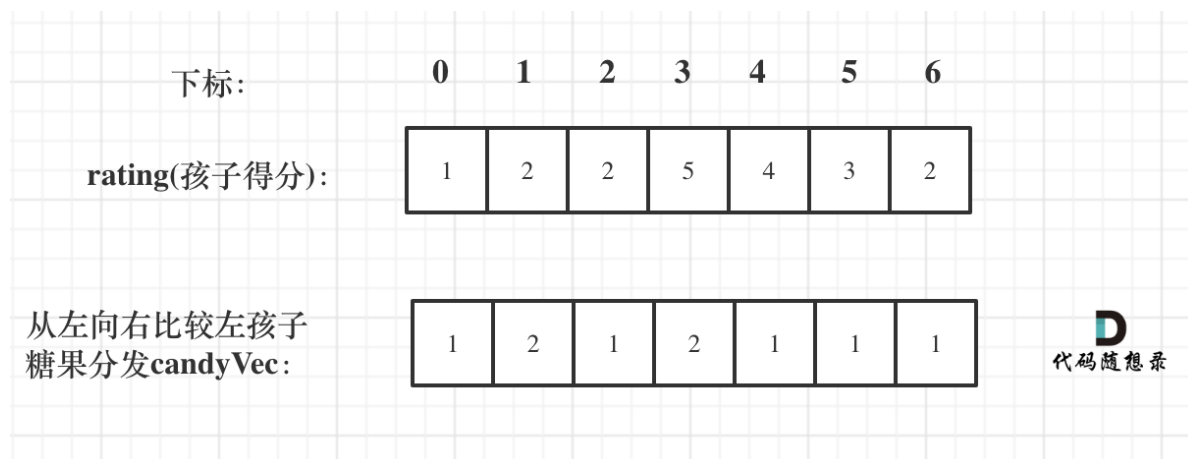
如果 $\text{ratings}[i] > \text{ratings}[i - 1]$ 那么 i 的糖 一定要比 $i - 1$ 的糖多一个，所以贪心： $\text{candyVec}[i] = \text{candyVec}[i - 1] + 1$

代码如下：

```
// 从前向后
for (int i = 1; i < ratings.size(); i++) {
    if (ratings[i] > ratings[i - 1]) candyVec[i] = candyVec[i - 1] + 1;
}
```

如图：

下标：	0	1	2	3	4	5	6
rating(孩子得分)：	1	2	2	5	4	3	2
从左向右比较左孩子糖果分发candyVec：	1	2	1	2	1	1	1



再确定左孩子大于右孩子的情况（从后向前遍历）

遍历顺序这里有同学可能会有疑问，为什么不能从前向后遍历呢？

因为如果从前向后遍历，根据 $\text{ratings}[i + 1]$ 来确定 $\text{ratings}[i]$ 对应的糖果，那么每次都不能利用上一次的比较结果了。

所以确定左孩子大于右孩子的情况一定要从后向前遍历！

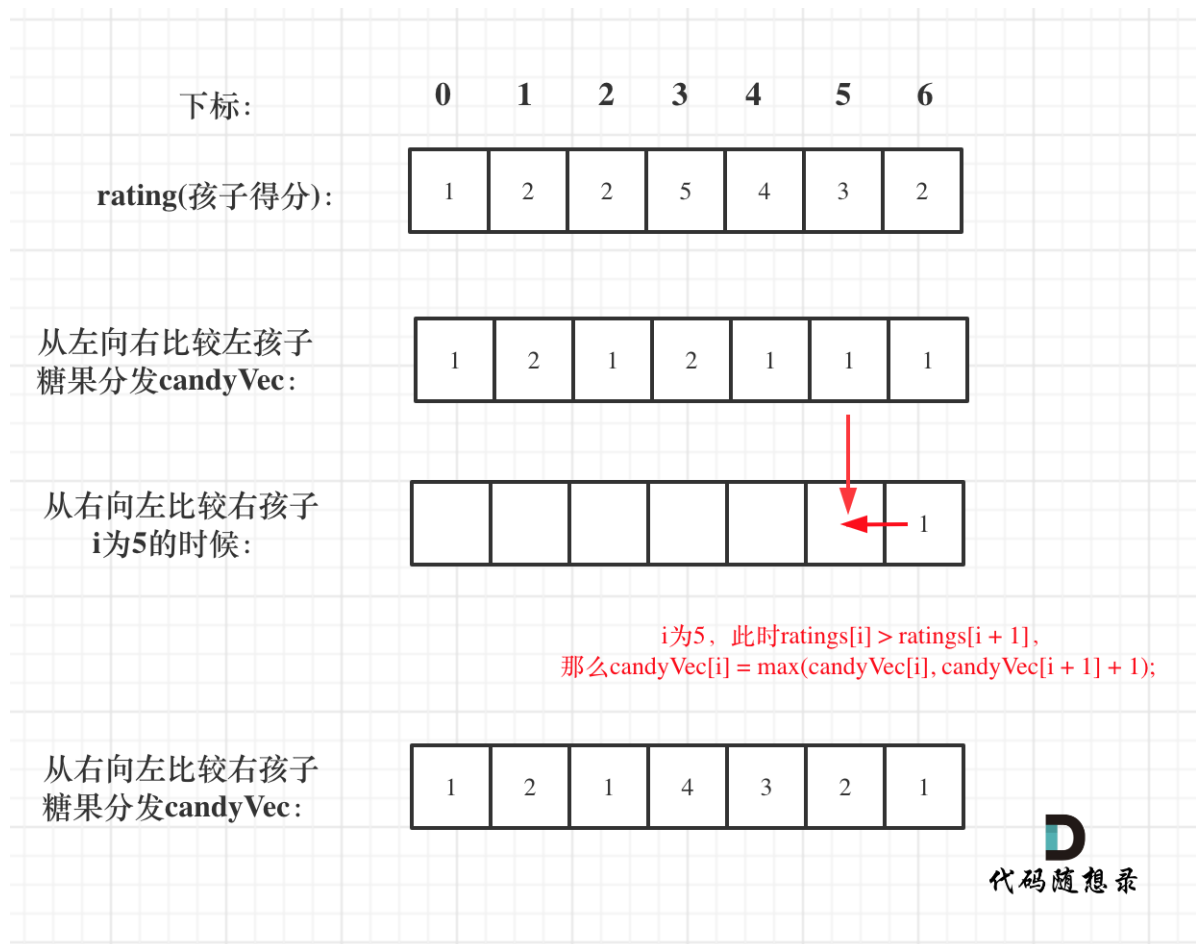
如果 $\text{ratings}[i] > \text{ratings}[i + 1]$ ，此时 $\text{candyVec}[i]$ （第 i 个小孩的糖果数量）就有两个选择了，一个是 $\text{candyVec}[i + 1] + 1$ （从右边这个加1得到的糖果数量），一个是 $\text{candyVec}[i]$ （之前比较右孩子大于左孩子得到的糖果数量）。

