IPMT

IF767 *(Processamento de Cadeias de Caracteres)*

Vinícius Carneiro Pereira Souza (vcps)

1. Implementação

A ferramenta foi escrita na linguagem de programação C++ (versão 11), contando com scripts de instalação e de execução de testes escritos em Bash e sendo voltada à plataforma Linux.

O programa principal está contido no arquivo *main.cpp*, o qual importa todos os demais módulos e permite a utilização de argumentos de linha de comando que alteram o funcionamento do programa. Tais argumentos são obtidos a partir do uso das funções contidas na biblioteca optionparser, a única biblioteca externa utilizada na ferramenta.

A ferramenta permite dois modos de operação: *indexação* e *busca*.

No modo *indexação* um arquivo de texto é fornecido, e um arquivo de índices (idx) é então construído e salvo no diretório atual. O arquivo de índices é binário, e é construído da seguinte maneira:

1. Para cada linha do arquivo de entrada, constrói-se o array de sufixos correspondente - usando uma implementação do algoritmo especificado aqui.
2. Executa-se compressão na linha de texto - usando uma implementação do algoritmo LZW.
3. Escreve-se então, no arquivo de índices, o conteúdo do array de sufixos, primeiramente a quantidade de elementos e então cada elemento do array.
4. Por fim, escreve-se a linha de texto comprimida, da mesma maneira que o array de sufixos: primeiramente a quantidade de códigos utilizada e depois cada um dos códigos que compõem a linha comprimida.

No modo *busca*, lê-se o arquivo de índices de duas em duas linhas (array de sufixos e linha comprimida), e procura-se por todos os padrões fornecidos - a busca é feita usando uma implementação do algoritmo de busca binária. Antes de se fazer a busca, porém, é necessário descomprimir o texto.

Após todos as ocorrências dos padrões terem sido encontradas, são imprimidas na tela as linhas do arquivo de texto original que contêm ocorrências ou apenas a quantidade de ocorrências, dependendo do parâmetro *--count* ter sido omitido na chamada da ferramenta ou não.

Opcionalmente, é possível passar a opção *--report-runtime* para a ferramenta, o que faz com que ela registre o tempo de execução da indexação/busca em cada linha de texto. Tal resultado pode ser encontrado no arquivo runtime*.csv*, ao final da execução.

Detalhes e Falhas

Atualmente a ferramenta só lida com linhas de texto com no máximo 65000 caracteres. Essa é uma limitação da implementação do algoritmo de construção dos arrays de sufixos e deve ser solucionada com o uso de uma estrutura de dados maior.

Outra limitação da ferramenta é o uso, também durante o algoritmo de construção dos arrays de sufixos, de uma implementação do algoritmo *quicksort*. Nessa situação seria mais eficiente fazer uso do algoritmo *radix sort*.

2. Testes e Resultados

Os dados utilizados durante o experimento foram gerados por um script próprio - data_gen.py -, enviado junto com a ferramenta. O script aceita os seguintes argumentos:

- ***line_size***: O número de caracteres que cada linha do texto gerado devia ter.
- ***dataset_name***: O nome do conjunto de dados sendo criado. Se a opção 'auto' for fornecida, o nome é criado a partir dos outros argumentos fornecidos.

O script gera um arquivo de padrões com 10 padrões de 100 caracteres, e então gera um arquivo de texto de 100 linhas, cada qual o número de caracteres especificado pelo argumento *line_size*. Em cada linha é colocada uma quantidade aleatória de padrões, e tais padrões também são escolhidos aleatoriamente, em posições aleatórias da linha. O resto dos caracteres da linha também são determinados aleatoriamente.

