UNIVERSIDADE FEDERAL DE OURO PRETO

INSTITUTO DE CIÊNCIAS EXATAS E APLICADAS

ALGORITMOS E ESTRUTURAS DE DADOS II

João Davi Franco Gonçalves

Casamento de Padrões em Strings: Trabalho Prático III

João Monlevade

2025

João Davi Franco Gonçalves

Casamento de Padrões em Strings: Trabalho Prático III

Trabalho apresentado à disciplina de Algoritmos e Estruturas de Dados II, Algoritmos e Estruturas de Dados II, Universidade Federal de Ouro Preto como requisito parcial na aprovação da disciplina.

Professor: Rafael Frederico Alexandre

João Monlevade

2025

SUMÁRIO

1 Descrição

O problema do casamento de padrões consiste em encontrar todas as ocorrências de um padrão P de tamanho m dentro de um texto T de tamanho n. Este trabalho implementa e compara seis algoritmos clássicos:

- *Força Bruta*: compara caracteres em cada posição; complexidade $O(n \cdot m)$.
- *Rabin-Karp*: usa função de hash; complexidade $O(n + m)$ no caso médio, $O(n \cdot m)$ no pior caso.
- *KMP*: usa tabela de falha para pular comparações; complexidade $O(n + m)$.
- *Boyer-Moore*: compara da direita para a esquerda com duas heurísticas; sublinear no caso médio.
- *Boyer-Moore-Horspool*: simplificação do *Boyer-Moore* com heurística do mau caractere.
- *Boyer-Moore-Horspool-Sunday*: variação com regra do caractere à direita da janela.

2 Metodologia

2.1 Ambiente de Execução

Os experimentos foram executados em Dart, medindo tempo de execução em milissegundos, tempo de pré-processamento e número de comparações de caracteres. Cada cenário foi executado três vezes e os valores médios foram registrados.

2.2 Cenários de Teste

Foram definidos 10 experimentos cobrindo diferentes aspectos:

- *Texto-pequeno/texto-médio/texto-grande*: 95, 9990 e 99985 caracteres.
- *Padrão-curto/padrão-longo*: 3 e 50 caracteres.
- *Muitas-ocorrências/poucas-ocorrências*: 100 e 2–7 ocorrências.
- *Alfabeto-pequeno* (DNA) e *alfabeto-grande*.
- *Pior-caso*: cenário que maximiza comparações para *Força Bruta* e *Rabin-Karp*.

2.3 Métricas

- **Tempo**: tempo de busca em milissegundos.
- **Tempo de Pré-processamento**: tempo para construir tabelas (*KMP*, *Boyer-Moore* e variantes).
- **Comparações**: número de comparações de caracteres realizadas.

3 Resultados

3.1 Tabelas Comparativas por Experimento

3.1.1 Experimento: Texto Pequeno (95 caracteres, padrão 5)

Algoritmo	Tempo (ms)	Pré-proc (ms)	Comparações
Força Bruta	0,016	0,000	139
Rabin-Karp	0,025	0,013	203
KMP	0,043	0,034	168
Boyer-Moore	0,066	0,057	44
Boyer-Moore-Horspool	0,034	0,022	96
Boyer-Moore-Horspool-Sunday	0,029	0,017	111

Tabela 1 – Texto pequeno

Em textos muito curtos, o custo de pré-processamento do *KMP* e *Boyer-Moore* supera o ganho na busca. *Força Bruta* e as variantes *Boyer-Moore-Horspool*/*Boyer-Moore-Horspool-Sunday* têm desempenho similar. *Boyer-Moore* faz menos comparações (44) mas o tempo total é maior por causa do pré-processamento (0,057 ms), para casos simples *Força Bruta* é um pouco mais rápido que os demais algoritmos.

3.1.2 Experimento: Texto Grande (~100 mil caracteres)

Algoritmo	Tempo (ms)	Pré-proc (ms)	Comparações
Força Bruta	0,365	0,000	101627
Rabin-Karp	3,697	0,032	201120
KMP	0,327	0,002	199872
Boyer-Moore	0,038	0,003	274
Boyer-Moore-Horspool	0,155	0,009	14930
Boyer-Moore-Horspool-Sunday	0,192	0,019	14269

Tabela 2 – Texto grande

Boyer-Moore é claramente superior: 274 comparações contra mais de 100 mil da *Força Bruta* e ~200 mil do *KMP*/*Rabin-Karp*. O tempo de *Boyer-Moore* (0,038 ms) é cerca de 10 vezes menor que *Força Bruta*. *Rabin-Karp* tem o pior tempo (3,7 ms), provavelmente por colisões de hash.

