Processamento de Cadeia de Caracteres - Projeto I

Ferramenta PMT

1. Alunos

- **José Nilton de Oliveira Lima Júnior <jnolj>**, implementou os algoritmos Wu-Manber e Boyer Moore. Implementou o frontend da ferramenta.
- **Júlia Feitosa <mjfc>**, implementou os algoritmos Aho-Corasick e Sellers. Realizou os testes.

2. Implementação

A ferramenta pmt desenvolvida possui as opções requisitadas na especificação do projeto. A notação utilizada neste relatório é a seguinte:

m = tamanho do padrão

n = tamanho do texto

p = nº de padrões do Aho-Corasick

α *= tamanho do alfabeto*

e = erro máximo da busca aproximada

A linguagem escolhida pela equipe foi C++. Os algoritmos desenvolvidos foram o Boyer-Moore, Aho-Corasick, Sellers e Wu-Manber. O Boyer-Moore é o que possui o melhor desempenho para padrões com $m > 5$, porém para padrões menores ele não é tão eficiente. Aho-Corasick

Dos algoritmos de busca aproximada, o Sellers possui uma performance superior nos casos em que o $e \rightarrow m$, porém o seu desempenho cai à medida que m aumenta. Já o Wu-Manber tem um desempenho estável mesmo para padrões maiores, porém sua performance piora quando $e \rightarrow m$, independente do tamanho do padrão. O alfabeto utilizado foi o ASCII estendido, com suporte a 256 caracteres. Uma classe foi criada para cada algoritmo, com funções auxiliares necessárias para as respectivas implementações. O desempenho de cada algoritmo está detalhado na seção de testes, e o critério de escolha do pmt é o seguinte:

Para a busca exata: Caso haja mais de um padrão, o Aho-Corasick é escolhido. Senão, o Boyer-Moore é selecionado.

Para a busca aproximada: Caso haja mais de um padrão ou $e > 4$ Sellers é escolhido. Senão, o Wu-Manber é selecionado.

2.1 Leitura de arquivos

Para que o os algoritmos de casamento de padrões sejam analisados de forma clara, é necessário que a operação de leitura de arquivos seja constante e otimizada, de forma a possibilitar comparações com ferramentas como o grep, que tem sua leitura de arquivos otimizadas. Para tal, foi criado na ferramenta uma classe para gerenciar a leitura dos arquivos

que utiliza a API em C de leitura através de file descriptors. Os arquivos são lidos em chunks de 327680 bytes, que são colocados em um buffer. Esses bytes são processados e é entregue através da função *FileReader::getLine()* a próxima linha não lida buffer. Caso não haja mais dados para se ler no buffer, um novo chunk é lido e sua primeira linha é entregue. Caso se tenha chegado ao final do arquivo, a função irá retornar -1, indicado que a leitura não foi bem sucedida.

A leitura de arquivos apresenta um caso especial, que é quando uma linha começa num chunk e termina em outro. Nesse caso, é retornada para a função *FileReader::getLine()* o pedaço da linha presente no buffer e um retorno igual a 1, que avisa que a string lida não representa uma linha completa. Essa estratégia foi escolhida pois no pior caso (uma linha dividida em vários chunks), o custo de memória da leitura não é aumentado. Dessa forma, foi necessário adaptar os algoritmos de busca para que os mesmos possam "continuar" seu processamento a partir de uma metade da linha. As seguintes modificações foram feitas nos algoritmos:

- Boyer Moore: Caso não seja o final da linha, os últimos *m* caracteres da linha são salvos e concatenados no início da próxima string processada.
- Aho Corasick: Caso não seja o final da linha, os estados da máquina de estado são salvos e retomados na próxima busca.
- Sellers: Caso não seja o final da linha, é salvo a linha atual na matriz de programação dinâmica que o processamento parou, além de sua tabela não ser resetada.
- Wu-Manber caso não seja o final da linha, a matriz S não é resetada.

