

1029. Two City Scheduling

25 March 2022 04:27 PM

A company is planning to interview $2n$ people. Given the array `costs` where `costs[i] = [aCosti, bCosti]`, the cost of flying the i^{th} person to city `a` is `aCosti`, and the cost of flying the i^{th} person to city `b` is `bCosti`.

Return the minimum cost to fly every person to a city such that exactly n people arrive in each city.

Example 1:

Input: `costs = [[10,20],[30,200],[400,50],[30,20]]`

Output: 110

Explanation:

The first person goes to city A for a cost of 10.

The second person goes to city A for a cost of 30.

The third person goes to city B for a cost of 50.

The fourth person goes to city B for a cost of 20.

The total minimum cost is $10 + 30 + 50 + 20 = 110$ to have half the people interviewing in each city.

Example 2:

Input: `costs = [[259,770],[448,54],[926,667],[184,139],[840,118],[577,469]]`

Output: 1859

Example 3:

Input: `costs = [[515,563],[451,713],[537,709],[343,819],[855,779],[457,60],[650,359],[631,42]]`

Output: 3086

$[[10, 20], [30, 200], [400, 50], [30, 20]]$

Output $\Rightarrow 10 + 30 + 50 + 20 = 110$

Understand the question

- * $2N$ folks
- * Divide it into 2 groups visiting city A vs city B
- * Both should have N folks
- * minimum cost to fly every person that exactly N people arrive in each city.
 \rightarrow greedy

cost A	B	loss Visiting city B over City A city A - city B	
[10, 20]		-10	-ve \Rightarrow it's significant that you will get benefited if you visit city A rather B
[30, 200]		-170	
[400, 50]		350	+ve \Rightarrow Here it's significant that you will get this much gain instead of loss if you visit B rather than A
[30, 20]		10	

we sort now, (city A - city B)

[30, 200] -170
[10, 20] -10
[30, 20] 10
[400, 50] 350

-ve \Rightarrow A +ve \Rightarrow B

cost = ~~30~~ 40 + 60 + 10

Sort the array in the basis of loss of visiting by cost A - cost B

```

class Solution {
    public int twoCitySchedCost(int[][] costs) {
        // sort them in basis of loss of city a and city b
        // a[0], a[1] -- price for city A, and B for first candidate
        // b[0], b[1] -- price for city A and B for 2nd candidate
        // a[0] - a[1] = cost saving on sending first candidate to city A instead of B
        // b[0] - b[1] = cost saving on sending 2nd candidate to city A instead of city B
        Arrays.sort(costs, (a,b) -> {
            return (a[0] - a[1]) - (b[0] - b[1]);
        });

        int total = 0;
        for(int i = 0; i < costs.length; i++){
            if(i < costs.length / 2){
                // select city A
                total += costs[i][0];
            } else {
                // select city B
                total += costs[i][1];
            }
        }
        return total;
    }
}

```

\Rightarrow first half after sorting will be city A

\Rightarrow next half will be city B

$$T.C \Rightarrow O(n \log n)$$

$$S.C \Rightarrow O(1)$$