两地调度

> 计算学部十大打卡活动——“龙舞编程新春会”编程打卡（2024-2-3）
>
> 力扣1029.两地调度

一、题目

公司计划面试 2n 人。给你一个数组 `costs` ，其中 `costs[i] = [aCosti, bCosti]` 。第 `i` 人飞往 `a` 市的费用为 `aCosti` ，飞往 `b` 市的费用为 `bCosti` 。

返回将每个人都飞到 `a` 、`b` 中某座城市的最低费用，要求每个城市都有 `n` 人抵达。

示例 1：

> **输入：** costs = [[10,20],[30,200],[400,50],[30,20]]
>
> **输出：** 110
>
> **解释：**
>
> > 第一个人去 a 市，费用为 10。
> > 第二个人去 a 市，费用为 30。
> > 第三个人去 b 市，费用为 50。
> > 第四个人去 b 市，费用为 20。
> >
> > 最低总费用为 10 + 30 + 50 + 20 = 110，每个城市都有一半的人在面试。

示例 2：

> **输入：** costs = [[259,770],[448,54],[926,667],[184,139],[840,118],[577,469]]
>
> **输出：** 1859

示例 3：

> **输入：** costs = [[515,563],[451,713],[537,709],[343,819],[855,779],[457,60],[650,359],[631,42]]
>
> **输出：** 3086

提示：

- `2 * n == costs.length`
- `2 <= costs.length <= 100`
- `costs.length` 为偶数
- `1 <= aCosti, bCosti <= 1000`

二、题解

思路——贪心

我们可以将问题转化，公司首先将这 $2N$ 个人全都安排飞往 B 市，再选出 N 个人改变它们的行程，让他们飞往 A 市。

如果选择改变一个人的行程，那么公司将会额外付出 `price_A - price_B` 的费用，这个费用可正可负。

因此最优的方案是，选出 `price_A - price_B` 最小的 N 个人，让他们飞往 A 市，其余人飞往 B 市。

代码

```
class Solution {
public:
    int twoCitySchedCost(vector<vector<int>>& costs) {
        sort(begin(costs), end(costs),
                []const vector<int> &v1, const vector<int> &v2) {
            return (v1[0] - v1[1] < v2[0] - v2[1]);
        });

        int ans=0;
        int n=costs.size()/2;
        int i;
        for(i=0;i<n;i++){
            ans+=costs[i][0]+costs[i+n][1];
        }
        return ans;
    }
};
```

复杂度分析

- 时间复杂度：$O(NlogN)$，需要对 `price_A - price_B` 进行排序。
- 空间复杂度：$O(1)$。