2.1 Algoritmos de Casamento Exato

Os algoritmos escolhidos pela equipe foram o Boyer-Moore, pelo seu desempenho satisfatório para padrões grandes e o Aho-Corasick, que permite a busca eficiente de vários padrões. O Aho-Corasick foi implementado utilizando um <vector> de duas dimensões para armazenar os estados do autômato. As dimensões máximas são α x ($\sum_{i=1}^{p} m$) +1. O melhor caso do seu desempenho é para a busca de padrões individuais pequenos em arquivos grandes e para a busca de múltiplos padrões. Os caracteres do alfabeto são mapeados para inteiros que devem estar entre 0 e 256. O Boyer-Moore foi implementado de acordo com o que foi demonstrado em sala, utilizando as heurísticas do Good Char e Bad Suffix

2.2 Algoritmos de Casamento Aproximado

Os algoritmos escolhidos pela equipe foram o Sellers, por sua simplicidade e o Wu-Manber, devido à sua robustez para padrões maiores de tamanho até 64. Ele utiliza o tipo *uint_fast64_t* para otimizar a busca.

O Sellers foi implementado seguindo o conteúdo visto em sala, utilizando como base o algoritmo de distância de edição. Esse algoritmo usa programação dinâmica para calcular o menor custo para as operações de inserção, remoção e troca que consegue transformar uma

palavra em outra. Para diminuir a complexidade de espaço, um array de tamanho *2 x m* é atualizado com os valores da distância de edição, reportando uma ocorrência sempre quando valor para o padrão inteiro é menor ou igual ao erro desejado. O bottleneck do desempenho do algoritmo é a necessidade de resetar os valores armazenados no array cada vez que um novo chunk do texto é processado, além disso o tamanho do array é diretamente proporcional ao tamanho do padrão e como $m << n$, o número de iterações do Sellers depende de n, o tornando lento para textos grandes.

3. Testes

Os testes foram executados a partir de scripts bash, que fazem a chamada à ferramenta pmt, com diferentes parâmetros inseridos pelo usuário dependendo do que se deseja testar. Os scripts encaminham para arquivos de texto informações sobre o tempo de execução do programa (determinado pelo valor "real" do comando *time*), sobre o tamanho do padrão ou o valor do erro utilizado. Os valores apresentados consideram a média do tempos de execução da ferramenta para os mesmos padrões 3 vezes. A ferramenta grep foi utilizada como benchmark, para os algoritmos exatos e agrep para os algoritmos de busca aproximada. Os scripts de testes do grep e agrep funcionam da maneira mencionada acima. A ferramenta pmt foi testada com as opções -c -a, algorithm_name ativas, a não ser que seja indicado o contrário. A parte de impressão da contagem de linhas foi desabilitada nos testes. Os textos utilizados foram obtidos de Pizza&Chili (http://pizzachili.dcc.uchile.cl/texts/nlang/). Os padrões foram obtidos de uma lista com as 1000 palavras mais comuns da língua inglesa (https://www.ef.com/english-resources/english-vocabulary/top-1000-words/) e também de trechos retirados diretamente dos textos, com não mais do que 200 caracteres. Para facilitar a utilização da ferramenta gnuplot, os padrões foram ordenados de acordo com seu comprimento.

Os testes foram realizados numa máquina virtual VMWARE Workstation, com Ubuntu 18.04.1 LTS 64-bit, 3.8GB de RAM, Intel® Core™ i7-7500U CPU @ 2.70GHz × 4, GNOME 3.28.2.

O computador host possui a seguinte configuração: Windows 10 Single Home Language 1703 64-bit, 8 GB de RAM , Intel® Core™ i7-7500U CPU @ 2.70GHz - 2.90GHz.

3.1 Casamento Exato

O script do primeiro teste utiliza um arquivo de padrões, lendo um por vez e executando a ferramenta para cada padrão. Os algoritmos testados foram o Boyer-Moore, Aho-Corasick a ferramenta grep. O tamanho do padrão e o output do pmt foram redirecionados para um arquivo, e após o cálculo da média, a ferramenta gnuplot foi utilizada para a construção dos gráficos. Neste teste, um texto em inglês de 1 GB foi usado juntamente com um arquivo de 7 padrões, que vão desde palavras em inglês a frases retiradas do texto.

O resultado após a execução do primeiro teste encontra-se na Figura 1. O algoritmo Aho-Corasick é o que possui o menor tempo de execução para padrões com *m* <= *2*, superando o desempenho do Boyer-Moore para padrões com *m* <= *4*. O algoritmo Boyer-Moore é o que mais se aproxima do desempenho da ferramenta grep.

