Processamento de Cadeia de Caracteres - Projeto 1

Alunos:

Lucas Henrique Cavalcanti Santos - lhcs

Roberto Costa Fernandes - rcf6

1. Identificação

O projeto descrito no presente relatório foi implementado pelos alunos Lucas Henrique Cavalcanti Santos e Roberto Costa Fernandes. Lucas ficou responsável pela implementação do algoritmo de Aho-Corasick, a interface de linha de comando e leitura de arquivos, enquanto Roberto ficou responsável pelos algoritmos Shift Or, Sellers e Wu-Manber.

Todo o código desenvolvido pela equipe pode se encontrado no Github: https://github.com/RC-Dynamics/string-processing.

2. Implementação

Algumas convenções serão utilizadas durante o relatório são elas:

- n é o tamanho do texto onde os padrões serão buscados;
- m é o tamanho do padrão a ser buscado;
- r é o erro máximo permitido pelos algoritmos de busca aproximada;

Nos algoritmos de Shift Or, Aho-Corasick e Wu-Manber é necessário definir um alfabeto a ser utilizado, assim nestes algoritmos o alfabeto é representado por uma string que contém os 128 primeiros caracteres da tabela ASCII.

2.1. Algoritmos para busca exata de padrões

Para os algoritmos de busca exata foram implementados o Shift Or e o Aho-Corasick. O algoritmo Shift Or foi escolhido pela sua simplicidade de implementação e por seu custo não modificar se houver ou não ocorrências. O algoritmo Aho-Corasick foi escolhido por ser possível realizar de maneira eficiente a busca paralelamente para diversos padrões. Porém o Shift Or tem uma limitação do tamanho quanto ao tamanho do padrão a ser procurado (irá ser explicado na seção 2.1.1). Logo as condições para a escolha automática do algoritmos são:

- Se fornecido apenas um padrão e m <= 64, irá ser executado o Shift Or;
- Se fornecido mais de um padrão, ou apenas um padrão com m > 64, será executado o Aho-Corasick;

Excluímos utilizar Shift Or para buscar vários padrões, porém sua execução não é eficiente, pois o algoritmo irá rodar de maneira sequencial para todos os padrões fornecidos, deixando de ser O(n) para ser O(num_padrões * n).

2.1.1. Shift Or

Na implementação do algoritmo Shift Or é necessário utilizar uma máscara de bits, para evitar o aumento da complexidade de implementação do algoritmo foi escolhido utilizar um *uint_fast64_t* da biblioteca *stdint.h*. Por esse tipo representar um inteiro em 64 bits, o tamanho máximo valor permitido de m é 64. Devido a esta limitação, a busca exata de padrão de tamanho maior que 64 é feita pelo Aho-Corasick.

2.1.2. Aho-Corasick

O Aho-Corasick usa como ideia construir uma árvore Trie com todos padrões a serem buscados. Na Trie os estados finais são o final das palavras, porém necessita atenção na aresta que sai das folhas, pois o final de uma palavra, pode ser o início de alguma outra, portanto deve ser considerado a borda entre os padrões para completar o algoritmo. Para representar a Trie foi usado um *unordered_map* indexado por um *pair* com o estado atual e letra escolhida. A única diferença no código foi a necessidade de construção de uma função de *hash* para o *unordered_map*, assim ele realiza em média consultas em tempo constante, no pior caso é linear.

2.2. Algoritmos para busca aproximada de padrões

Os algoritmos implementados para a busca aproximada de padrões foram o Sellers e o Wu-Manber. O Wu-Manber foi escolhido pois sua complexidade é O(m + n), enquanto o Sellers tem complexidade O(m x n), porém ele tem uma limitação quanto ao tamanho do padrão a ser buscado, por isso a escolha automática dos algoritmos ficou:

- Se o padrão fornecido tiver m <= 64, será escolhido o Wu-Manber;
- Caso contrário será escolhido Sellers;

Uma limitação da ferramenta é que não é permitido executar uma busca aproximada passando um arquivo contendo os padrões.

2.2.1. Wu-Manber

A implementação do Wu-Manber desenvolvida pela equipe contém a mesma limitação da Shift Or. Essa limitação ocorre pois a equipe escolheu utilizar a mesma

estrutura de dados do Shift Or para a máscara de bits, como explicada na seção 2.1.1.

