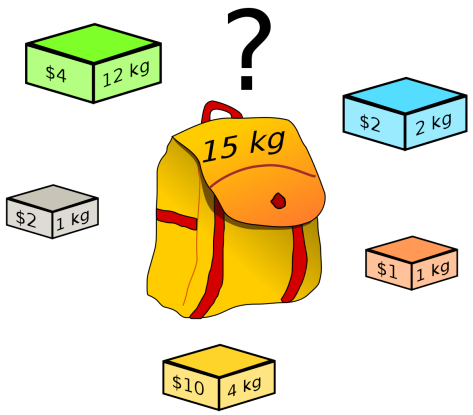
Projeto 2 – Problema da Mochila

O problema da mochila é um problema clássico de otimização com muitas aplicações: planos de corte de materiais, seleção de investimentos e portfólios, geração de chaves de criptografia, etc. Neste projeto, vamos implementar uma ferramenta para construir e analisar soluções para o problema da mochila, implementando algoritmos básicos usados na prática.



O problema da mochila pode ser formalizado da seguinte maneira: temos uma mochila com capacidade P e um conjunto I de n itens; cada item $i \in I$ tem um valor v_i e um peso p_i associados. O objetivo é selecionar um subconjunto $J \subseteq I$ de itens para levar na mochila, de modo que o valor total dos itens selecionados seja o maior possível (i.e. max $\sum_{j \in J} v_j$), e o peso total desses itens não ultrapasse a capacidade da mochila (i.e. $\sum_{j \in J} p_j \leq P$).

Consideremos o exemplo a seguir. Temos uma instância do problema da mochila com $n = 6$ itens $I = \{A, B, C, D, E, F\}$. A capacidade da mochila é $P = 14$. Os valores e pesos de cada item são dados pela tabela abaixo.

Item	Valor	Peso
A	8	6
B	3	5
C	1	3
D	6	6
E	9	8
F	5	4

Uma solução para o problema da mochila é representada pelo conjunto de itens selecionados. Uma possível solução é $S_1 = \{A, C, F\}$, com valor total de 14 e peso total de 13. A solução é viável pois o peso total é menor que a capacidade da mochila. A solução $S_2 = \{A, E\}$ é melhor que a anterior, pois seu valor total é 17. Essa solução também é válida, pois seu peso total é 14, que é menor ou igual que a capacidade da mochila. Já a solução $S_3 = \{A, E, F\}$ não é viável, pois seu peso total de 18 ultrapassa a capacidade da mochila.

A exemplo da instância discutida acima, neste projeto vamos considerar somente instâncias do problema da mochila com $n = 6$ itens, valores e pesos $v, p \in [1, 9]$, e capacidade $P \in [10, 99]$. Todos os valores são inteiros. Uma instância do problema da mochila será representada por uma string contendo o valor e peso (separados por vírgula) de cada um dos seis itens (os quais são separados por ponto e vírgula). Ao final é dada a capacidade da mochila. A instância apresentada acima é representada pela string "8,6;3,5;1,3;6,6;9,8;5,4;14".

1 Funcionalidades esperadas

O programa deve ler a instância como um argumento na chamada e apresentar o seguinte menu:

1. Avaliar solução
2. Resolver via heurística: maior valor
3. Resolver via heurística: menor peso
4. Resolver via heurística: melhor razão
5. Resolver via busca exaustiva
6. Sair

Na **opção 1**, o usuário deverá informar uma solução contendo os itens selecionados (e.g. "ACF" ou "ABD"), e o programa avalia e informa sua viabilidade (se satisfaz a restrição de capacidade) e seu valor total. Nas **opções 2 a 4**, o programa deverá construir uma solução usando as heurísticas descritas abaixo. Na **opção 5**, o programa deverá informar a melhor solução via busca exaustiva. Ou seja, ele deverá gerar todas as soluções possíveis e identificar a melhor delas. Finalmente, a opção 6 é usada para encerrar a execução.

Heurística do maior valor: a heurística inicia com uma solução vazia (nenhum item selecionado) e avalia um item por vez. Caso a adição do item não excede a capacidade da mochila, ele é adicionado à solução. A ordem na qual os itens são avaliados (e possivelmente adicionados à solução) considera seus valores, do item mais valioso para o menos valioso.

Heurística do menor peso: segue a mesma estratégia da heurística anterior, mas ordena os itens pelos seus pesos, do menor para o maior.

Heurística da melhor razão: segue a mesma estratégia da heurística anterior, mas ordena os itens pela razão entre o valor e o peso (i.e. v/p), da maior para a menor.

Um desafio: após desenvolvida a ferramenta, gere 100 instâncias do problema da mochila aleatórias e avalie a qualidade das soluções produzidas pelas três heurísticas. Para isso, considere o desvio médio relativo da melhor solução (obtida via busca exaustiva). Qual heurística é melhor? Faça as modificações necessárias para que essa bateria de experimentos não exija a intervenção do usuário.

Uma observação: em geral, o problema da mochila é resolvido por algoritmos de programação dinâmica, que encontram a melhor solução para instâncias com muitos itens de maneira eficiente (complexidade pseudo-polinomial). Veja mais em https://www.geeksforgeeks.org/0-1-knapsack-problem-dp-10.

2 Etapas de desenvolvimento

1. Fazer a leitura da entrada;
2. Definir variáveis para armazenar os dados e extraí-los a partir da entrada lida;
3. Calcular os valores para ordenação nas heurísticas;
4. Implementar o menu do programa;
5. Implementar a avaliação de soluções;
6. Desenvolver a heurística do maior valor;

7. Desenvolver a heurística do menor peso;
8. Desenvolver a heurística da melhor razão;
9. Implementar a busca exaustiva.