Trabalho Final - Complexidade de Algoritmos

Arthur Lodetti Gonçalves, Gustavo Piacentini da Silva, Rafael Eduardo Gonçalves

Centro de Ciências Tecnológicas – Universidade do Estado de Santa Catarina (UDESC)
Joinville – SC – Brasil
lodettiarthur@gmai.com, rafaeleduardo.g@hotmail.com,
gustavopiacentinisilva@gmail.com

1. Introdução

Este trabalho final, da matéria de complexidade de algoritmos ministrada por Leandro Israel Pinto, busca demonstrar uma possível solução heurística para um problema NP-difícil - sem solução em tempo polinomial - aplicado a alguma situação do mundo real. Dentro deste relatório iremos contextualizar o problema escolhido, descrever a técnica implementada, assim como analisar sua eficiência e expor os resultados dos testes realizados.

2. Definição do Problema

O tema que escolhemos se baseia no conceito do "Problema da Mochila", nele temos uma mochila com capacidade C e uma lista de itens que tem peso e valor concretos, o objetivo é preencher a mochila buscado maximizar o somatório dos valores dentro dela sem que a soma dos pesos desses itens ultrapasse sua capacidade.

A situação que buscamos simular é de um indivíduo que quer arrumar sua mala para uma viagem e busca colocar o máximo de itens que ele considera valiosos dentro dela. Por isso podemos encaixar a situação em um "Problema da Mochila 0/1", já que não podemos adicionar mais de um mesmo item ou fracionar um dos itens.

Entretanto alguns dos objetos são frágeis e podem não suportar o peso dos outros itens se forem colocados próximos ao fundo da mala. Portanto é necessário levar em consideração o quão resistente cada item é e posicioná-los de acordo.

3. Solução Proposta

Cada item possui um peso e um valor próprio, e pode-se utilizar esses valores para calcular suas eficiências:

$$eficiência = \frac{valor}{peso}$$

Essa medida é utilizada para organizar a lista de itens, que em seguida é analisada pelo algoritmo. Como buscamos sempre tomar a melhor decisão localmente - inserir o item de maior eficiência - fica claro que se trata de um algoritmo guloso (Greedy algorithm).

Porém, neste trabalho, queremos levar em consideração o quão frágil cada item é, e assim adicionamos uma medida de fragilidade em cada um deles. Tal medida representa quanto peso o item é capaz de aguentar sem quebrar ou apresentar danos.

E, para que o algoritmo continue funcionando, mudamos o cálculo da eficiência:

$$eficiência = \frac{valor}{peso + \frac{beta}{fragilidade + 1}}$$

Sendo que beta é um coeficiente limitante da influência da fragilidade, pois quando ultrapassa o restante da capacidade disponível, essa passa a ser irrelevante para o cálculo da eficiência.

O algoritmo completo então organiza a lista baseada na eficiência dos itens e passa a inseri-los em ordem decrescente. Essa inserção é feita de cima para baixo na mala, ou seja, começa adicionando o item que ficará ao topo da mala na lista, verificando se a capacidade não será ultrapassada, e se o próximo item tem fragilidade grande o suficiente para suportar os itens acima. Essas verificações são realizadas até que a mala fique cheia.

4. Testes Realizados e Resultados

Para compreender o funcionamento do algoritmo e seu esforço, foram realizados testes com diferentes tamanhos de entrada para verificar como o mesmo se comporta com diferentes cargas de entrada.

Tabela 1 .Tamanho da Entrada e Diferentes Betas e seus Tempos de Execução(Em segundos)

Beta/Entrada	1000	10000	50000	100000	500000	1000000	2000000	4000000
0	0.000471	0.004686	0.022741	0.059835	0.319948	0.612357	1.246842	2.579997
10	0.000437	0.004272	0.027212	0.061143	0.358694	0.671249	1.414565	2.912276
20	0.000409	0.004225	0.026072	0.061371	0.326215	0.683100	1.433710	2.940960
50	0.000346	0.003979	0.027164	0.066281	0.326579	0.690283	1.429298	2.979223
100	0.000362	0.003732	0.026978	0.062054	0.328078	0.683445	1.438199	2.978723

Os valores em apresentados foram gerados com um código Python auxiliar que importou as principais funções do arquivo original.

Os testes foram feitos da seguinte forma, o código realiza testes de desempenho e eficácia sobre um algoritmo de mochila com itens frágeis, variando dois fatores principais: o número de itens e o parâmetro beta, que afeta a sensibilidade à fragilidade dos itens.

