Problem 2861: Maximum Number of Alloys

Problem Information

Difficulty: Medium
Acceptance Rate: 39.91%
Paid Only: No
Tags: Array, Binary Search

Problem Description

You are the owner of a company that creates alloys using various types of metals. There are \`n\` different types of metals available, and you have access to \`k\` machines that can be used to create alloys. Each machine requires a specific amount of each metal type to create an alloy.

For the \`ith\` machine to create an alloy, it needs \`composition[i][j]\` units of metal of type \`j\`. Initially, you have \`stock[i]\` units of metal type \`i\`, and purchasing one unit of metal type \`i\` costs \`cost[i]\` coins.

Given integers \`n\`, \`k\`, \`budget\`, a **1-indexed** 2D array \`composition\`, and **1-indexed** arrays \`stock\` and \`cost\`, your goal is to **maximize** the number of alloys the company can create while staying within the budget of \`budget\` coins.

All alloys must be created with the same machine.

Return _the maximum number of alloys that the company can create_.

Example 1:

Input: n = 3, k = 2, budget = 15, composition = [[1,1,1],[1,1,10]], stock = [0,0,0], cost = [1,2,3] **Output:** 2 **Explanation:** It is optimal to use the 1st machine to create alloys. To create 2 alloys we need to buy the: - 2 units of metal of the 1st type. - 2 units of metal of the 2nd type. - 2 units of metal of the 3rd type. In total, we need 2 * 1 + 2 * 2 + 2 * 3 = 12 coins, which is smaller than or equal to budget = 15. Notice that we have 0 units of metal of each type and we have to buy all the required units of metal. It can be proven that we can create at most 2 alloys.

Example 2:

Input: n = 3, k = 2, budget = 15, composition = [[1,1,1],[1,1,10]], stock = [0,0,100], cost = [1,2,3] **Output:** 5 **Explanation:** It is optimal to use the 2nd machine to create alloys. To create 5 alloys we need to buy: - 5 units of metal of the 1st type. - 5 units of metal of the 2nd type. - 0 units of metal of the 3rd type. In total, we need 5 * 1 + 5 * 2 + 0 * 3 = 15 coins, which is smaller than or equal to budget = 15. It can be proven that we can create at most 5 alloys.

Example 3:

Input: n = 2, k = 3, budget = 10, composition = [[2,1],[1,2],[1,1]], stock = [1,1], cost = [5,5] **Output:** 2 **Explanation:** It is optimal to use the 3rd machine to create alloys. To create 2 alloys we need to buy the: - 1 unit of metal of the 1st type. - 1 unit of metal of the 2nd type. In total, we need 1 * 5 + 1 * 5 = 10 coins, which is smaller than or equal to budget = 10. It can be proven that we can create at most 2 alloys.

Constraints:

* `1 <= n, k <= 100` * `0 <= budget <= 108` * `composition.length == k` * `composition[i].length == n` * `1 <= composition[i][j] <= 100` * `stock.length == cost.length == n` * `0 <= stock[i] <= 108` * `1 <= cost[i] <= 100`

Code Snippets

C++:

```
class Solution {
public:
int maxNumberOfAlloys(int n, int k, int budget, vector<vector<int>>&
composition, vector<int>& stock, vector<int>& cost) {

}
};
```

Java:

```
class Solution {
public int maxNumberOfAlloys(int n, int k, int budget, List<List<Integer>>
composition, List<Integer> stock, List<Integer> cost) {
}
}
```

Python3:

```
class Solution:
def maxNumberOfAlloys(self, n: int, k: int, budget: int, composition:
List[List[int]], stock: List[int], cost: List[int]) -> int:
```