Ferramenta PMT

Valdemir de Andrade - vabj@cin.ufpe.br
Mateus de Freitas Leite - mfl3@cin.ufpe.br

1 Introdução

O *pmt* é um programa utilizado para buscas de padrões em arquivos de textos, de maneira similar à ferramenta GNU Grep [1]. Possui a seguinte interface de uso:

$ *pmt* [*options*] *pattern textfile* [*textfile ...*]

Aqui, *pattern* é o padrão a ser procurado e *textfile*, um arquivo contendo o texto a ser processado à procura do padrão de entrada. Múltiplos arquivos de texto podem ser selecionados. Há também a possibilidade de especificar um conjunto de arquivos de texto a partir de *wildcards*.
As opções disponíveis são as seguintes:

- *-a --algorithm*: determina um algoritmo específico a ser utilizado pela ferramenta. Os argumentos válidos para essa opção são: *bm*, para o algoritmo de Boyer-Moore; *kmp,* para o KMP; *sel*, para o algoritmo de Sellers; *ukk*, para o de Ukkonen. Para os algoritmos de busca aproximada (Sellers/Ukkonen), é obrigatório a especificação da distância de edição (opção *-e --edit*).
- *-c --count*: imprime apenas a quantidade total de ocorrências do padrão no texto.
- *-e --edit*: determina a distância de edição (apenas para algoritmos de busca aproximada).
- *-p --pattern*: especifica um arquivo de texto contendo múltiplos parâmetros. Caso essa opção seja habilitada, o parâmetro obrigatório *pattern* é considerado como sendo um arquivo de texto.

Visto que os integrantes da dupla se juntaram tardiamente, cada um implementou sua própria versão inicial de todos os algoritmos de busca aproximada. Quanto aos algoritmos de busca aproximada, o de Sellers foi inicialmente implementado por Valdemir, enquanto o de Ukkonen fora designado para Mateus. Após a conclusão da implementação de todos os algoritmos escolhidos, os integrantes se reuniram para discutir ideias para otimização e correção das implementações. Apesar de ainda existirem algumas limitações quanto ao desempenho, avanços conseguiram ser realizados, tais como a leitura em bloco do arquivo de texto (economizando assim memória) e a otimização dos algoritmos de Boyer-Moore e o de Sellers.

2 Implementação

2.1 Algoritmos de busca exata

Para a busca exata, utilizamos 2 algoritmos (além do algoritmo de força bruta), sendo eles o KMP e o Boyer-Moore. Embora sejam ambos métodos de janela deslizante e, portanto, são bastante similares, eles foram escolhidos pela facilidade na escolha das estruturas de dados para a implementação das tabelas de deslocamentos.
O algoritmo padrão da ferramenta utiliza o KMP para padrões "pequenos". Em textos grandes, isso indica que há maiores chances de o KMP ser mais eficiente que o Boyer-Moore, pois o número de elementos necessários do alfabeto para o processamento do padrão é mais reduzido. Para padrões "grandes", utilizou-se o algoritmo de Boyer-Moore.
Muito embora fosse levado em conta o tamanho do alfabeto na escolha do algoritmo, houve uma tentativa de processar o conjunto utilizado dinamicamente, o que afetou no desempenho da ferramenta, ou seja, a falta da definição do alfabeto na parametrização da ferramenta fez com que não houvesse como determinar eficientemente seu tamanho. Por isso, a escolha do algoritmo de busca exata se deu apenas pelo tamanho do padrão.
Dessa maneira, seja m o tamanho do padrão de entrada. O algoritmo da ferramenta foi definido da seguinte forma:

- Se $m \leq 1$, utiliza-se o algoritmo de força bruta.
- Se $m > 5$, utiliza-se o Boyer-Moore.
- Caso contrário, utiliza-se o KMP.