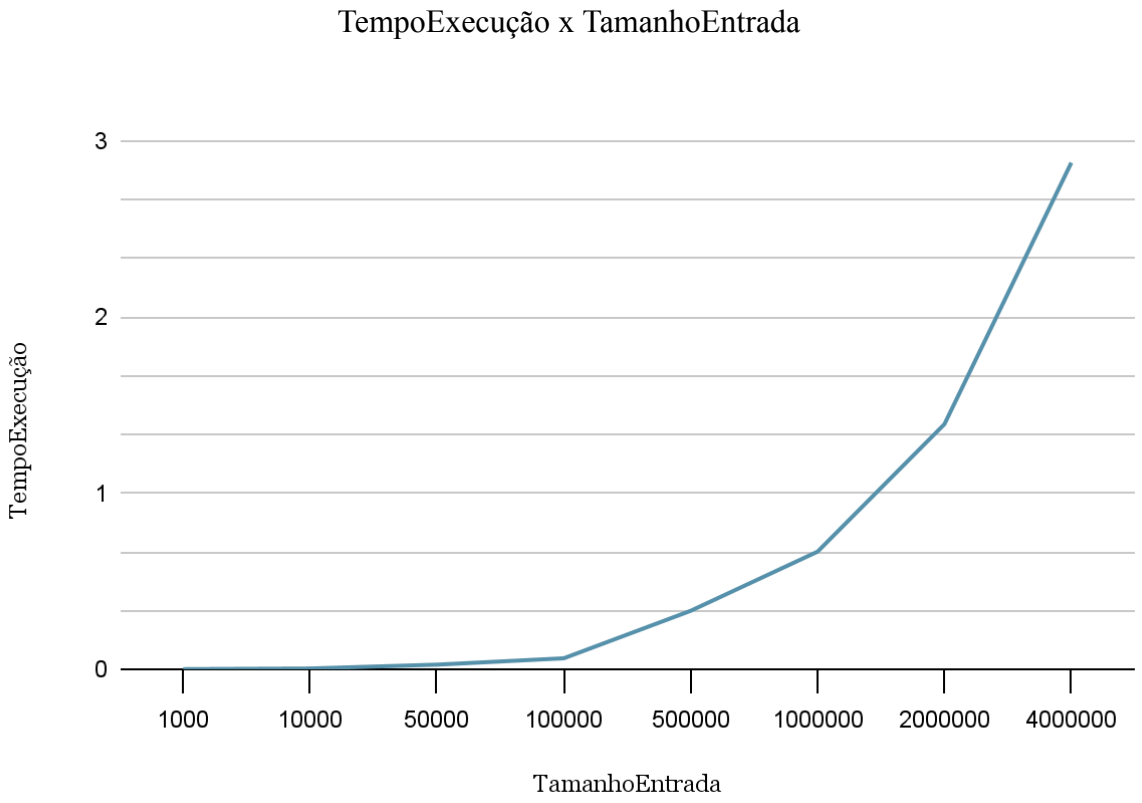

Figura 1. Gráfico de linha do tempo de execução

5. Conclusão

O algoritmo tratado no trabalho pode ser considerado como “Algoritmo guloso iterativo”, como trata a literatura de Cormen et al. [Cormen et al. 2022] **(ta certo isso?)**. Ele se enquadra pois a estratégia gulosa (greedy) é implementada para encontrar uma solução.

No entanto, algumas modificações foram feitas a partir dessa ideia tradicional. A questão de fragilidade, que indica o peso suportado por um objeto, além de um coeficiente limitante beta, que serve como limitador da influência da fragilidade sobre o cálculo, mostraram-se essenciais para a implementação da atividade. Uma vez que, com esses valores, obtemos resultados mais precisos sobre o conjunto de elementos, do que apenas usar o valor e peso como medidas. Então, verifica-se o elemento mais eficiente e coloca-o no topo, ordenando a mochila de cima para baixo. Dessa forma, os elementos inferiores são menos eficientes porém são capazes de suportar o peso acima, se tornando soluções para o problema.

Por fim, é possível analisar com base nos gráficos que o algoritmo tem um desempenho eficiente mesmo com grandes entradas de itens, processando o grande volume de dados em poucos segundos.

6. Referências

Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2022). Algoritmos: Teoria e Pratica ´ . GEN LTC, 4 edition.