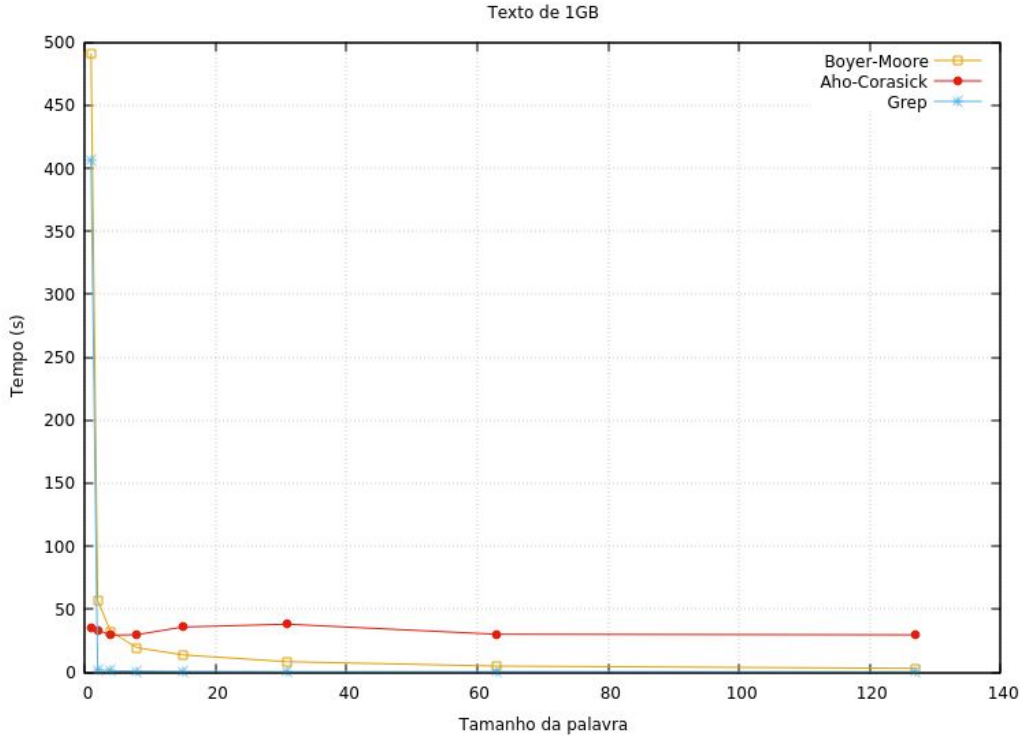

Figura 3.1 - Teste 1: Comparação do desempenho de algoritmos na busca exata de um padrão por vez

O segundo teste permitiu analisar melhor o desempenho dos algoritmos de busca exata, dessa vez habilitando a opção -p, do pmt e -f, do grep, para que as ferramentas recebam como input um arquivo de padrões e utilizem seus mecanismos de leitura internos.