2.2 Algoritmos de busca aproximada

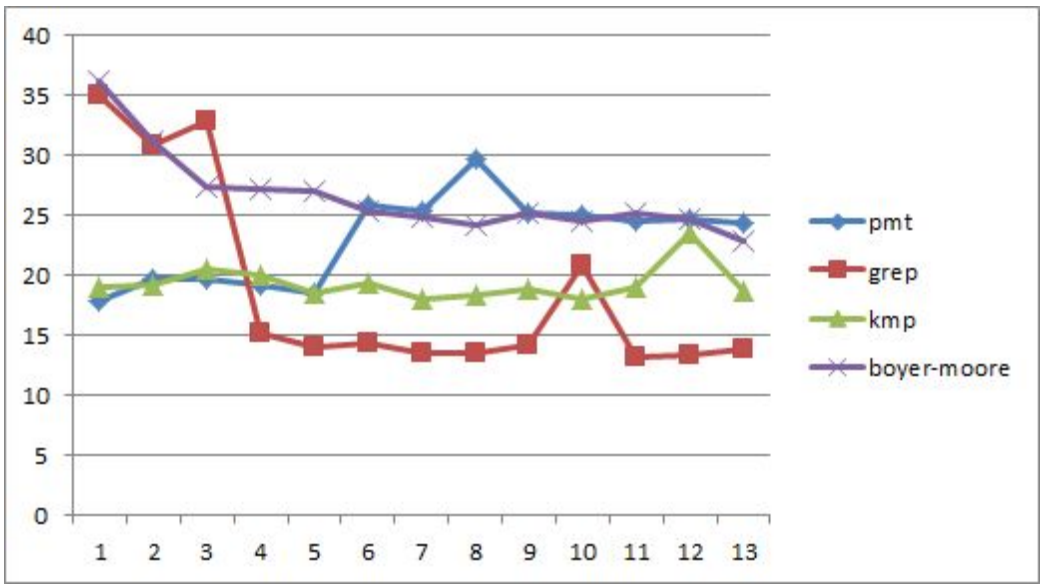
Para a busca aproximada, foram implementados os algoritmos o Sellers e o Ukkonen. A criação da máquina de estados do Ukkonen pode crescer muito dependendo do tamanho do alfabeto, mas o processamento do texto compensa nos casos em que os padrões são suficientemente pequenos, mas o texto em si é grande. O algoritmo do Sellers, por sua vez, possui complexidade de espaço linear - $o(m)$, onde m é o tamanho do padrão, mas seu processamento é da ordem de $o(mn)$, onde n é o tamanho do texto. Dessa forma, a ideia inicial era aproveitar o algoritmo de Sellers apenas para os casos em que o algoritmo de Ukkonen não fosse teoricamente viável. No entanto, a implementação da função de transição neste último se mostrou bastante ineficiente (tanto na teoria, como na prática), visto que se tratava de uma *hash* bastante sujeita à colisões. Dessa forma, o algoritmo padrão da ferramenta para busca aproximada é o de Sellers, para todo padrão P e todo texto T.

3 Testes e Resultados

Foram executados dois testes (um para busca exata, outro para busca aproximada) utilizando apenas um padrão de entrada. Ambos os testes foram feitos com textos em inglês. Eles foram executados em um *desktop* com processador AMD A10 Quad-Core (3.5 GHz), sistema operacional Ubuntu 14.10 e 8 GB de memória RAM.
Os arquivos de texto foram extraídos do *corpus* presente em [2]. O primeiro teste consistiu em um arquivo de texto de 1.1 GB, enquanto o segundo possuía um arquivo de 50 MB. O conjunto de padrões utilizados para os dois testes possui 13 cadeias. Cada um deles foi processado individualmente. Outros detalhes serão apresentados na seção seguinte.