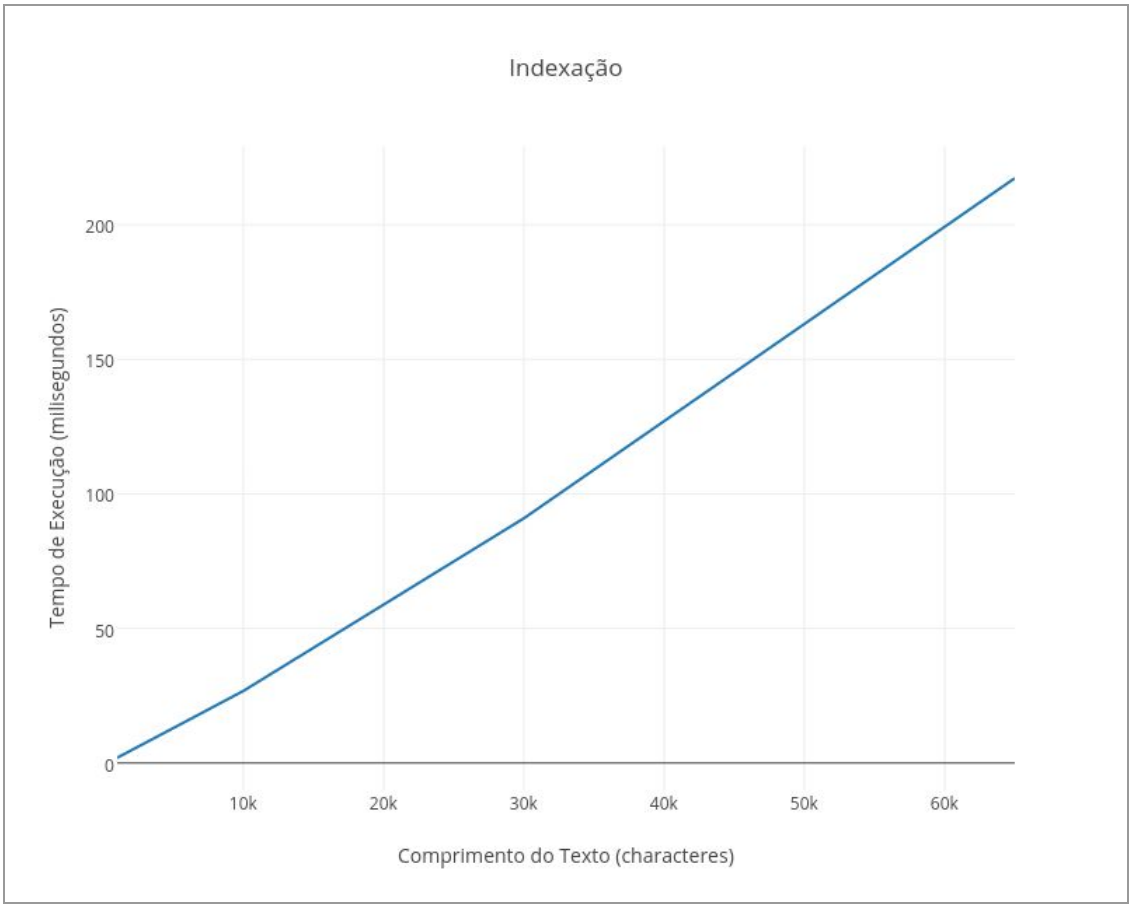
O ambiente de testes consistiu numa máquina pessoal com as seguintes configurações:

- Processador Intel(R) Core(TM) i7-3630QM CPU @ 2.40GHz
- Dois pentes de memória RAM DDR3, totalizando 6GB

