IPMT – Indexed Pattern Matching Tool

Nome da ferramenta: IPMT

Grupo: Danilo Neves Ribeiro (dnr2)
Vitor Hugo Antero de Melo (vham)

Descrição do funcionamento da ferramenta

A ferramenta IPMT foi desenvolvida utilizando a linguagem de programação C++. Implementamos os seguintes algoritmos estudados em sala de aula que serão separados em duas categorias:

Algoritmo de indexação:

Árvore de Sufixos (Ukkonen)

As árvores de sufixo, baseada no algoritmo de Ukkonen [1] como vistas em sala, são mais eficientes na complexidade de tempo para construção de índices. Utilizamos esta opção para fazer a indexação do texto e ela se mostrou eficiente. Em [2] há um detalhamento da construção da árvore, e nosso algoritmo foi baseado neste site.

Algoritmo de compressão/descompressão:

LZW

Para realizar a compressão e descompressão dos arquivos e das estruturas de dados, utilizamos o algoritmo de Lempel-Ziv-Welch (LZW), que é comumente utilizado em compressão de arquivos GIF. Nós seguimos as dicas de implementação encontradas em [3]. Utilizamos a estrutura *map* para o dicionário tanto na compressão quanto na descompressão.

Detalhes de implementação

Os arquivos de entrada são carregados completos na memória, em uma única string. Decidimos utilizar essa estratégia pois é necessário varrer a string de entrada para criar a árvore de sufixo de uma única vez.

A implementação do projeto segue a estrutura descrita na especificação. Os arquivos do código fonte se encontram na pasta *src/*, descritos a seguir:

ipmt.cpp

Este é o arquivo de interface com o usuário, ele importa um segundo arquivo que contém a implementação dos algoritmos, *ipmt_algorithm.h*. Seus objetivos incluem imprimir a ajuda, fazer *parse* nos argumentos de entrada, lidar com *wildcards* etc. Ele também verifica se a execução deve ser do tipo *index* ou *search*, e chama as funções correspondentes.

ipmt_algorithm.h

Define algumas constantes e as assinaturas dos métodos que são implementados em *ipmt_algorithm.cpp*.

ipmt_algorithm.cpp

Este arquivo contém os algoritmos de compressão e descompressão, e a lógica que chama as funções para criar índices e executar buscas. Implementa as funções definidas em *ipmt_algorithm.h*, onde as principais funções são *ipmt_index_*()* e *ipmt_search()*, que servem de ligação entre a interface com o usuário (*ipmt.cpp*) e suas funções.

SuffixTree.cpp

Por questões de organização foi decidido criar uma estrutura para comportar todas as variáveis que definem a árvore de sufixos, bem como funções e variáveis que ajudam a construí-la. Caso essa estrutura não fosse criada seria necessário passar várias estruturas e *arrays* por parâmetro, ou fazer tudo em um único método, o que avaliamos inadequado. A estrutura da arvore foi representada por um array de arestas, já que desta forma seria mais direto a codificação e descodificação. Além disso foi utilizado a estrutura map de c++ para mapear quais as arestas que saiam de um determinado nó, considerando um determinado caractere (esta estrutura é faz busca em tempo logarítmico, talvez usar um hashmap aumentaria a parformance). Para serializar a arvore criamos uma string, onde o primeiro inteiro representa o numero de nós na arvore, e os demais n-1 inteiros representam as arestas (com informação do no inicial, no final label da aresta).

Bugs e limitações

Observamos que para muitos arquivos pequenos a compressão fica maior do que o arquivo original. Isso ocorre devido à baixa quantidade de padrões repetidos. Outra limitação ocorre no tamanho do dicionário não ser limitado, o que pode causar estouro de pilha dependendo do tamanho da entrada e da memória disponível durante a execução.

Outro bug foi que quando compilado no Linux a árvore joga um exceção na hora de dar free na memoria alocada. Este erro não ocorre no Windows (sistema operacional que utilizamos para fazer o código). Para solucionar o problema tivemos que forçar o encerramento

do programa. Isso não afeta o resultado final, contudo não é possível indexar mais de um arquivo de texto na mesma execução.

Referências

[1] - E. Ukkonen. On-line construction of suffix trees. Algorithmica, 14(3):249-260, September 1995.
[2] - http://marknelson.us/1996/08/01/suffix-trees/
[3] - https://www.cs.duke.edu/csed/curious/compression/lzw.html