João Victor Ottoni Garcia e Rafael Carvalho Avidago Geraldo

Trabalho Prático 3

Brasil - São João del Rei/MG

2024

João Victor Ottoni Garcia e Rafael Carvalho Avidago Geraldo

Trabalho Prático 3

Trabalho de Projeto e Análise de Algoritmos relacionado a implementacao de um algoritmo que dada uma sequência de caracteres, verifica a correspondencia exata de padroes.

Universidade Federal De São João del Rei – UFSJ

Faculdade de Ciência da Computação

Orientador: Leonardo Chaves Dutra da Rocha

Brasil - São João del Rei/MG

2024

Sumário

1 Introdução

No desafio apresentado, sugere-se a implementação de algoritmos que lidem com a busca de padrões exatos em cadeias de caracteres. A tarefa proposta envolve a busca de uma substring específica dentro de uma string, conforme descrito em queries fornecidas como entrada. A implementação teve em vista a utilização eficiente de memória e processamento, utilizando estruturas de dados alocadas dinamicamente e respeitando as boas práticas da computação. Por exemplo, em administração de recursos, a exclusão de um recurso pode impactar recursos vizinhos, demandando uma estratégia para otimizar a eficiência total. Na atribuição de tarefas, pode ser preciso selecionar quais atividades realizar e quais dispensar para aumentar a eficiência ou os efeitos. A análise de desempenho foi medida através dos tempos de usuário e sistema, permitindo assim uma comparação direta entre as duas abordagens implementadas. Este relatório documenta o processo de desenvolvimento e os estudos a cerca dos algoritmos utilizados.

2 Conceitos Abordados No Trabalho

2.1 Algoritmo Boyer-Moore

O algoritmo Boyer-Moore é bem conhecido pela sua eficiência na busca de padrões em textos longos, em especial quando o padrão a ser encontrado é relativamente curto. Desenvolvido em 1977 por Robert S. Boyer e J. Strother Moore, este algoritmo é marcado por usar duas heurísticas principais que reduzem de maneira significativa o número de comparações necessárias durante a busca, sendo elas a heurística do bom sufixo e a heurística do mau caractere. A heurística do mau caractere funciona identificando caracteres do texto que não são correspondentes ao padrão. Já a heurística do bom sufixo permite que após uma falha de correspondência, o padrão seja deslocado para alinhar o próximo sufixo mais longo do padrão com o texto, evitando assim as comparações repetitivas. Essas duas heurísticas fazem com que o Boyer-Moore seja especialmente eficiente em casos onde existem muitas correspondências parciais, permitindo que o algoritmo ignore grandes porções do texto.

Entretanto, o algoritmo de Boyer-Moore não é isento de desvantagens. Em caso de padrões com caracteres repetidos, o desempenho pode piorar. Apesar disso, na maioria dos casos práticos, o algoritmo se mostra capaz de realizar comparações sublineares, tornando-o altamente eficiente e amplamente utilizado em sistemas de busca, editores de texto e ferramentas de processamento de linguagem natural, onde é essencial a capacidade de realizar buscas rápidas em grandes escalas de texto.

2.2 Algoritmo Knuth-Morris-Pratt

O algoritmo Knuth-Morris-Pratt, também conhecido por KMP, foi introduzido em 1977 por Donald Knuth, Vaughan Pratt e James H. Morris. Este algoritmo foi desenvolvido para ser eficiente ao evitar redundâncias, tornando-o particularmente eficaz em grandes textos com padrões repetitivos. O KMP começa com um pré-processamento do padrão, durante o qual é criada uma tabela de prefixo, que possibilita ao algoritmo evitar retrocessos após uma falha de correspondência.

Embora o KMP necessite de um pré-processamento para construir a tabela de prefixos, o que pode gerar uma sobrecarga inicial, essa parte é compensada pela eficácia durante a busca. O algoritmo oferece uma solução robusta e eficiente para problemas que envolvem a correspondência de padrões, proporcionando assim resultados rápidos e confiáveis em uma grande variedade de contextos.

3 Implementação

Neste Trabalho Prático, utilizamos dois algoritmos de casamento exato de caracteres, sendo um deles o Boyer-Moore e o outro o KMP. Dedicaremos esta seção à explicação da implementação do código e seu funcionamento.