2.2.2. Sellers

O algoritmo Sellers não tem a mesma limitação do tamanho do arquivo do Wu-Manber, pois não utiliza uma máscara de bits, porém o Sellers tem uma maior complexidade que o Wu-Manber.

3. Testes e Resultados

3.1. Realização dos Testes

Para realização dos testes foram escolhidos cinco arquivos com textos em inglês retirados do *Pizza&Chili* (http://pizzachili.dcc.uchile.cl/texts/nlang/). Os arquivos possuem tamanhos de 50MB, 100MB, 200MB, 1GB e 2GB, e foram testados com os algoritmos desenvolvidos pela equipe, variado o tamanho do padrão, procurado entre padrões de tamanho 1 e tamanho 25.

Além de utilizar a base de dados retirada do *Pizza&Chili*, também foram realizados os testes para todos os algoritmos com os mesmos padrões e com as mesmas condições na base *shakespeare.txt*. Esse teste foi realizado, pois o algoritmo de Aho-Corasick se mostrou ineficiente para arquivos muito grandes, o que impediu a realização dos testes na base do *Pizza&Chili*.

Pelo grande número de casos de teste a se avaliar, um *script* em *bash* foi desenvolvido pela equipe. O *script* executa o *pattern matching text* e salva o tempo de execução de cada busca em um arquivo *csv*, em cada execução do script é avaliado os tempos de execução dos padrões pré definidos, e nos quatro tamanhos de texto diferentes.

A escolha do algoritmo a se utilizar no script é feita através de passagem de parâmetro, já os padrões e arquivos são retirados de arquivos auxiliares.

Para comparar os algoritmos desenvolvidos com algum *ground-truth* foi incluído no script a opção de utilizar o *grep* e o *agrep*. Porém, por conta de algumas restrições do *agrep* quanto ao tamanho do erro e parâmetros de entrada, nem todos padrões foram testados nele.

Todos os testes foram realizados na mesma máquina e, dentro do possível, nas mesma condições. A configuração da máquina de testes é:

- Ubuntu 18.04
- Processador i7-2600
- 8 GB RAM
- 60GB SSD

Todas as imagens geradas pelos testes foram adicionadas ao repositório e ao entregáveis, porém nem todas estão presentes nesse relatório pois a equipe

julgou que ficaria repetitivo, as imagens se encontram em: projeto1/test/graphs/imgs.

3.1.1. Algoritmos Exatos

A avaliação dos algoritmos exatos foi realizada com o grep, Shift-Or e Aho-Corasick, porém o algoritmo Aho-Corasick só foi utilizado com arquivos de 5MB, devido a demora na realização dos testes com este algoritmo para casamento de um único padrão. Portanto para avaliar o Aho-Corasick no seu caso ideal, também foi realizado testes de busca de múltiplos padrões.

Foi incluído na busca exata os algoritmos aproximados *agrep*, Wu Manber e Sellers com erro 0, aumentando o número de algoritmos comparados.