那么又要贪心了，局部最优：取 $\text{candyVec}[i + 1] + 1$ 和 $\text{candyVec}[i]$ 最大的糖果数量，保证第 i 个小孩的糖果数量即大于左边的也大于右边的。全局最优：相邻的孩子中，评分高的孩子获得更多的糖果。

局部最优可以推出全局最优。

所以就取 $\text{candyVec}[i + 1] + 1$ 和 $\text{candyVec}[i]$ 最大的糖果数量， **$\text{candyVec}[i]$ 只有取最大的才能既保持对左边 $\text{candyVec}[i - 1]$ 的糖果多，也比右边 $\text{candyVec}[i + 1]$ 的糖果多。**

如图：



D
代码随想录

所以该过程代码如下:

```
// 从后向前
for (int i = ratings.size() - 2; i >= 0; i--) {
    if (ratings[i] > ratings[i + 1]) {
        candyVec[i] = max(candyVec[i], candyVec[i + 1] + 1);
    }
}
```

题解

贪心C++

```
class Solution {
public:
    int candy(vector<int>& ratings)
    {
        //两次贪心
        //一次是从左到右遍历, 只比较右边孩子评分比左边大的情况。
        //一次是从右到左遍历, 只比较左边孩子评分比右边大的情况。
        vector<int> candy(ratings.size(), 1);
        //从前向后遍历贪心
        for(int i = 1; i < ratings.size(); i++)
        {
            if(ratings[i] > ratings[i-1])
            {
                //右边评分高于左边
                candy[i] = candy[i-1] + 1;
            }
        }
    }
};
```

```

    }
}

//从后向前遍历贪心
for(int i = ratings.size()-2; i >= 0; i--)
{
    if(ratings[i] > ratings[i+1])
    {
        //左边评分高于右边
        candy[i] = max(candy[i],candy[i+1]+1); //这时需要让中间的高于两边的
    }
}

//统计结果并返回
int result = 0;
for(int i=0;i<candy.size();i++)
{
    result += candy[i];
}
return result;
}
};

```

执行结果: **通过** [显示详情 >](#)

[添加备注](#)

执行用时: **20 ms** , 在所有 C++ 提交中击败了 **49.36%** 的用户

内存消耗: **17.3 MB** , 在所有 C++ 提交中击败了 **53.47%** 的用户

通过测试用例: **48 / 48**

炫耀一下:



[写题解, 分享我的解题思路](#)

提交结果	执行用时	内存消耗	语言	提交时间	备注
通过	20 ms	17.3 MB	C++	2022/02/14 15:02	添加备注

Python

```
class Solution:
    def candy(self, ratings: List[int]) -> int:
        #贪心算法 分两个路线进行遍历

        candy = [1] * len(ratings) #初始化candy数组，每个孩子最少分得一块糖果
        for i in range(1,len(ratings)): #先考虑右孩子比左孩子评分高的情况 从前向后遍历
            if ratings[i] > ratings[i-1]:
                candy[i] = candy[i-1] + 1
        for j in range(len(ratings) - 2,-1,-1): #先考虑左孩子比右孩子评分高的情况 从后
        向前遍历
            if ratings[j] > ratings[j+1]:
                candy[j] = max(candy[j],candy[j+1] + 1)
        return sum(candy)
```

思考

本题采用了两次贪心的策略：

- 一次是从左到右遍历，只比较右边孩子评分比左边大的情况。
- 一次是从右到左遍历，只比较左边孩子评分比右边大的情况。

局部最优推出了全局最优