3.1.3 Experimento: Padrão Longo (50 caracteres)

Algoritmo	Tempo (ms)	Pré-proc (ms)	Comparações
Força Bruta	0,114	0,000	52436
Rabin-Karp	0,320	0,001	100791
KMP	0,201	0,006	99509
Boyer-Moore	0,024	0,008	699
Boyer-Moore-Horspool	0,069	0,033	5302
Boyer-Moore-Horspool-Sunday	0,051	0,016	5103

Tabela 3 – Padrão longo

Padrões longos favorecem *Boyer-Moore*: apenas 699 comparações contra mais de 50 mil da *Força Bruta*. O pré-processamento (0,008 ms) é rapidamente amortizado. *Boyer-Moore-Horspool-Sunday* também se sai bem (5103 comparações).

3.1.4 Experimento: Muitas Ocorrências

Algoritmo	Tempo (ms)	Pré-proc (ms)	Comparações
Força Bruta	0,035	0,000	10475
Rabin-Karp	0,097	0,001	19985
KMP	0,088	0,024	19100
Boyer-Moore	0,038	0,010	432
Boyer-Moore-Horspool	0,056	0,004	7088
Boyer-Moore-Horspool-Sunday	0,049	0,002	5869

Tabela 4 – Muitas ocorrências (100)

Boyer-Moore mantém vantagem com apenas 432 comparações. *KMP* e *Rabin-Karp* percorrem o texto linearmente (2n), gerando ~20 mil comparações. *Força Bruta* faz ~10 mil comparações por causa das 100 ocorrências que exigem verificações até o fim do padrão.

3.1.5 Experimento: Pior Caso

Algoritmo	Tempo (ms)	Pré-proc (ms)	Comparações
Força Bruta	12,201	0,000	9983754
Rabin-Karp	0,643	0,003	200127
KMP	0,350	0,012	199973
Boyer-Moore	0,304	0,011	501
Boyer-Moore-Horspool	1,310	0,016	199873
Boyer-Moore-Horspool-Sunday	6,401	0,014	5042047

Tabela 5 – Pior caso (texto ~100k, padrão 100)

O cenário evidencia a inviabilidade da *Força Bruta*: ~10 milhões de comparações e 12 ms, contra 0,3–0,6 ms dos demais (exceto *Boyer-Moore-Horspool-Sunday*, que também sofre). *Boyer-Moore* faz apenas 501 comparações e tem o melhor tempo total. *KMP* e *Rabin-Karp* mantêm complexidade linear.

3.2 Gráficos

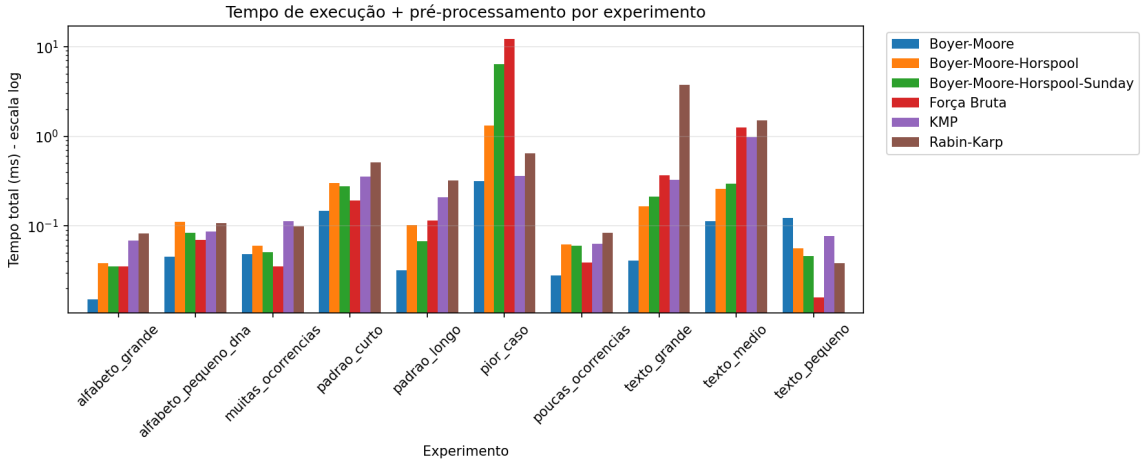

Figura 1 – Tempo total (execução + pré-processamento) por experimento e algoritmo. Escala logarítmica no eixo Y para permitir visualização dos valores menores.

O gráfico reforça que o experimento *pior-caso* domina a escala em tempo absoluto (*Força Bruta* ≈ 12 ms). Com escala logarítmica, é possível comparar os demais experimentos: *Boyer-Moore* (azul) tende a ter as barras mais baixas em *alfabeto-grande*, *padrão-longo*, *texto-grande* e *pior-caso*. *Rabin-Karp* (marrom) apresenta barras altas em *texto-grande* e *texto-médio*. Em *texto-pequeno*, os tempos são similares entre algoritmos.