3.1.1.1. Teste em Arquivo de 5MB

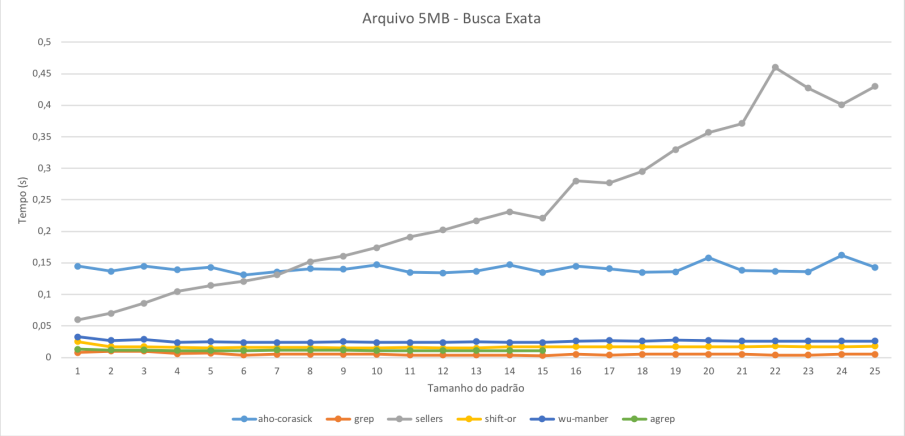

Figura 1 - Busca exata em arquivo 5MB

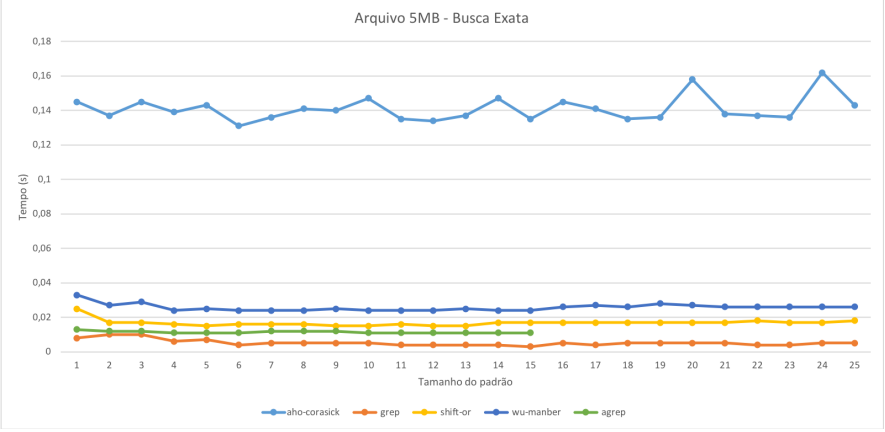

Figura 2 - Busca exata em arquivo 5MB

Na busca exata de um padrão em arquivo de 5MB de palavras em inglês, observa-se na Figura 1, o crescimento linear do Sellers com o tamanho do padrão procurado, e uma certa desvantagem do Aho-Corasick para os outros. Quando observamos a Figura 2, vemos que o Aho-Corasick fica próximo aos 140ms,

enquanto todos os outros algoritmos ficam abaixo de 40ms, e com performance praticamente constante. Neste teste o grep se mostra como opção mais eficiente.

