O problema da mochila fracionária

Também conhecido como *"problema fracionário da mochila"* ou *"problema da mochila contínua"*. Sua resolução é fácil, pode ser aplicada por um algoritmo guloso sendo muito eficiente.

Problemática: Imagine que há um número x de objetos que devemos colocar em uma mochila de capacidade C. Cada objeto tem peso P e um valor V. Ainda é possível escolher uma fração de cada objeto I, para colocar na mochila, de modo que respeite a capacidade da mochila e maximize o valor dos objetos dentro da mesma.

Problema da mochila fracionária: Dados vetores (P1, P2, … , Pn) , (V1, V2, … , Vn) , (X1, X2, … , Xn), uma variável C, é uma variável I. Deve-se buscar a maximização de X*V sob as restrições de $X*P \leq C$ e $0 \leq XI \leq 1$ para todo I. Pode se dizer que P é o vetor de peso de cada objeto, V é o vetor de valores, X é o vetor de quantidade do objeto, C é a capacidade da mochila e I trata-se do número de objetos. O valor ideal de uma mochila fracionária, compreende-se que seja X*V, que não ultrapasse C, mas que ocupe o valor máximo determinado.

P	Vetor de peso dos itens.
V	Vetor de valor dos itens.
C	Capacidade da mochila.
X	Quantidade de objetos.
I	Objetos.

Resolução através de um algoritmo guloso: É utilizado o algoritmo guloso, pois em cada interação ele se utiliza do objeto de maior valor para iniciar o preenchimento da mochila, sem se preocupar com o que pode acontecer depois. Esse método otimiza tempo, pois o algoritmo não se arrepende de um valor atribuído ao componente de seleção X, mesmo que não seja o melhor no final do processo.

O algoritmo guloso gera uma mochila viável, ou seja, preenche ela visando colocar objetos de maior valor específico, mas para isso exige que os dados estejam em ordem crescente, com seus valores determinados para valor e peso.

Ex..: V1/P1 <= V2/P2 <= ... <= Vn/Pn

Descrição do problema em Pseudocódigo: (P, V, n, C)

```
enquanto n >= 1 e Pn <= C faça
        Xn ← 1
        C ← C – Pn
        n ← n – 1

se n >= 1 então
        Xn ← C / Pn
        para I ← n-1 decrescendo até 1 faça
        XI ← 0

devolva X
```

Desempenho: O consumo de tempo do algoritmo é de O(n), sendo o consumo de tempo de mesma ordem do tempo gasto com a leitura dos dados, sendo um algoritmo eficiente e rápido. É estimado que o algoritmo consuma $\Omega(n)$ unidades de tempo, mesmo no seu melhor caso.