



Problema da Mochila 0/1

Objetivo Geral

O objetivo deste trabalho prático é que o aluno desenvolva e implemente três abordagens clássicas para resolver o Problema da Mochila 0/1: **Força Bruta**, **Algoritmo Guloso** e **Programação Dinâmica**. Além de analisar teoricamente a complexidade das abordagens, o aluno deverá realizar uma análise empírica, medindo o tempo de execução para diferentes valores de n e comparando a eficiência de cada abordagem.

Descrição do Problema

O Problema da Mochila 0/1 é um problema clássico de otimização combinatória que pode ser descrito da seguinte forma:

Dado um conjunto de n itens, cada um com um peso w_i e um valor v_i , e uma mochila que pode suportar um peso máximo W , determine a combinação de itens que maximiza o valor total sem exceder a capacidade da mochila. Cada item pode ser incluído ou excluído da mochila na sua totalidade (não é permitido fracionar itens).

Versão 1: Força Bruta

O paradigma de **Força Bruta** é uma abordagem direta para resolver problemas de otimização, onde todas as combinações possíveis de soluções são geradas e avaliadas, e a melhor solução é escolhida. No contexto do Problema da Mochila 0/1, isso significa gerar todas as combinações possíveis de itens (incluindo ou excluindo cada item) e calcular o valor total de cada combinação, respeitando a restrição de capacidade da mochila. Esse algoritmo deve fornecer uma **solução ótima** para qualquer entrada.

Implementar a solução de força bruta em Linguagem C. Criar a equação de recorrência que descreve a solução (se aplicável) e analisar a complexidade temporal do algoritmo. Utilizar método de substituição ou árvore de recorrência para demonstrar a complexidade do algoritmo.

Versão 2: Algoritmo Guloso

O **Algoritmo Guloso** é uma abordagem que toma decisões localmente ótimas na esperança de encontrar a solução global ótima. No contexto da mochila, o algoritmo guloso tenta maximizar o valor total da mochila selecionando itens com base na razão valor/peso de cada item, de forma decrescente, até atingir a capacidade máxima. Essa abordagem não garante uma solução ótima para o Problema da Mochila 0/1, portanto, sua implementação pode fornecer uma **solução aproximada** para qualquer entrada.

Implementar o algoritmo guloso em Linguagem C. Analisar a solução e a sua complexidade temporal, explicando por que não garante a solução ótima no caso da mochila 0/1.

Versão 3: Programação Dinâmica

A **Programação Dinâmica** é uma técnica de otimização que resolve problemas complexos dividindo-os em subproblemas menores, armazenando os resultados dos subproblemas para evitar recomputação. No Problema da Mochila 0/1, a programação dinâmica constrói uma tabela que armazena o valor máximo possível para cada subproblema definido pelos primeiros i itens e uma capacidade w . Esse algoritmo deve fornecer uma **solução ótima** para qualquer entrada.

Implementar a solução usando programação dinâmica em Linguagem C. Criar a equação de recorrência que descreve a solução (se aplicável) e analisar a complexidade temporal do algoritmo. Demonstrar a complexidade do algoritmo.

Tarefas Específicas:

1. Implementação:

- Implemente as três abordagens descritas (Força Bruta, Algoritmo Guloso e Programação Dinâmica) na Linguagem C.
- As implementações devem ser feitas de forma modular, com funções separadas para cada abordagem.

2. Equações de Recorrência:

- Para cada abordagem, derive a equação de recorrência (se aplicável) que descreve o comportamento do algoritmo.
- Expresse claramente as condições base e as transições entre subproblemas.

3. Análise de Complexidade:

- Analise a complexidade temporal de cada algoritmo.

- Discuta as diferenças na eficiência de cada abordagem com base nas suas implementações e na teoria.
- 4. **Análise Empírica:**
 - Realize medições de tempo de execução para diferentes valores de n (número de itens) em ambas as abordagens.
 - Compare os tempos de execução medidos para a solução de força bruta, algoritmo guloso e para a solução com programação dinâmica.
 - A análise empírica deve incluir:
 - Um gráfico comparando os tempos de execução das três abordagens para diferentes valores de n .
 - Discussões sobre como os tempos de execução observados se relacionam com as complexidades teóricas.
- 5. **Relatório:**
 - Entregue um relatório documentando o trabalho realizado, incluindo:
 - As implementações em C.
 - As equações de recorrência derivadas.
 - A análise de complexidade temporal.
 - A análise empírica, com gráficos e discussões sobre os resultados obtidos.
 - Comparações entre as abordagens teórica e empírica.

Crerios de Avaliao:

- **Correção das Implementaes:** As implementaes devem estar corretas, gerando os resultados esperados para os testes fornecidos [3,0].
- **Equaes de Recorrncia:** As equaes devem ser corretamente derivadas e justificadas [1,5].
- **Anlise de Complexidade:** A anlise deve ser precisa, clara e deve refletir o comportamento real dos algoritmos [1,5].
- **Anlise Empírica:** A anlise empírica deve ser bem estruturada, com medies precisas e grficos claros que mostrem a comparao entre as abordagens [2,0].
- **Clareza e Organizao do Relatório:** O relatório deve ser bem estruturado, com explicaes claras e concisas [2,0].

Data de Entrega:

- O trabalho deve ser entregue at 09/10/2025 no Tidia.

Referncias:

- Material de aula sobre anlise de algoritmos.
- Livros recomendados:
 - "Introduction to Algorithms" de Cormen et al.
 - "Algorithm Design Manual" de Skiena.

Observaes Finais:

Este trabalho deve ser feito em duplas (no mximo) e visa aprofundar o entendimento de complexidade de algoritmos, alm de desenvolver habilidades prticas na anlise e implementao de algoritmos. A anlise empírica permitir aos alunos observar como as diferenas teóricas se manifestam em prticas reais, proporcionando uma compreenso mais profunda da eficiência algorítmica. Bom trabalho!