3.1.1.2. Teste em Arquivo de 50MB

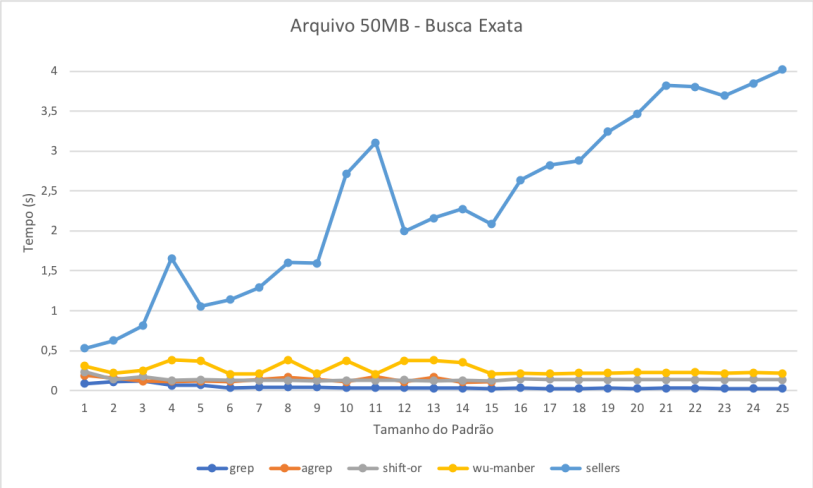

Figura 3 - Busca exata em arquivo 50MB

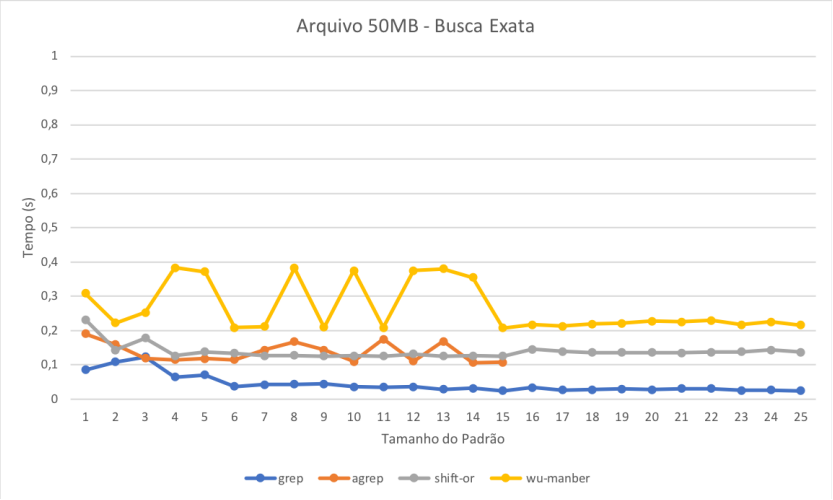

Figura 4 - Busca exata em arquivo 50MB

As figuras 3 e 4 mostram os resultados do tempo da busca exata do algoritmos implementados para um arquivo de *english.50MB*. Como pode ser observado na figura 3, o algoritmo de Sellers tem seu desempenho piorado a medida que aumenta-se o tamanho do padrão, por isso ele foi removido da figura 4, para facilitar a comparação dos outros algoritmos.

3.1.1.3. Teste em Arquivo de 100MB

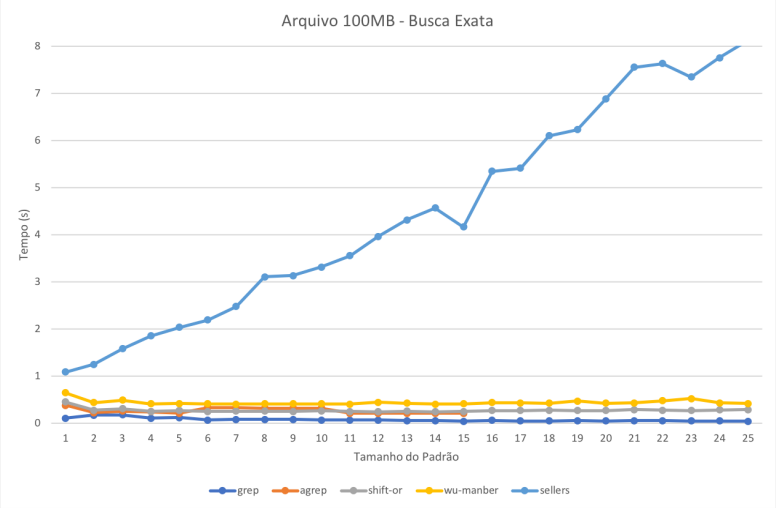

Figura 5 - Busca exata em arquivo 100MB

Dobrando o tamanho do arquivo, observa-se na Figura 5 um aumento proporcional nos tempo de busca, e evidencia ainda mais a relação dos algoritmos com o tamanho do padrão e arquivo.

3.1.1.4. Teste em Arquivo de 200MB

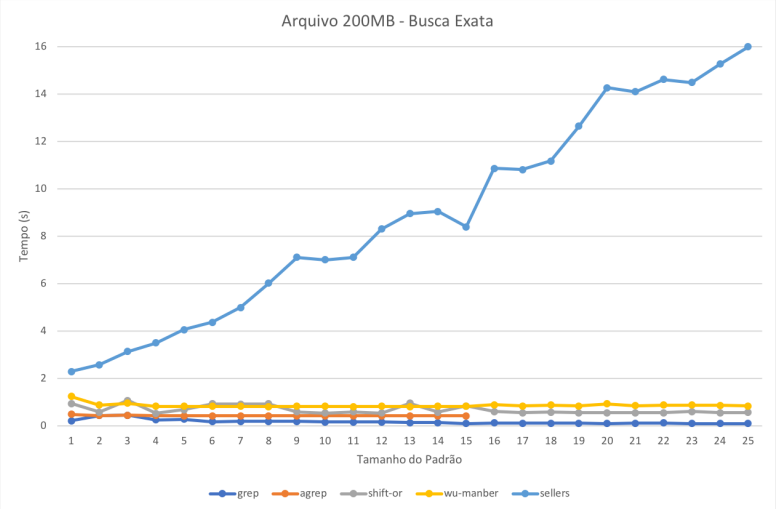

Figura 6 - Busca exata em arquivo 200MB

Aumentando o arquivo de busca para 200MB, é possível observar através da Figura 6, que o comportamento dos algoritmos não foi alterado significativamente, houve apenas um crescimento linear do tempo baseado no tamanho do arquivo.

