Problem 2548: Maximum Price to Fill a Bag

Problem Information

Difficulty: Medium
Acceptance Rate: 64.00%
Paid Only: Yes
Tags: Array, Greedy, Sorting

Problem Description

You are given a 2D integer array `items` where `items[i] = [pricei, weighti]` denotes the price and weight of the `ith` item, respectively.

You are also given a **positive** integer `capacity`.

Each item can be divided into two items with ratios `part1` and `part2`, where `part1 + part2 == 1`.

* The weight of the first item is `weighti * part1` and the price of the first item is `pricei * part1`.
* Similarly, the weight of the second item is `weighti * part2` and the price of the second item is `pricei * part2`.

Return _**the maximum total price** to fill a bag of capacity_ `capacity` _with given items_. If it is impossible to fill a bag return `-1`. Answers within `10-5` of the **actual answer** will be considered accepted.

Example 1:

Input: items = [[50,1],[10,8]], capacity = 5 **Output:** 55.00000 **Explanation:** We divide the 2nd item into two parts with part1 = 0.5 and part2 = 0.5. The price and weight of the 1st item are 5, 4. And similarly, the price and the weight of the 2nd item are 5, 4. The array items after operation becomes [[50,1],[5,4],[5,4]]. To fill a bag with capacity 5 we take the 1st element with a price of 50 and the 2nd element with a price of 5. It can be proved that 55.0 is the maximum total price that we can achieve.

Example 2:

Input: items = [[100,30]], capacity = 50 **Output:** -1.00000 **Explanation:** It is impossible to fill a bag with the given item.

Constraints:

* `1 <= items.length <= 105` * `items[i].length == 2` * `1 <= pricei, weighti <= 104` * `1 <= capacity <= 109`

Code Snippets

C++:

```cpp
class Solution {
public:
double maxPrice(vector<vector<int>>& items, int capacity) {


}
};
```

Java:

```java
class Solution {
public double maxPrice(int[][] items, int capacity) {


}
}
```

Python3:

```python
class Solution:
def maxPrice(self, items: List[List[int]], capacity: int) -> float:
```