Universidade Federal de Viçosa
Campus Florestal
CCF 212 - Algoritmos e Estruturas de Dados II
Prof.ª Gláucia Braga e Silva

Trabalho Prático 1 - Algoritmos e Estruturas de Dados II

Poocle Search Tool

Alan Araújo dos Reis - 5096
Gabriel Rodrigues Marques - 5097
Luiz César Galvão Lima - 4216
Marcos Biscotto de Oliveira - 4236

Florestal - MG

2023

Sumário

1. Introdução

Este Trabalho Prático (TP) tem como objetivo a elaboração de um programa que funcione como um mecanismo de busca para os Projetos Orientados em Computação (POC's) do curso de Ciência da Computação da Universidade Federal de Viçosa - Campus Florestal. Para tal objetivo, o programa foi construído utilizando-se da árvore PATRICIA para armazenar o conteúdo textual presente nos POC's e a técnica de índices invertidos para cálculo de peso das palavras e relevância dos documentos.

2. Organização

Em relação à organização do repositório do projeto, consideramos deixá-lo da forma mais organizada possível. Para isso utilizamos pastas para separar os diversos arquivos presentes no TP.

- *docs*: possui os arquivos *".pdf"* da especificação do TP juntamente com essa documentação e o arquivo de imagem com o nome do programa.

- *files*: contém o arquivo de entrada e outros quinze arquivos de texto relativos aos POC's, que são compostos pelo título, resumo e palavras chaves (quando houver). Existe também um arquivo de texto que junta todos os quinze arquivos em um único arquivo, para fim de consulta e comparação com os demais arquivos. Nesta pasta temos também um arquivo Python utilizado para realizar a manipulação automatizada dos outros arquivos.

- *"headers"* e *"sources"*: contém respectivamente os arquivos *".h"* e *".c"* utilizados para a construção do programa.

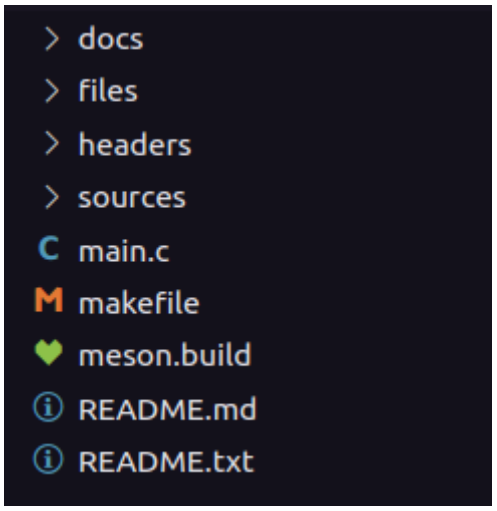

Imagem 1: Organização do Projeto

3. Desenvolvimento

Inicialmente o grupo buscou se informar da melhor maneira de implementar os algoritmos necessários para o trabalho, indo para as monitorias e conversando com alunos que já fizeram a disciplina. Tendo obtido essas informações, Alan e Gabriel ficaram responsáveis pela implementação da manipulação de arquivos, índices, worlde e PATRICIA, Marcos com a interface gráfica através do GIMP Tool Kit e o Luiz com os cálculos de relevâncias, cada qual sendo aberto sobre a situação/desenvolvimento de sua parte para o grupo. Caso alguém encontrasse algum problema e não conseguisse solucionar, foi definido que buscasse ajuda de outros membros do grupo, contribuindo para cooperação. Também, quando necessário, buscamos ajuda de outros grupos ou de um aluno com conhecimento no problema (monitores incluso). Foi utilizado o GitHub para versionamento de código e como meio de compartilhar o mesmo e o Discord para as reuniões.

3.1 Estrutura do Poocle Search Tool

Como solicitado, tentamos encapsular as funções sempre que possível. Portanto, cada parte do código pôde ser testada separadamente antes de ser integrada ao programa, facilitando as implementações. Dividimos o TP em 10 TAD's cada qual com suas estruturas e funções próprias: *docList, index, wordle, patTree, fileManagement, math, searchEngineTasks, poocleMenu* e *GTKInterface*. A árvore PATRICIA, o manejador de arquivos, os cálculos de relevância e o menu serão abordados em tópicos separadas, pois compõe a parte mais importante do programa.

- *include*: aqui estão presentes todas as bibliotecas da linguagem utilizadas para elaboração do programa, além da definição para valores inteiros e para cores utilizadas para personalização.

- *docList*: por meio da estrutura lista encadeada implementamos uma lista de documentos que contém toda a coleção de documentos que foi informada no arquivo de entrada. Na estrutura do documento estão armazenadas informações importantes que serão utilizadas em outros campos do programa: *docName* (nome do documento), *idDoc* (identificador), *rDoc* (relevância) e *nWordle* (número de palavras distintas). Além disso, existem funções para pesquisar por um documento com base em suas informações.