3.1.1.5. Teste em Arquivo 1024MB

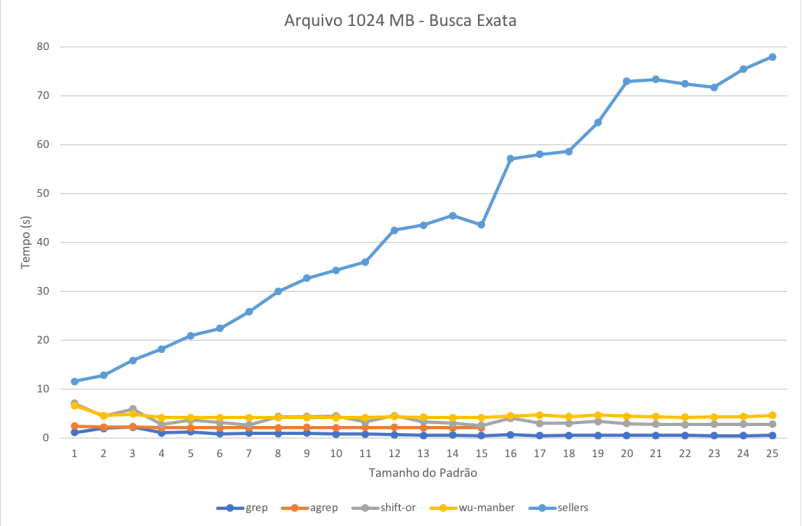

Figura 7 - Busca exata em arquivo 1024MB

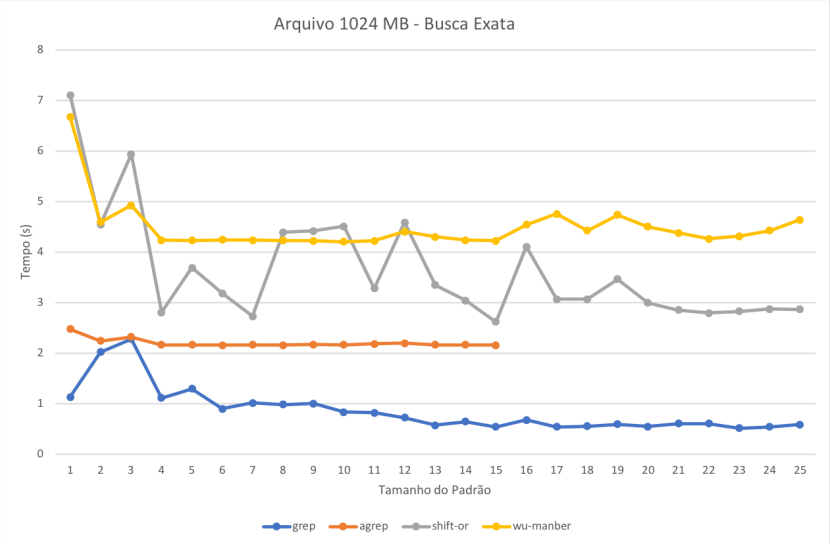

Figura 8 - Busca exata em arquivo 1024MB

Quando utilizado um arquivo de 1024MB observa-se através da Figura 7, que os comportamentos dos algoritmos se mantiveram, porém o tempo de busca aumentou novamente conforme aumentou o tamanho do arquivo.

Olhando a Figura 8, nota-se que a classificação dos algoritmos por tempo de busca, permanece a mesma desde o arquivo de 5MB.

