

Atividade PJBL – Somativa 4

Análise de Técnicas de Projeto de Algoritmos

Giovanna Caroline Zettel Francisco

Leonardo Augusto Dolvitsch

Matheus Paul Lopuch

<https://github.com/owMath/DesignTechniquesPJBL>

Introdução

Este relatório apresenta a implementação e análise de três diferentes técnicas de projeto de algoritmos, aplicadas a um contexto prático de um sistema de biblioteca. Cada técnica é demonstrada através de um problema específico, com suas respectivas implementações e análises de eficiência.

Escolha do Problema Computacional

Para demonstrar as três técnicas de projeto de algoritmos, foi escolhido o Sistema de Gerenciamento de Biblioteca como contexto prático.

Esta escolha foi baseada nos seguintes critérios:

1. Aplicabilidade Prática: É um problema real e comum, facilmente compreensível e com aplicações no dia a dia.
2. Diversidade de Operações: Permite demonstrar diferentes tipos de operações:
 - Ordenação de dados (Merge Sort)
 - Busca eficiente (Busca Binária)
 - Processamento de dados (Multiplicação de Matrizes)
3. Escalabilidade: O sistema pode lidar com diferentes tamanhos de dados, permitindo demonstrar a eficiência dos algoritmos em diferentes cenários.
4. Contexto Unificado: Todas as técnicas podem ser aplicadas em um mesmo contexto, facilitando a comparação entre elas.

Problemas Específicos Resolvidos

1. Ordenação de Livros (Merge Sort)

- Problema: Manter o catálogo de livros ordenado por número de identificação
- Aplicação: Facilita a localização física dos livros na biblioteca

2. Busca de Livros (Busca Binária)

- Problema: Encontrar rapidamente um livro específico no catálogo ordenado
- Aplicação: Atendimento rápido aos usuários da biblioteca

3. Análise de Empréstimos (Multiplicação de Matrizes)

- Problema: Calcular estatísticas de empréstimos por categoria
- Aplicação: Ajuda na gestão do acervo e planejamento de aquisições

Esta escolha permite demonstrar como diferentes técnicas de projeto podem ser aplicadas para resolver problemas distintos dentro de um mesmo contexto prático, facilitando a compreensão e comparação das abordagens.

Problema 1. Dividir e Conquistar: Merge Sort

Ordenação de uma lista de livros por número de identificação em um sistema de biblioteca.

Descrição da Técnica:

A técnica de Dividir e Conquistar consiste em:

1. Dividir o problema em subproblemas menores
2. Resolver os subproblemas recursivamente
3. Combinar as soluções dos subproblemas

Implementação:

O Merge Sort foi implementado seguindo estes passos:

1. Dividir a lista em duas metades
2. Ordenar recursivamente cada metade
3. Mesclar as duas metades ordenadas

Análise de Eficiência

- Complexidade Temporal: $O(n \log n)$
- Complexidade Espacial: $O(n)$
- Vantagens:
 - Estável
 - Performance consistente
 - Bom para grandes conjuntos de dados
- Desvantagens:
 - Requer espaço adicional
 - Não é in-place

Problema 2. Decrementar e Conquistar: Busca Binária

Busca eficiente de livros por número de identificação em uma lista ordenada.

Descrição da Técnica:

A técnica de Decrementar e Conquistar:

1. Reduz o tamanho do problema em cada iteração
2. Resolve o problema menor
3. Combina a solução com o resultado anterior

Implementação:

A Busca Binária foi implementada:

1. Dividindo o espaço de busca pela metade
2. Comparando o elemento do meio
3. Reduzindo o espaço de busca baseado na comparação

Análise de Eficiência

- Complexidade Temporal: $O(\log n)$
- Complexidade Espacial: $O(1)$
- Vantagens:
 - Extremamente eficiente

- Complexidade logarítmica
- Baixo uso de memória

- Desvantagens:

- Requer lista ordenada
- Não eficiente para listas pequenas

Problema 3. Transformar e Conquistar: Multiplicação de Matrizes

Processamento de dados de empréstimos de livros usando multiplicação de matrizes.

Descrição da Técnica:

A técnica de Transformar e Conquistar:

1. Transforma o problema em uma forma mais fácil de resolver
2. Resolve o problema transformado
3. Converte a solução de volta para o problema original

Implementação:

A Multiplicação de Matrizes foi implementada:

1. Transformando as matrizes em arrays NumPy
2. Utilizando operações vetorizadas
3. Aplicando o algoritmo de Strassen

Análise de Eficiência:

- Complexidade Temporal: $O(n^2.807)$
- Complexidade Espacial: $O(n^2)$

- Vantagens:

- Código limpo e legível
- Performance otimizada
- Operações vetorizadas

- Desvantagens:

- Requer biblioteca externa
- Overhead de memória

Comparação das Técnicas

Eficiência Temporal:

1. Busca Binária: $O(\log n)$
2. Merge Sort: $O(n \log n)$
3. Multiplicação de Matrizes: $O(n^2.807)$

Eficiência Espacial:

1. Busca Binária: $O(1)$
2. Merge Sort: $O(n)$
3. Multiplicação de Matrizes: $O(n^2)$

Aplicabilidade:

1. Merge Sort: Melhor para ordenação de grandes conjuntos de dados
2. Busca Binária: Ideal para busca em conjuntos ordenados
3. Multiplicação de Matrizes: Eficiente para processamento de dados em matriz

Validação dos Algoritmos:

Todos os algoritmos implementados foram validados através de testes automatizados, garantindo sua corretude e eficiência. Os testes incluem:

Merge Sort:

- Teste com lista vazia
- Teste com lista de um elemento
- Teste com lista já ordenada
- Teste com lista desordenada
- Teste com elementos repetidos

Busca Binária:

- Teste com elemento presente
- Teste com elemento não presente
- Teste com primeiro elemento
- Teste com último elemento
- Teste com lista vazia

Multiplicação de Matrizes:

- Teste com matrizes 2x2
- Teste com matrizes de dimensões diferentes
- Teste com matrizes incompatíveis

Os testes foram implementados usando o framework pytest e todos passaram com sucesso, confirmando que:

1. Os algoritmos funcionam corretamente para todos os casos de teste
2. As complexidades temporais e espaciais estão de acordo com o esperado
3. As implementações são robustas e tratam casos especiais adequadamente

Conclusão

Cada técnica de projeto de algoritmos tem suas particularidades e é mais adequada para certos tipos de problemas. A escolha da técnica deve ser baseada nas características do

problema e nos requisitos de eficiência. Neste projeto, demonstramos como diferentes técnicas podem ser aplicadas em um contexto prático de um sistema de biblioteca, cada uma com suas vantagens e desvantagens.