DESAFIO: Otimizando Valor no Armazém

Forma de entrega: link do código fonte salvo no Gist do Github
Linguagens aceitas: Javascript, Java, C#, Python

A empresa de logística, Armazenamento Eficiente S.A., está enfrentando desafios com a otimização do espaço em seu novo armazém. O armazém, localizado estrategicamente perto de um grande centro comercial, tem capacidade para armazenar até C metros cúbicos de mercadorias.

Você, como especialista em logística e otimização, foi chamado para resolver esse desafio. A empresa trabalha com uma variedade de fornecedores que oferecem produtos variados, cada um com um preço de venda específico e um tamanho definido. Para manter a eficiência e rentabilidade da operação, você precisa escolher cuidadosamente quais produtos devem ser armazenados no armazém.

Para tal, você recebeu dados dos N produtos disponíveis. Cada produto i tem um preço de venda prices[i] e ocupa volume[i] metros cúbicos de espaço no armazém.

Selecione os produtos de modo a maximizar o valor total armazenado no armazém, sem exceder sua capacidade máxima C.

Regras

Você não pode armazenar frações de produtos; ou um produto inteiro é armazenado, ou não é.
Só se pode comprar uma unidade de cada produto listado.
A soma dos tamanhos dos produtos armazenados não pode exceder a capacidade total do armazém C.
Não é preciso considerar as dimensões dos produtos, apenas o volume ocupado.

Entrada

- Um número inteiro C representando a capacidade total do armazém (em metros cúbicos).
- Um número inteiro N representando o número de produtos disponíveis.
- Um array de inteiros prices de tamanho N, onde prices[i] é o preço de venda do produto i.
- Um array de inteiros volume[i] de tamanho N, onde volume[i] é o tamanho do produto i (em metros cúbicos).

Saída

Um número inteiro representando o **valor máximo** dos produtos que podem ser armazenados no armazém sem exceder sua capacidade.

Entrada 1	**Saída 1**

`{`
`  "C": 10,`
`  "N": 4,`
`  "prices": [5, 12, 8, 1],`
`  "volume": [4, 8, 5, 3]`
`}`
	`13`

Entrada 2	Saída 2

{
  "C": 10,
  "N": 4,
  "prices": [5, 15, 8, 1],
  "volume": [4, 8, 5, 3]
}
	15

Entrada 3	Saída 3

{
  "C": 4,
  "N": 3,
  "prices": [1, 2, 3],
  "volume": [4, 5, 1]
}
	3

Entrada 4	Saída 4

{
  "C": 3,
  "N": 3,
  "prices": [1, 2, 3],
  "volume": [4, 5, 6]
}
	0

Entrada 5	Saída 5

{
  "C": 0,
  "N": 3,
  "prices": [20, 30, 40],
  "volume": [10, 20, 30]
}
	0

Entrada 6	Saída 6

{
  "C": 100,
  "N": 0,
  "prices": [],
  "volume": []
}
	0

Entrada 7	Saída 7
(favor pegar os dados no link) https://gist.github.com/acenelio/c73090174fdf5634420711b3f846ec3a	78970

Assinaturas:

Javascript:
```
function maxWarehouseValue(C, N, prices, volume)
```

Java:
```
public int maxWarehouseValue(int C, int N, int[] prices, int[] volume)
```

C#:
```
public int MaxWarehouseValue(int C, int N, int[] prices, int[] volume)
```

Python:
```
def max_warehouse_value(C, N, prices, volume)
```