3.1.1.6. Teste em Arquivo 2108MB

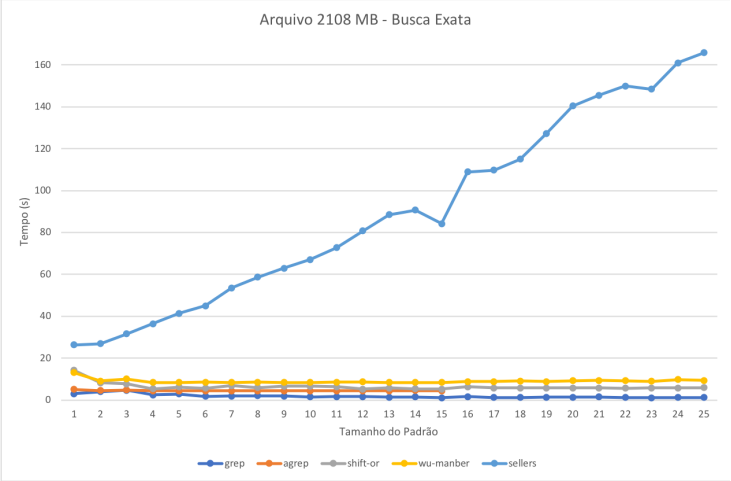

Figura 9 - Busca exata em arquivo 2108MB

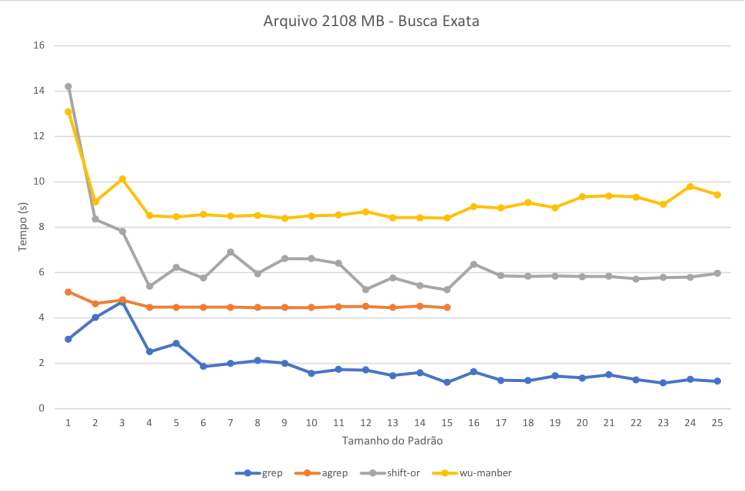

Figura 10 - Busca exata em arquivo 2108MB

Agora no maior arquivo de 2108MB pode-se observar através da Figura 9, que os algoritmos desenvolvidos mantiveram seu desempenho proporcional ao tamanho do arquivo, ou seja, comparando com a Figura 7 os algoritmos dobraram seu tempo.

Também fica evidente na Figura 10, que o Wu-Manber e Shif-Or começam com o tempo de procura maior, pois é afetado pelo número de ocorrências daquele padrão no texto.

3.1.1.7. Múltiplos Padrões em Arquivo 5MB

Este teste foi realizado entre o Aho-Corasick e o grep, variando o número de padrões procurados, para observar a dependência do algoritmo com número de padrões procurados.

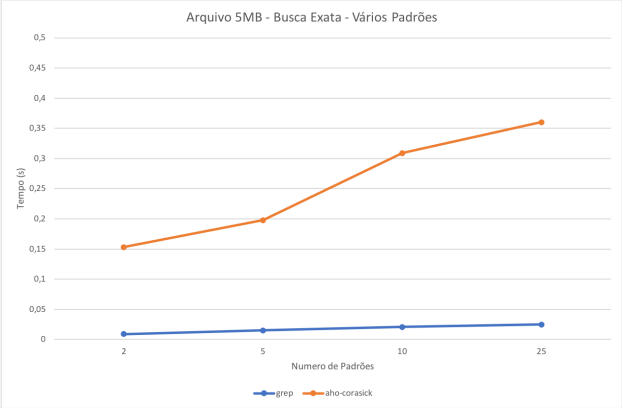

Figura 11 - Busca exata de múltiplos padrões em arquivo 5MB

Através da Figura 11 é possível observar que o Aho-Corasick é mais lento quanto mais padrões são procurados. Comparado ao grep, o Aho-Corasick implementado perde em todos casos.

3.1.2. Algoritmos Aproximados

No caso dos algoritmos Wu Manber e Sellers, além de variar as entradas como descrito anteriormente variou-se o tamanho do erro entre 0 e 7, criando ainda mais casos de testes. No relatório só estarão presentes os resultados para erro 2 e 7 e somente nos arquivos de 100MB e 2108MB, porque foi observado que o tamanho do arquivo não estava gerando nenhum *overhead* além do esperado. Outra modificação apresentada é que o tamanho do padrão tem que sempre ser maior que erro r, por isso os gráficos de erro 2 começa com padrões tamanho 3 e os gráficos de erro 7 começam com padrões de tamanho 8.