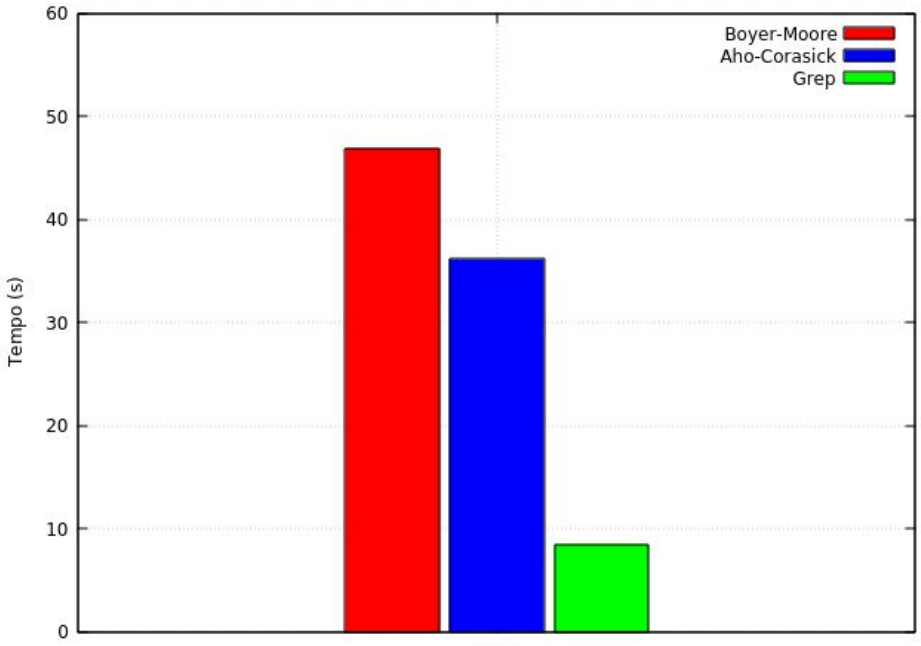

Figura 3.2 - Teste 2: Comparação do desempenho do Boyer-Moore, Aho-Corasick e Grep com um arquivo de padrões de 1 GB

O terceiro e quarto testes comparam o desempenho do Aho-Corasick em dois cenários: Com o número de padrões fixo e tamanho do texto variável e com o número de padrões variável, porém o mesmo texto. Os padrões utilizados tinham tamanho entre 1 e 15 caracteres.

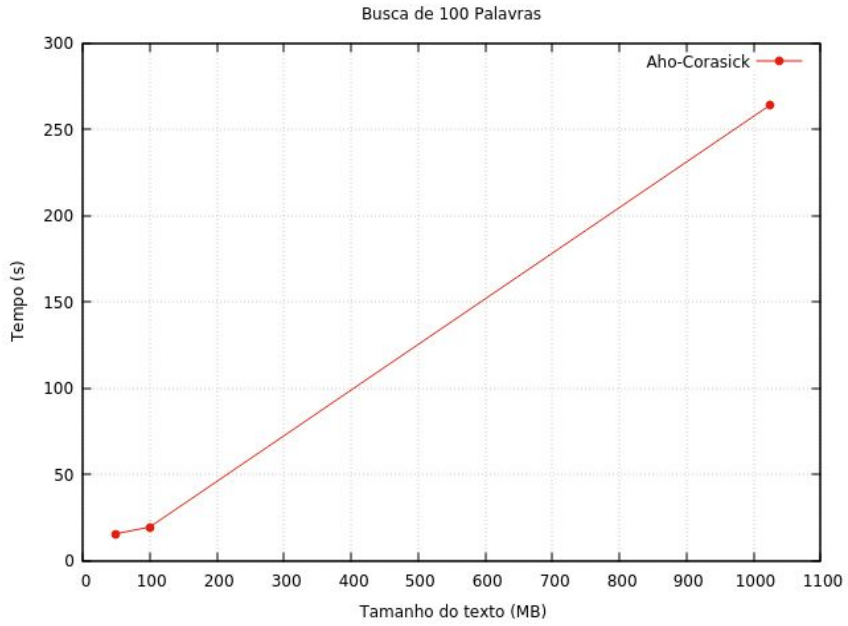

Figura 3.3 - Teste 3: Comparação do desempenho do Aho-Corasick para textos entre 100 e 1024MB e padrões com m = [1...15]

