Algorithms Design

Project 2 - Dynamic Programing

4012 - Semester

The Legendary Dungeon of Byteburg!

Overview

In the enchanted land of Byteburg, a fabled dungeon has emerged, rumored to be brimming with unimaginable treasures. Our task is to guide Sir Jaime as he explores the dungeon, collecting treasures along the way. However, some treasures are more valuable than others, and Sir Jaime must devise a strategy to **collect at least m gold coins** while **minimizing encounters with dangerous creatures**.
The dungeon consists of multiple interconnected rooms, each containing a treasure with an assigned value. **Some treasures are more valuable than others**, indicating their worth in gold coins.
Also countering creatures came up with the following scoring system — we assigned an integer to each counter:

- $b_i = 1 \longrightarrow$ regular creature.
- $b_i = 2 \longrightarrow$ monster creature.

So, according to this rating system, he countering $b_1 + b_2 + \ldots + b_n$ points.
For example, if n = 5, m = 7 and gold coins are a = [5, 3, 2, 1, 4], b = [2, 1, 1, 2, 1], then Sir Jaime can achieve the following treasure sets (not all options are listed below):

- Room with numbers 1, 4 and 5. In this case, he will collect $a_1 + a_4 + a_5 = 10$ gold coins and lose $b_1 + b_4 + b_5 = 5$ points.
- Room with numbers 1 and 3. In this case, he will collect $a_1 + a_3 = 7$ gold coins and lose $b_1 + b_3 = 3$ points.
- Room with numbers 2 and 5. In this case, he will collect $a_2 + a_5 = 7$ gold coins and lose $b_2 + b_5 = 2$ points.

To conquer the Legendary Dungeon of Byteburg, we'll employ a dynamic programming approach. By breaking down the dungeon exploration into subproblems and utilizing memoization, we can determine the optimal path that maximizes the total value of treasures collected while minimizing encounters with dangerous creatures.

Input

The first line of each test case contains two integers n and m the number of rooms in dungeon and the number of coins to be collected. $(1 \leq n \leq 2 \times 10^5, \;\; 1 \leq m \leq 10^9)$

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ the value of each treasure.

The third line of each test case contains n integers $b_1, b_2, \ldots, b_n$ the convenience points of each counter. $(1 \leq b_i \leq 2)$

Output

For each test case:

- -1, if there is no set of rooms, exploring which will collect at least m coins.
- The minimum number of convenience points that Sir Jaime will lose if such a set exists.

Examples

Input:

```
5 7
5 3 1 2 4
2 1 1 2 1
```

Output:

2

because it is optimal to explore rooms with numbers 2 and 5, collecting 7 gold coins. $b_2 + b_5 = 2$

Input:

```
1 3
2
1
```

Output:

-1

because by exploring the only room, Sir Jaime will be able to collect only 2 gold coins. coins out of the 3 needed.

Goal

1. **Design Dynamic Programming Algorithm that can help Sir Jaime for obtaining maximum treasures.**

Attention!

1. All of the projects will be tested for similarities by a coded script, So if we find an obvious similarity between 2 or more projects **all** of them will get **-100** points.
2. You are free to choose any programming language you desire in this problem.
3. You have to write a document file and **explain deeply** what you've done in your code and the algorithm you wrote, Explain the algorithm **before** you begin talking about the code.
4. It's necessary to present your implementation. There is no score for just uploading the documentation.