3.1.2.1. Erro máximo 2

3.1.2.1.1. Teste em Arquivo 100MB

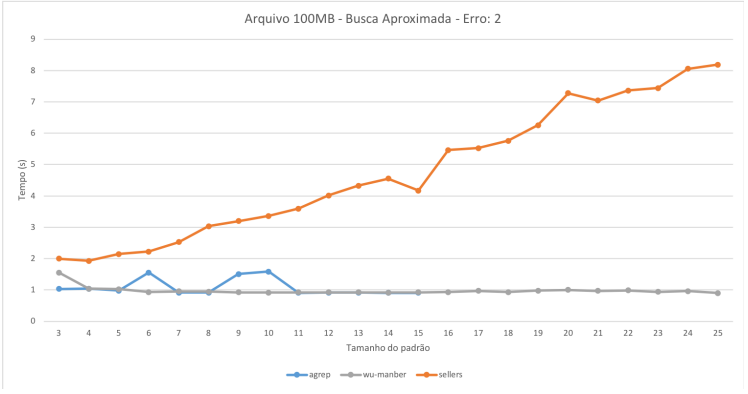

Figura 12 - Busca aproximada com erro máximo de 2 no arquivo de 100MB

Com erro dois, podemos observar através da Figura 12, que o Sellers manteve seu comportamento desde da busca exata, e que o Wu-Manber ganhar do agrep a partir do segundo padrão.

3.1.2.1.2. Teste em Arquivo 2108MB

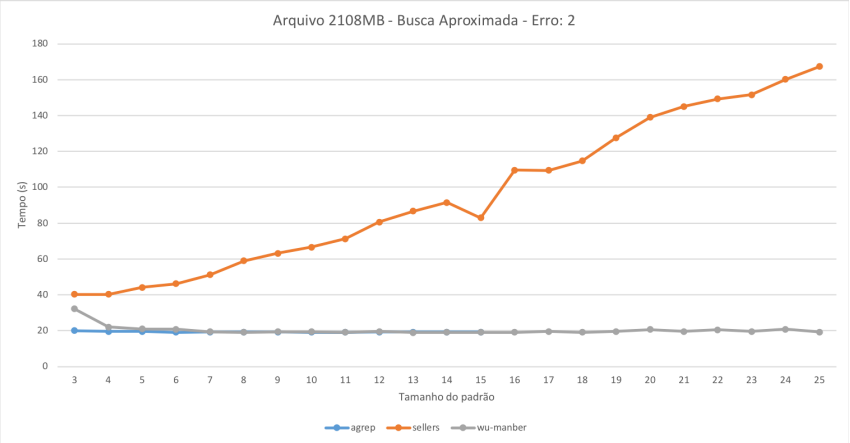

Figura 13 - Busca aproximada com erro máximo de 2 no arquivo de 2108MB

No teste com 2108MB, apresentado na Figura 13, os comportamentos se mantiveram, apenas crescendo o tempo proporcionalmente ao tamanho do arquivo procurado. Também é possível observar que os padrões pequenos, os quais mais ocorrem, afetam a performance do Wu-Manber.

3.1.2.2. Erro máximo 7

3.1.2.2.1. Teste em Arquivo 100MB

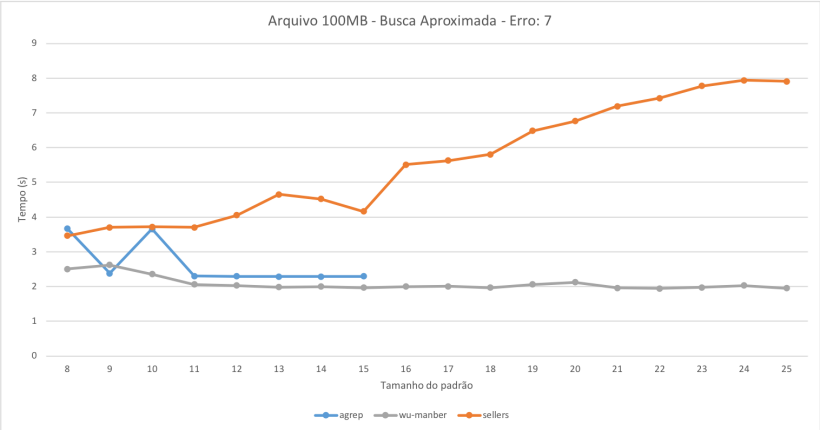

Figura 14 - Busca aproximada com erro máximo de 7 no arquivo de 100MB

Com erro de sete caracteres, podemos observar através da Figura 14, que os tempos de busca para todos algoritmos cresceram, e que o Wu-Manber continua ganhando do agrep e Sellers.