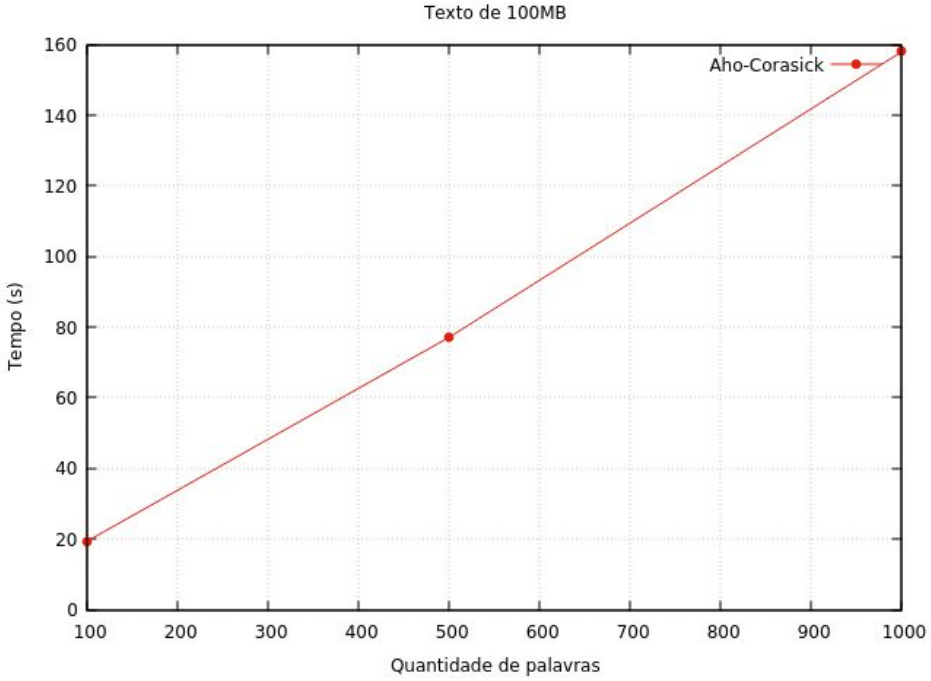

Figura 3.4 - Teste 4: Comparação do desempenho do Aho-Corasick um texto de 100MB e p = [100...1000]

3.2 Casamento Aproximado

O script do quinto teste realiza a leitura de um único padrão e executa os algoritmos de busca aproximada variando o erro entre 0 e m-2, onde m é o tamanho do padrão. Os algoritmos testados foram o Sellers e o Wu-Manber. O resultado do teste está presente na Figura 3. O padrão utilizado foi a palavra “responsibility”, de tamanho 14. Neste teste, um texto em inglês de 100MB. Nesse caso, o algoritmo de Sellers é mais vantajoso para um erro >= 4, pois o seu tempo de execução se mantém abaixo do Wu-Manber. É possível concluir que o Wu-Manber é mais apropriado para buscas onde o tamanho do padrão varia, porém o Sellers é mais vantajoso quando o padrão permanece o mesmo e o erro desejado muda.