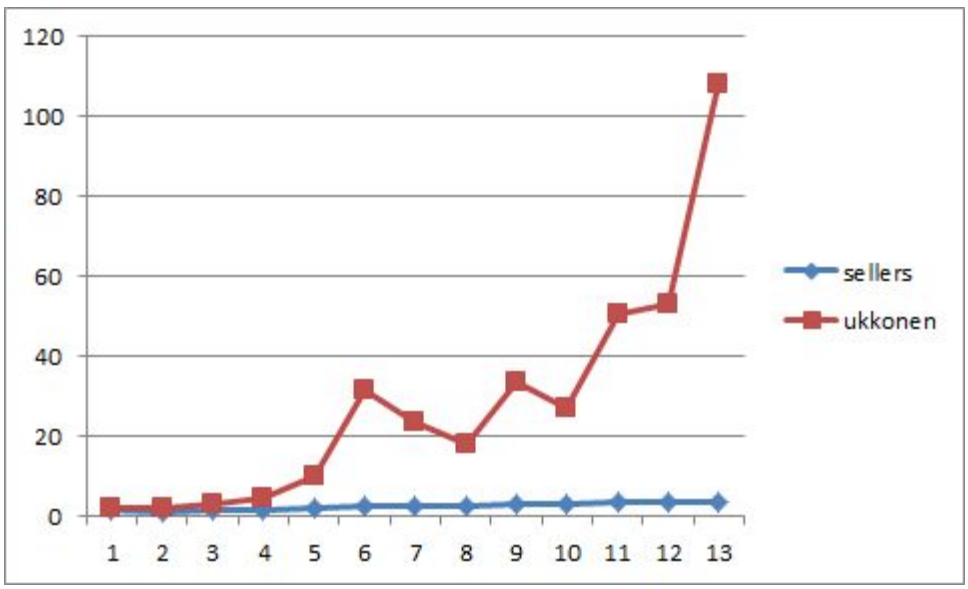
3.1 Teste com textos em inglês

Para a realização desse teste, utilizamos dois arquivos de texto do corpora do Pizza&Chili [2]. Para os padrões, foram utilizadas palavras em inglês variando de tamanho 1 a 13. Os eixos do gráfico representam o tamanho do padrão (horizontal) e o tempo levado pelo algoritmo (vertical).



Teste 1

No primeiro teste, foi usado um arquivo texto de 1Gb, e o conjunto de padrões utilizados foi $\Pi = \{a, of, the, book, write, normal, chapter, document, footnotes, processing, parentheses, governmental, reactionaries\}$.
O gráfico do teste 1 mostra que, para padrões pequenos, a ferramenta *grep* e o algoritmo de Boyer-Moore têm um desempenho fraco comparado com o KMP. Já para padrões mais longos, o *grep* apresenta um melhor desempenho que os dois algoritmos implementados na ferramenta *pmt*, mantendo-se abaixo dos 15 segundos em quase todos os padrões acima de 4 letras.
Diante da estrutura dos padrões, as construções dos algoritmos de Boyer-Moore e o KMP utilizam um pré-processamento que não é muito bem aproveitado, visto que as maiores bordas dos padrões são pequenas ou inexistentes, o que explica o porquê do seu desempenho melhorar conforme os padrões aumentam. O KMP, por possuir um comportamento estável durante todo o teste, apresentou-se como uma solução estável para padrões de vários tamanhos, o que reitera os resultados presentes na literatura.



Teste 2

Nesse teste, foi usado um arquivo de 50MB, com os mesmos padrões usados no teste anterior.
A distância máxima de edição utilizada para cada padrão de tamanho m foi $e_{max} = m - 1$.
Nesse exemplo, foi clara a vantagem do Sellers sobre o Ukkonen. Isso se deve ao fato de que o algoritmo Ukkonen utilizou de uma função de *hash* bastante ineficiente para a representação da função de transição. Esta limitação, somada ao fato de sua etapa de pré-processamento possuir custo exponencial, fez com que o Ukkonen possuísse desempenho inferior ao Sellers em todos os cenários testados.

4 Conclusão

A utilização de um algoritmo depende bastante da situação onde ele deverá ser empregado, mesmo entre algoritmos de custo computacional de mesma ordem de grandeza no caso médio. Uma grande dificuldade durante a implementação da ferramenta foi durante a otimização da memória utilizada para a leitura do arquivo de texto e de pré-processamento, pois não havia uma solução muito óbvia para conciliar essas duas etapas (pelo menos com os algoritmos de busca exata implementados). Isso poderia ser evitado se, por exemplo, o algoritmo de Aho-Corasick fosse utilizado para múltiplos padrões, já que ele fora projetado para essa finalidade.

Outro ponto importante a ser reiterado é a escolha das estruturas de dados ideais para a implementação de uma técnica. Pontos que poderiam ser explorados para melhorar o desempenho podem ser afetados caso uma escolha equivocada seja realizada, tal qual aconteceu com o algoritmo de Ukkonen.