3.1.2.2.2. Teste em Arquivo 2108MB

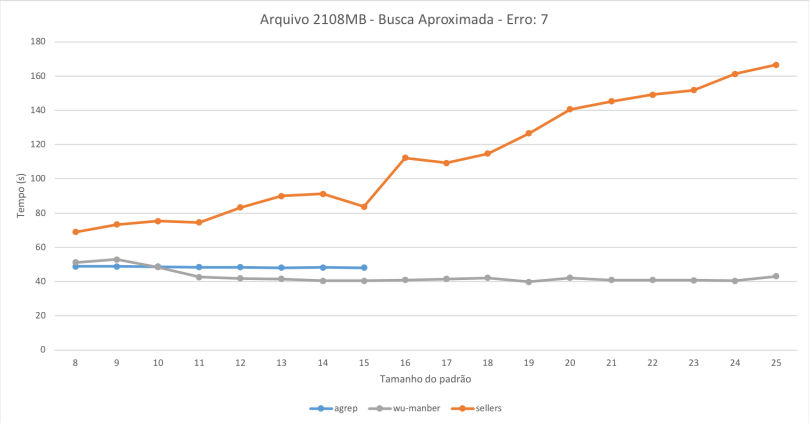

Figura 15 - Busca aproximada com erro máximo de 7 no arquivo de 2108MB

Ao aumentar o tamanho do arquivo e a quantidade de erros, observamos na Figura 15, que os tempos de cada busca ficaram maiores que 40 segundos, mas o WU-Manber mais uma vez se manteve como o melhor algoritmo.

4. Conclusões e Trabalhos Futuros

Com os resultados obtidos foi possível obter algumas conclusões:

- O fato do algoritmo Shift Or se mostrar com um desempenho muito bom, próximo ao grep, já era esperado por causa da sua baixa complexidade;
- O algoritmo de Aho-Corasick demonstrou um resultado ruim, o que estava dentro do esperado, porém esse algoritmo tem como vantagem a possibilidade de busca de vários padrões em paralelo;
- O fato do algoritmo de Aho-Corasick não conseguir rodar em arquivos muito grandes foi inesperado;
- Com o algoritmo de Wu-Manber também foi possível observar um bom desempenho, o que era esperado por ele ser uma extensão do Shift Or para busca aproximada;
- O fato do algoritmo de Sellers ter um crescimento aproximadamente linear no tempo de execução à medida que o tamanho da padrão aumenta é comprovada por sua complexidade ser O(m x n);
- A ferramenta *agrep* tem resultado estranho, o que gera uma desconfiança nos seus resultados, porém sua utilização foi necessária para servir como base de comparação para os algoritmos de busca aproximada;
- Na busca por múltiplos padrões a eficiência do Aho-Corasick varia bastante dependendo da intersecção do prefixo dos padrões;
- Na busca aproximada como esperado, ao aumentar o número de erros todos os algoritmos ficaram mais lentos. E o tempo cresceu também proporcionalmente ao tamanho do arquivo.

Como trabalhos futuros e melhorias para o projeto apresentado fica como ideias:

- Implementar a busca de vários padrões para todos os algoritmos, para servirem como comparação para o Aho-Corasick;
- Modificar a estrutura de dados utilizada no Aho-Corasick para uma implementação mais eficiente;
- Modificar o Aho-Corasick para que fique possível realizar a busca em arquivos maiores;
- Utilizar uma estrutura de dados para tornar possível a busca do Shift Or e Wu-Manber para padrões com tamanho maior que 64;
- Realizar a busca para padrões com tamanho maior que 25;
- Utilizar outras bases de dados para servirem de comparação para os algoritmos implementados.
- Utilizar outras implementações de busca aproximada para servirem como base de comparação para os algoritmos de busca aproximadas;