Problem 2412: Minimum Money Required Before Transactions

Problem Information

Difficulty: Hard
Acceptance Rate: 41.92%
Paid Only: No
Tags: Array, Greedy, Sorting

Problem Description

You are given a **0-indexed** 2D integer array `transactions`, where `transactions[i] = [costi, cashbacki]`.

The array describes transactions, where each transaction must be completed exactly once in **some order**. At any given moment, you have a certain amount of `money`. In order to complete transaction `i`, `money >= costi` must hold true. After performing a transaction, `money` becomes `money - costi + cashbacki`.

Return _the minimum amount of_`money` _required before any transaction so that all of the transactions can be completed**regardless of the order** of the transactions._

Example 1:

Input: transactions = [[2,1],[5,0],[4,2]] **Output:** 10 **Explanation:** Starting with money = 10, the transactions can be performed in any order. It can be shown that starting with money < 10 will fail to complete all transactions in some order.

Example 2:

Input: transactions = [[3,0],[0,3]] **Output:** 3 **Explanation:** - If transactions are in the order [[3,0],[0,3]], the minimum money required to complete the transactions is 3. - If transactions are in the order [[0,3],[3,0]], the minimum money required to complete the transactions is 0. Thus, starting with money = 3, the transactions can be performed in any order.

Constraints:

* `1 <= transactions.length <= 105` * `transactions[i].length == 2` * `0 <= costi, cashbacki <= 109`

Code Snippets

C++:

```
class Solution {
public:
long long minimumMoney(vector<vector<int>>& transactions) {

}
};
```

Java:

```
class Solution {
public long minimumMoney(int[][] transactions) {

}
}
```

Python3:

```
class Solution:
def minimumMoney(self, transactions: List[List[int]]) -> int:
```