3.1 Boyer-Moore

Como mencionado anteriormente, o algoritmo de Boyer-Moore foi implementado fazendo uso de duas heurísticas principais. O código começa com a função heurística_mau_caráter, que é responsável por construir uma tabela que armazena a última posição de cada caractere no padrão. Essa tabela é utilizada posteriormente durante a busca, para determinar o quanto o padrão pode ser deslocado no texto quando um não casamento é encontrado, evitando assim comparações desnecessárias.

A função boyer_moore_buscar realiza a busca, comecando do final do padrao e comparando-o com o texto. Se o padrao nao corresponder ao texto em uma posicao determinada, o algoritmo utiliza a tabela de mau carater para decidir o quanto do padrao deve ser deslocado a direita, permitindo assim grandes saltos no texto, ao inves de avancar uma posicao de cada vez.

3.1.1 Pseudocódigo

```
INICIO:
1. Preprocessar o padrão para as tabelas de MC e BS
2. i = 0;
3. enquanto i <= n - m faça
4.     j = m - 1;
5.     enquanto j >= 0 e texto[i + j] == padrao[j] faça
6.         j = j - 1;
7.     se j < 0 então
8.         retornar i;
9.         i = i + deslocamento do Bom Sufixo;
10.    caso contrário
11.        i = i + max(deslocamento do MC, deslocamento do BS);
12. retornar -1;
FIM
```

3.2 Knuth-Morris-Pratt

O algoritmo de Knuth-Morris-Pratt inicia com a construção da tabela de prefixo, que registra as informações sobre os prefixos do padrão que também são sufixos. Esse pré-processamento permite que o algoritmo avance de maneira eficaz no texto sem precisar retroceder e examinar novamente caracteres quando ocorre uma falha de correspondência.

A função kmp_buscar então utiliza a tabela de prefixos para realizar a busca no texto. Durante a busca, se os caracteres do texto e do padrão são correspondentes o algoritmo avança em ambos. Se uma discrepância é encontrada após uma série de correspondências, o KMP usa a tabela de prefixos para determinar o próximo ponto de comparação no padrão. Evitando assim redundância.

3.2.1 Pseudocódigo

```
INICIO:
1. Construir o array de falha do padrão:
2.     k = 0;
3.     para j de 1 até m-1 faça
4.         enquanto k > 0 e padrao[k] != padrao[j] faça
5.             k = array_falha[k - 1];
6.         se padrao[k] == padrao[j] então
7.             k = k + 1;
8.         array_falha[j] = k;
9. i = 0;
10. j = 0;
11. enquanto i < n faça
12.    se texto[i] == padrao[j] então
13.        j = j + 1;
14.        se j == m então
15.            retornar i - m + 1;
16.    caso contrário, se j > 0 então
17.        j = array_falha[j - 1];
18.    caso contrário
19.        i = i + 1;
20. retornar -1;
FIM
```

4 Resultados e Discussões

Nesta seção, mostramos os resultados alcançados ao realizar testes em ambos os algoritmos. Esses resultados são essenciais para a avaliação do desempenho dos algoritmos em várias situações de uso. A avaliação é feita com base em vários testes de performance, com ênfase no tempo de execução do usuário e do sistema.