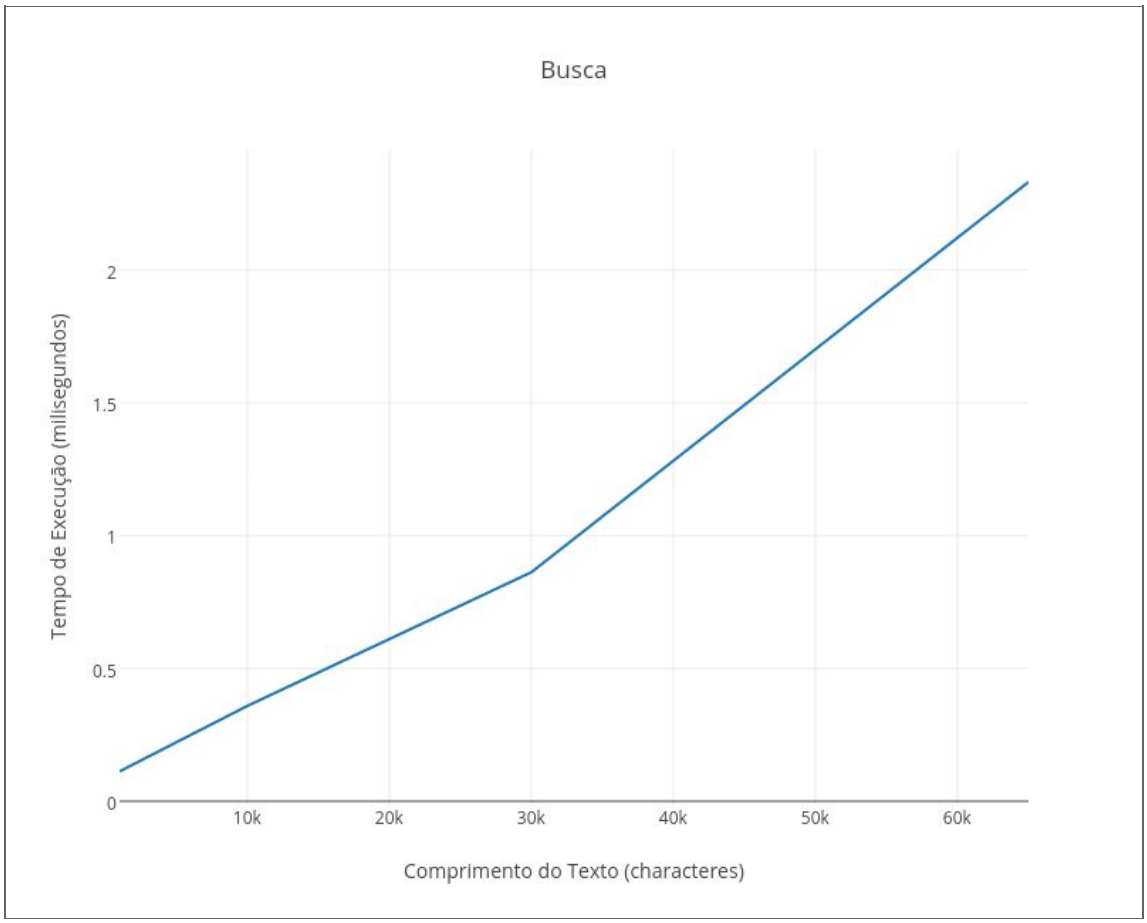
O experimento consistiu em gerar 4 conjuntos de dados, cada um com um tamanho de linha diferente:

Nome	**Número de caracteres por linha**
1000_ls	1000
10000_ls	10000
30000_ls	30000
65000_ls	65000

Abaixo seguem os dados obtidos:



Como esperado, um número maior de caracteres por linha acarreta um aumento do tempo de execução da indexação, principalmente pelo fato do array de sufixos a ser criado ter aumentado.



Novamente, por conta do aumento do número de caracteres por linha, temos um aumento do tempo de execução da busca, que precisa de mais repartições do algoritmo de busca binária para encontrar as ocorrências no array de sufixos dos padrões fornecidos.

Conclusão

Novamente, foi interessante constatar através da implementação e posterior experimentação a veracidade das afirmações feitas acerca dos algoritmos implementados em aula. Com a implementação de diferentes abordagens de indexação (árvores de sufixos) e de compressão (algoritmos da família LZ77, por exemplo) seria possível ter resultados mais interessantes.