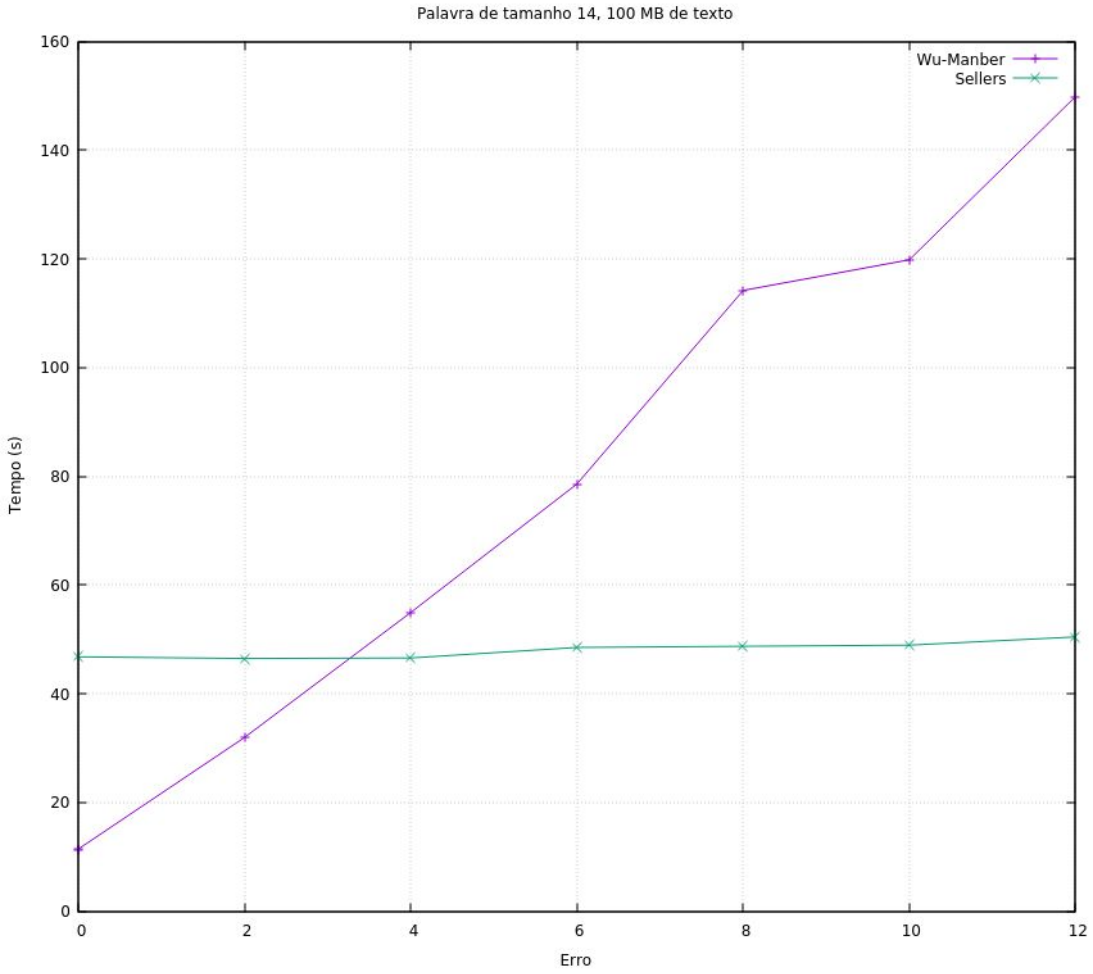

Figura 3.5 - Teste 5: Comparação entre Wu-Manber e Sellers para o mesmo padrão, porém com valores de erro diferentes

O sexto teste reforça o resultado do primeiro, mostrando que mesmo com o aumento no tamanho do padrão, o algoritmo de Sellers ainda é vantajoso para valores de $e > 4$, compensando a sua lentidão inicial. O desempenho do Wu-Manber é inversamente proporcional a diferença entre o tamanho do padrão e o valor do erro. O padrão utilizado é o mesmo do teste anterior, com $e <= 6$.

