LAED1 - Trabalho Prático 1
Problema da Mochila
Ramon Griffo Costa
201622040112

Para resolver o problema da mochila, implementei primeiramente o algoritmo guloso, calculei o valor da razão do valor pelo peso de cada item e usei o algoritmo de ordenação por seleção para ordenar os itens em ordem crescente de custo benefício, dessa forma eu pude pegar os últimos itens do vetor, que tinham o maior custo benefício e colocá-los na mochila até que o peso não suportasse mais, como a parte com maior complexidade desse algoritmo foi a ordenação por seleção, a ordem de complexidade dele é $O(n^2)$, que é a ordem de complexidade para qualquer caso do algoritmo de seleção. Como o algoritmo guloso não cobre todas as instâncias ele não chega a uma solução ótima.
Os arquivos de entrada foram gerados dentro dos padrões pedidos usando o gerador_de_entradas.c implementado pelo Bruno Damacena, também aluno de LAED, disponibilizado em seu github*, gerei 8 entradas diferentes com tamanhos de mochila e números de produtos variados.
O arquivo de saída gerado é bem claro, ele tem descrições dos itens colocados na mochila, com o número, peso e valor do item, e no final ele diz o peso total colocado e o valor total arrecadado.
Para fazer o algoritmo de tentativa e erro eu pensei numa solução backtracking, onde ele colocaria itens na mochila até não caber mais, e para cada item colocado ele salvaria a combinação atual e verificaria se ela é a com maior valor até o momento, se ela for, ele salva num vetor melhor combinação e salva num inteiro o valor da maior combinação, e quando não coubesse mais nada na mochila ele removeria o último item e tentaria outro, repetidamente até não ter mais itens, e retiraria o anterior, e faria isso até cobrir todas as possibilidades, tal algoritmo tem ordem de complexidade exponencial, por sempre testar as instâncias existentes, e pelo mesmo motivo ele sempre chega a uma solução ótima.
Inicialmente tentei implementar o backtracking usando lastros de repetição, depois pensei em fazer usando recursividade, me parecia mais simples, porém depois pensei em implementar o algoritmo tentativa e erro usando outra técnica, utilizei de um vetor auxiliar para armazenar zeros e uns, q representam basicamente itens que entram (1) ou não (0) na mochila, e dessa forma eu uso a soma binária no vetor para percorrer todos os casos possíveis testando se o peso não estourou e se o valor é maior que o valor máximo já encontrado, dessa forma eu percorri todas as possibilidades e encontrei a solução ótima em qualquer caso, o problema é que como a ordem de complexidade é extremamente grande, para valores minimamente grandes o tempo de execução tende ao infinito.

*https://github.com/BrunoDamacena/OProblemaDaMochila/tree/master/gerador_de_entrada