4.0.1 Exposição Gráfica

4.0.1.1 Boyer-Moore

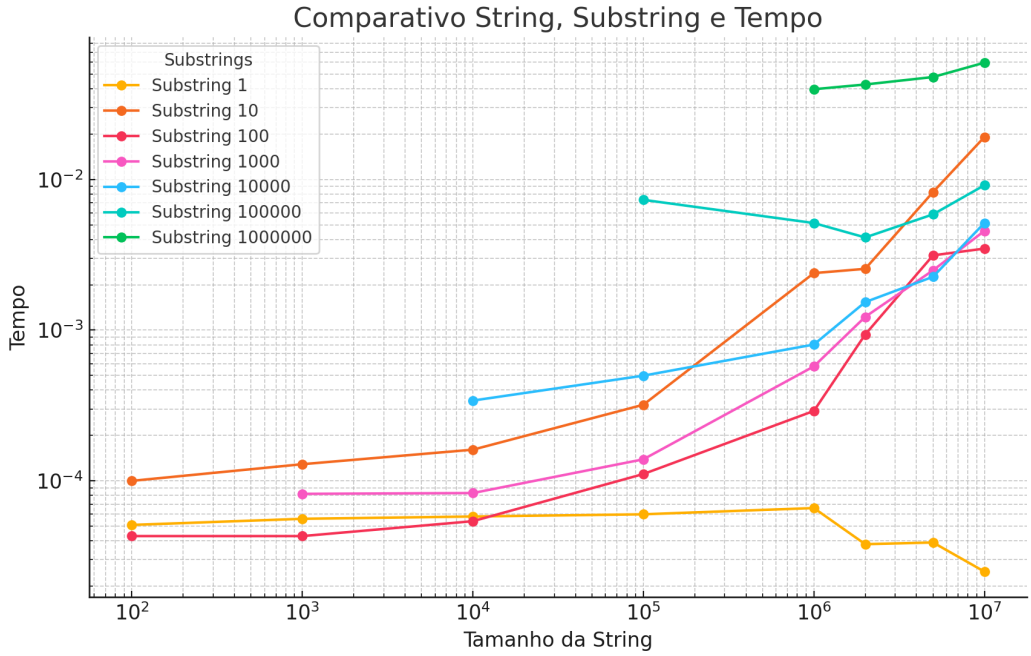

Figura 1 – Tempo de execução em função do tamanho da string e da substring.

O gráfico representa o comportamento do algoritmo Boyer-Moore. Embora seja conhecido por ser extremamente eficiente em muitos casos práticos, especialmente em buscas com grandes strings e grandes padrões, o gráfico indica que pode apresentar uma variação maior no tempo de execução dependendo do tamanho e da complexidade do padrão e do texto.

4.0.1.2 KMP

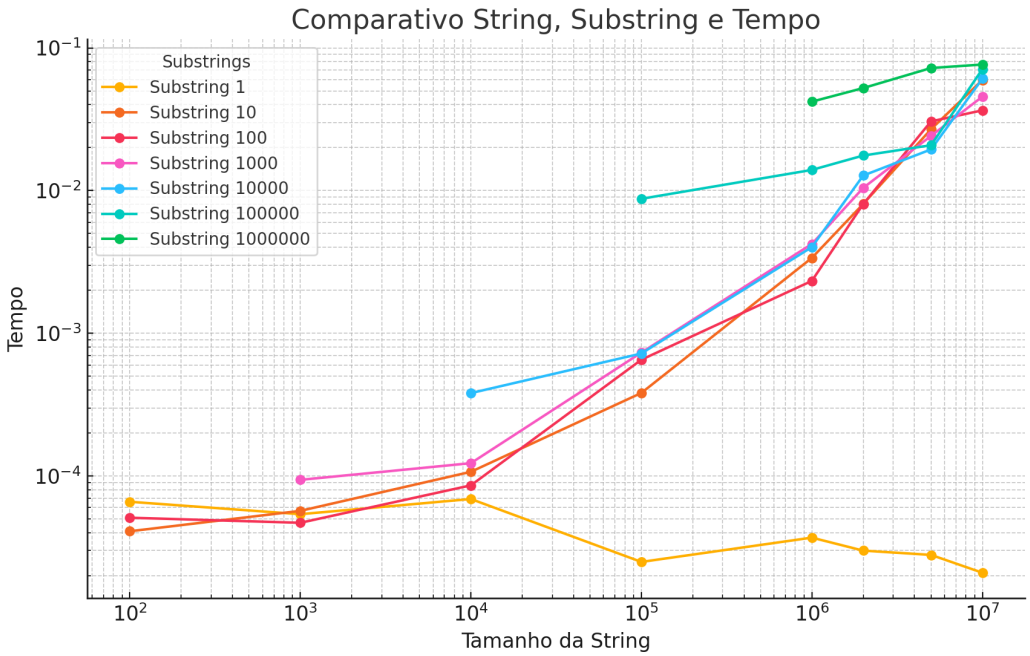

Figura 2 – Tempo de execução em função do tamanho da string e da substring.

O segundo gráfico reflete o comportamento do KMP. É notório que o tempo de execução do algoritmo cresce de forma lenta e consistente com o aumento do tamanho das strings, um comportamento esperado visto que o KMP é conhecido por sua eficiência para buscas de textos grandes. O gráfico exibe que mesmo com o aumento no tamanho da string, o KMP mantém um tempo de execução razoável, destacando sua escalabilidade.