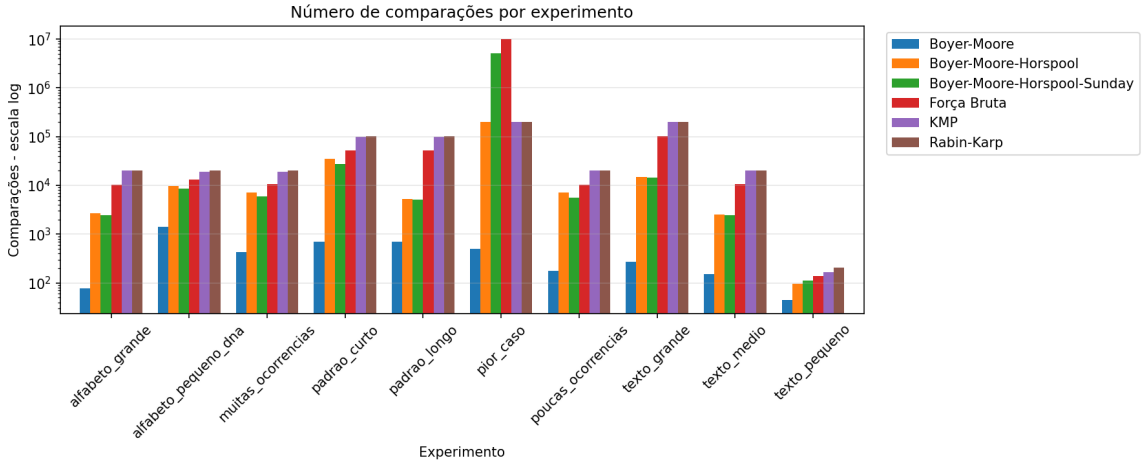

Figura 2 – Número de comparações por experimento e algoritmo. Escala logarítmica no eixo Y.

As comparações refletem a eficiência algorítmica. *Boyer-Moore* (azul) apresenta número de comparações muito baixos na maioria dos experimentos, especialmente em *alfabeto-grande*, *padrão-longo*, *poucas-ocorrências* e *pior-caso*. *Força Bruta* (vermelho) e *Boyer-Moore-Horspool-Sunday* (verde) explodem em *pior-caso* (~10M e ~5M comparações). *KMP* e *Rabin-Karp* mantêm comparações próximas de $2n$ na maioria dos cenários.

4 Conclusões

Com base nos resultados, respondemos às questões propostas:

4.1 Qual algoritmo apresentou melhor desempenho médio?

Boyer-Moore foi o algoritmo com melhor desempenho médio. Nos experimentos com *texto-grande*, *padrão-longo*, *alfabeto-grande*, *poucas-ocorrências/muitas-ocorrências* e *pior-caso*, ele obteve o menor tempo total e o menor número de comparações. A combinação das heurísticas do mau caractere e do bom sufixo permite saltos grandes no texto, reduzindo drasticamente as comparações. O custo de pré-processamento ($O(m + |\Sigma|)$) é amortizado rapidamente em textos maiores.

4.2 Em quais casos o algoritmo de Força Bruta se mostrou inviável?

Força Bruta se mostrou inviável em:

- *Pior-caso*: 12 ms e quase 10 milhões de comparações contra 0,3 ms do *Boyer-Moore*, cerca de 40 vezes mais lento.
- *Texto-grande*: 0,365 ms e 101 mil comparações contra 0,038 ms e 274 comparações do *Boyer-Moore*, 10 vezes mais lento e 370 vezes mais comparações.
- *Padrão-longo*: 52 mil comparações contra 699 do *Boyer-Moore*.

Em textos muito curtos (95 caracteres), *Força Bruta* ainda é competitiva por não ter custo de pré-processamento, mas a partir de textos médios e grandes torna-se claramente desvantajosa.

4.3 O custo de pré-processamento do KMP e do Boyer-Moore compensa?

Sim, na maioria dos cenários. Em textos pequenos (95 caracteres), o pré-processamento do *Boyer-Moore* (0,057 ms) supera o tempo de busca (0,066 ms), e o algoritmo como um todo fica mais lento que *Força Bruta*. Porém, a partir de textos médios (10 mil caracteres), o pré-processamento representa menos de 10% do tempo total e o ganho em comparações compensa amplamente. Em *texto-grande*, *Boyer-Moore* gasta 0,003 ms em pré-processamento e 0,035 ms na busca, totalizando 0,038 ms, 10 vezes mais rápido que *Força Bruta*. Para *KMP*, o mesmo padrão se observa: em textos grandes o custo de pré-processamento é insignificante perante o ganho em pular comparações desnecessárias.

4.4 Qual algoritmo é mais simples de implementar e manter?

Força Bruta é o mais simples: dois laços aninhados que comparam caracteres posição a posição. Não há estruturas auxiliares, tabelas ou funções de hash. A simplicidade da *Força Bruta*, porém, não compensa o desempenho ruim em cenários reais. *Boyer-Moore-Horspool* ou *Boyer-Moore-Horspool-Sunday* oferecem bom compromisso entre simplicidade e eficiência para uso geral.