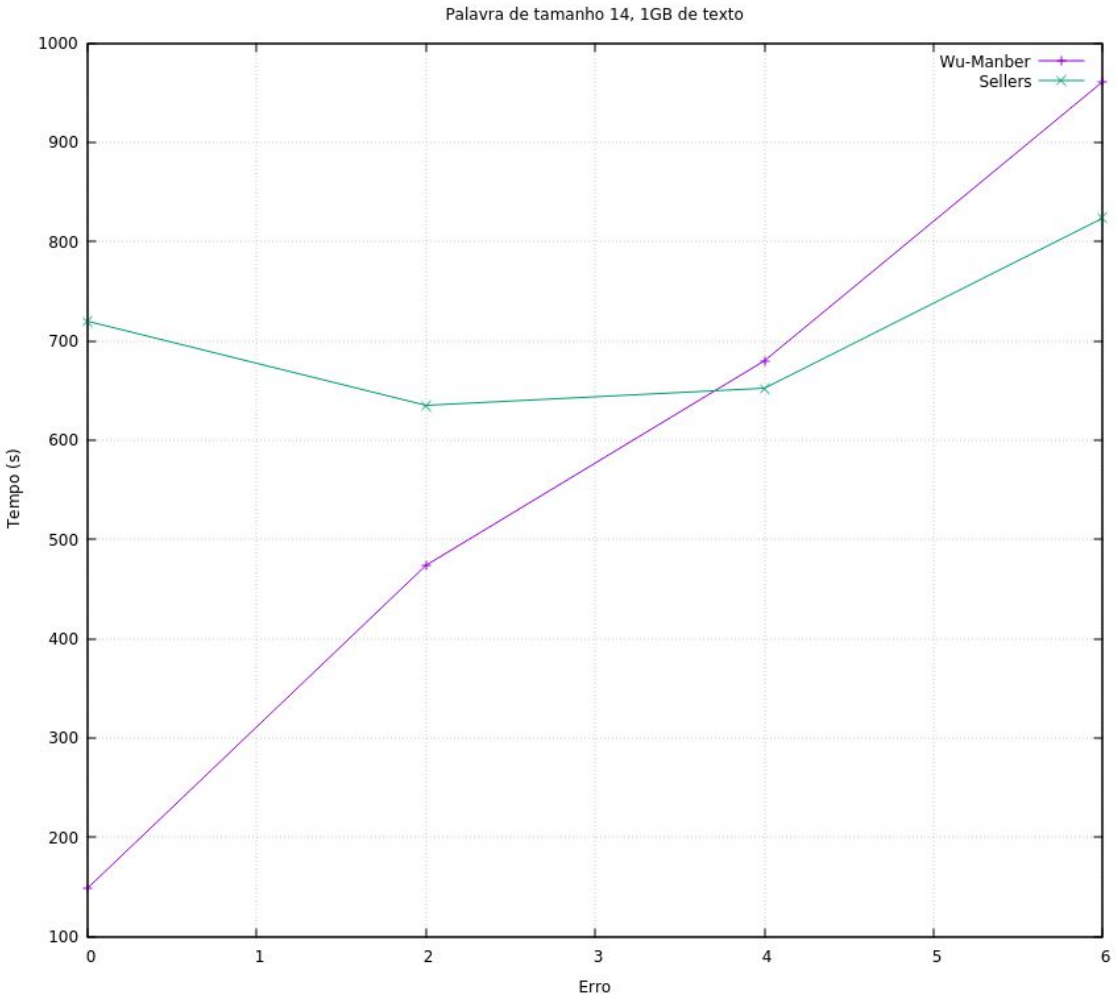

Figura 3.6 - Teste 6 : Comparação entre Wu-Manber e Sellers para o mesmo padrão, porém com valores de erro diferentes

O sétimo teste comparou o desempenho do agrep, dos algoritmos de busca aproximada para uma lista de padrões com $m = [1...8]$ & $e = m$-1 num texto de 100 MB. O agrep permite um erro máximo de 8, por isso m foi escolhido para respeitar essa limitação. O Sellers teve um desempenho melhor que o Wu-Manber, devido ao fato do erro ser próximo do tamanho do padrão.

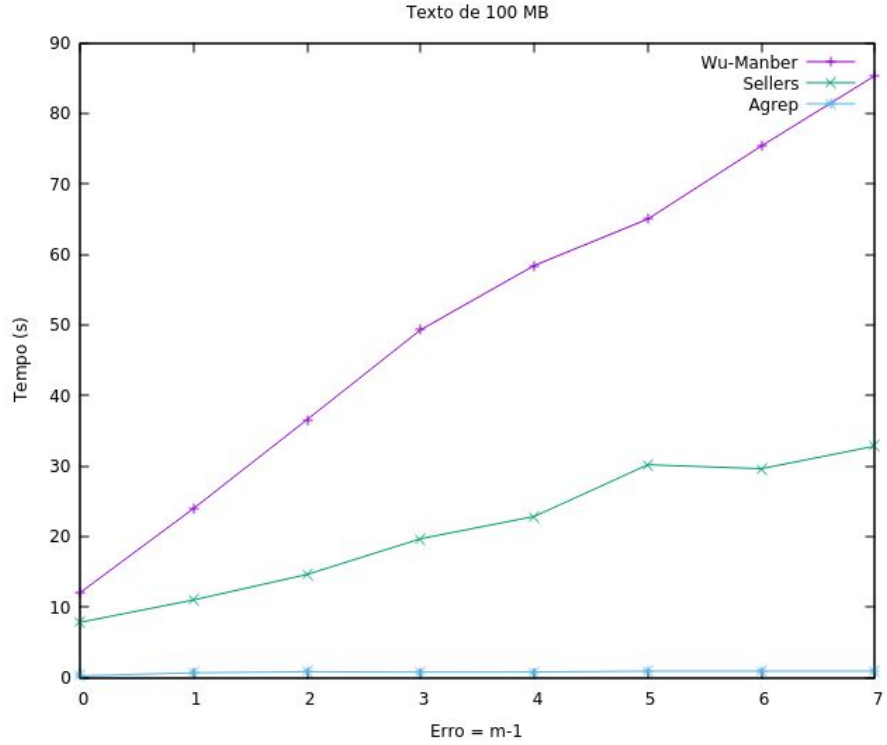

Figura 3.7 - Teste 7 : Comparação entre Wu-Manber e Sellers e Agrep para padrões diferentes, com o erro próximo ao tamanho do padrão

O oitavo teste mostra o desempenho do Wu-Manber até o tamanho limite permitido para um padrão *(m = 64)*, com *e = m - 1*, em um texto de 100 MB. Novamente o Sellers obteve uma performance melhor com o aumento do erro e para *e = 0* e *m = 1* os dois algoritmos tem o mesmo tempo de execução.