4.0.2 Eficiência e Escalabilidade

Quando comparada a eficiência, o KMP, como evidenciado pelo segundo gráfico, demonstra um crescimento linear no tempo de execução em relação ao aumento da string. Isso mostra a capacidade do algoritmo de lidar com grandes volumes de dados de maneira eficiente, mantendo o tempo de execução sob controle de maneira previsível. A escalabilidade do KMP é clara, pois ele continua a funcionar de forma eficiente mesmo com o aumento da entrada de dados.

Em contrapartida, o algoritmo de Boyer-Moore, mostrado no primeiro gráfico, apresenta uma eficiência que, embora alta em casos práticos, pode variar dependendo das características da string e da substring. Sua escalabilidade pode ser menos consistente em comparações ao KMP, com variações no tempo de execução conforme o tamanho das strings aumenta. Apesar disso, em cenários ideais, Boyer-Moore pode demonstrar

um desempenho melhor que o algoritmo KMP, especialmente em textos e em padrões específicos onde suas otimizações são mais eficazes.

4.0.3 Análise de Complexidade

4.0.3.1 Boyer-Moore

O algoritmo de Boyer-Moore tem sua complexidade de tempo de $O(n/m)$, o que o torna mais eficiente para buscas em strings longas com substrings curtas, com ganhos significativos em termos de tempo de execução quando comparado ao KMP. Entretanto, no pior cenário, sua complexidade pode chegar a $O(n * m)$, especialmente em textos com uma grande quantidade de repetições.

4.0.3.2 Knuth-Morris-Pratt

O algoritmo KMP possui uma complexidade de tempo O(n + m), que pode ser simplificada para O(n), o que o torna uma boa escolha para buscas em textos longos com padrões de tamanhos variados. A análise de complexidade confirma que o KMP mantém um tempo de execução constante por iteração; isso se deve ao uso da tabela de prefixos.

4.0.4 Aplicações Práticas

Os dois algoritmos podem ser empregados em diversos contextos que necessitam de uma busca eficaz de padrões em conjuntos volumosos de informações. Geralmente são utilizados em editores de texto e sistemas de processamento de documentos, realizando tarefas como pesquisa e substituição. Em uma outra situação, na área da bioinformática, os algoritmos desempenham um papel crucial na análise de sequências de DNA, RNA e proteínas, sendo essencial para a identificação de sub-sequências específicas.

Outra área em que esses algoritmos são empregados é na compressão de dados, onde a identificação de padrões em fluxos de dados desempenha um papel crucial na diminuição do tamanho dos arquivos. Dentro da área de segurança da informação, esses métodos também podem ser utilizados para detectar malware e analisar pacotes de rede, auxiliando na identificação de padrões suspeitos em grandes conjuntos de dados. Além disso, os dois algoritmos podem ser empregados nas mídias sociais para identificar tendências ou tópicos emergentes por meio de uma análise em grande escala, a partir de palavras-chave ou frases repetidas nos comentários.

4.1 Conclusão

A comparação entre os algoritmos evidencia algumas diferenças significativas em termos de desempenho e aplicabilidade, dependendo das características do texto e do

padrão a serem analisados.

O KMP, com sua complexidade de tempo O(n + m), é especialmente eficiente em situações onde se trabalha com textos longos e padrões de vários tamanhos. Seu método de funcionamento evita a redundância de comparações ao pré-processar o padrão, garantindo assim uma performance estável e previsível, sendo particularmente vantajoso em situações onde o padrão possui muitos prefixos repetidos.

Por outro lado, Boyer-Moore demonstra um desempenho superior quando o texto é grande e o padrão curto, beneficiando-se de suas heurísticas. Estas permitem que o algoritmo "pule"grandes partes do texto, o que proporciona uma aceleração significativa em comparação ao KMP. Entretanto, em casos onde o padrão é muito repetitivo, ou a string é limitada, o desempenho de Boyer-Moore se aproxima de uma busca linear.

Em suma, explorando situações onde se deseja um desempenho consistente de maneira independente do padrão, o KMP se mostra mais eficiente, enquanto isso, o Boyer-Moore é preferível em contextos onde o padrão é curto com poucas variações e o texto extenso. A escolha entre ambos os algoritmos deve ser levada em consideração as características específicas do problema a ser resolvido.

5 Referências

Slide apresentado em sala de aula(Processamento de Cadeias de Caracteres.)

acervolima.com

geeksforgeeks.com