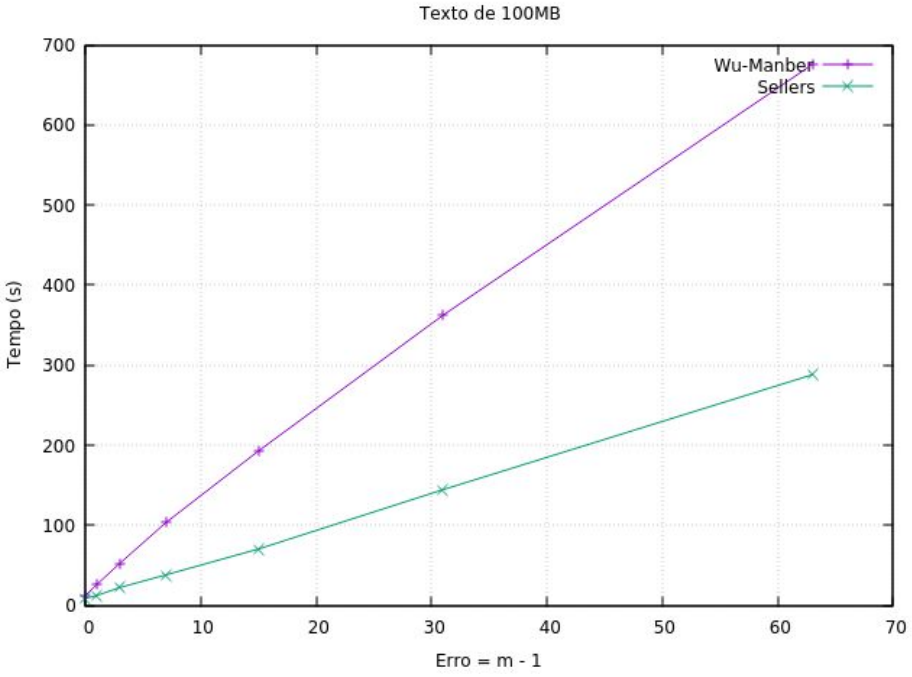

Figura 3.8 - Teste 8 : Comparação entre Wu-Manber e Sellers para *e = [0...63]*

4. Bugs

Durante os testes, um bug foi encontrado no Aho-Corasick. Utilizando a opção -c no Aho-Corasick para arquivos de texto grandes, a contagem de algumas palavras está diferente daquela encontrada por outros algoritmos. De acordo com os testes realizados, essa diferença não passou de +-10, para 10^4 ocorrências.

5. Conclusões

Analisando os resultados dos algoritmos de busca exata, podemos concluir que o Boyer-Moore é o mais eficiente (exceto para padrões pequenos) e é o que mais se aproxima da ferramenta grep. Já o Aho-Corasick é o mais indicado para o processamento de múltiplos padrões, com a melhor performance para a busca de mais de um padrão dentre os algoritmos implementados. No caso da busca de um único padrão, o Aho-Corasick supera o Boyer-Moore apenas em casos em que o padrão é muito pequeno. A ferramenta grep teve um tempo de execução menor do que todos os algoritmos implementados pela equipe. Para os textos de 100 MB, o tempo de execução do algoritmo mais rápido (Boyer-Moore) é no máximo de duas vezes maior que o grep. Já para um texto de 1GB, para padrões com $m >= 60$, o tempo de execução é aproximadamente cinco vezes maior.

Já para os algoritmos de busca aproximada, o Sellers possui um melhor desempenho para os casos em que o erro é próximo ao tamanho do padrão. Para casos em que o erro é consideravelmente menor que o tamanho do padrão, ou que o erro permanece constante e o comprimento do padrão aumenta, o Wu-Manber é o mais indicado. A ferramenta desenvolvida possui algumas vantagens em relação ao agrep, pois permite erros maiores do que 8, e no caso do Sellers, padrões maiores que 64.