- *index*: também utilizando de listas encadeadas foi implementado o índice invertido que estará contido dentro de cada palavra. O índice é estruturado da forma *<qtde, idDoc>*, que indicam respectivamente quantidade de palavras e qual é o documento referente.

- *wordle*: no sentido de organizar melhor o programa, criamos um TAD que integra palavra e a lista de índice invertido, juntamente com suas funções. A estrutura é formada por *wordChar* e *indexList*.

- *searchEngineTasks*: aqui foram implementadas as funções que serão chamadas durante a execução do programa: receber o arquivo de entrada, construir o índice invertido, imprimir índice invertido e pesquisar. Essas funções chamam outras funções que foram implementadas nos outros TAD's.

3.2 Árvore PATRICIA

A árvore PATRICIA foi implementada a partir dos códigos do Nivio Ziviani que foram disponibilizados, fazendo as devidas alterações para que fosse possível armazenar palavras. A estrutura do nó interno continua com os apontadores para direita e esquerda, armazenando a maior letra que difere entre as palavras e o índice onde elas se diferem. No nó externo está localizado a palavra e o índice invertido da mesma, unidos pelo tipo *wordle*, que foi detalhado anteriormente.

Nesse sentido, decidimos escolher a maior letra para evitar que o *'\0'* fosse armazenado dentro do nó interno quando fossem comparadas palavras de tamanhos diferentes. Essa decisão foi tomada baseada no fato de que utilizando a menor letra o *'\0'* iria colocar todas as palavras maiores a direita e as menores a esquerda, gerando problemas de ordenação lexicográfica e comprometendo o próprio balanceamento da árvore. Chegamos a essa conclusão em conjunto com o Grupo 8, após uma discussão sobre este determinado assunto.

Outro ponto, é que na inserção da PATRICIA realizamos a contabilização das palavras que estão sendo inseridas. Como parâmetro passamos a palavra em si e o identificador do documento de origem. Desse modo, conseguimos contabilizar a quantidade daquela palavra no documento específico e também contabilizar as palavras únicas no documento. Essa decisão foi tomada para que futuramente o cálculo seja feito de maneira mais fácil. As funções implementadas na lista de documentos e na lista de índice invertido permitem que isso seja feito por meio das funções de busca e verificação.

3.3 Manejador de Arquivos

A princípio foi implementado um programa na linguagem Python para facilitar a leitura dos arquivos. Isso poderia ser feito manualmente, mas decidimos automatizar esse processo para facilitar a coleta do conteúdo textual dos POC's. Dessa maneira, basta você copiar todo o conteúdo escrito do PDF e colocá-lo dentro de um arquivo *.txt*. O programa, a partir do arquivo de entrada previamente formatado e informado de maneira correta, removerá qualquer caractere não alfanumérico presente no corpo do texto através de expressões regulares. Além disso, foi recomendado que stopwords também fossem removidas dos documentos,

isso foi feito por meio da biblioteca NLTK (Natural Language Toolkit). A aplicação não é vital para o funcionamento do programa, mas facilita o processo.

No programa são realizadas as operações de leitura e escrita. Primeiramente é realizado o processo de receber o arquivo de entrada, suas informações com o nome dos documentos são lidas e armazenadas sequencialmente na lista de documentos. É importante frisar que para o funcionamento correto do programa o arquivo de entrada seja preenchido corretamente, modificando caso algum documento seja inserido ou removido. Em seguida, é realizada a construção do índice invertido por meio da inserção na árvore PATRICIA utilizando as informações dos documentos na lista de documentos.

3.4 Cálculo de Relevância

Foi utilizado como métrica para cálculo de relevância de cada documento a técnica TF-IDF (Term Frequency - Inverse Document Frequency), que é um cálculo que avalia o quão relevante é um documento dentro de uma coleção de documentos dado um termo ou conjunto de termos. O cálculo foi demonstrado na especificação do trabalho, a partir disso nós o adaptamos para funcionar de acordo com a implementação do projeto. O ćalculo é feito sequencialmente por documento, verificando primeiramente se aquele termo existe dentro da árvore PATRICIA e, em seguida, se o termo apareceu naquele documento que está sendo feito o cálculo. Após essa verificação, são feitos os devidos cálculos de peso do termo e relevância do documento.

Nessa etapa também foi implementado a parte de imprimir os arquivos por relevância. Inicialmente, pensamos em ordenar a lista de documentos, o que acabou se tornado um problema. Alternativamente, estamos armazenando as relevâncias em um array e fazendo a ordenação do mesmo com uma função própria da linguagem. Em seguida, pesquisamos os documentos por relevância e os imprimimos. Dessa forma, preservamos a ordem da lista de documentos.

3.5 Menu Interativo

Para que consiga executar o buscador com todas as suas funcionalidades, será apresentado ao usuário um menu interativo com as funções e ferramentas necessárias para interagir com o programa. Foram feitas duas versões, uma em terminal para idealizar o projeto e outra definitiva utilizando o GTK como foi solicitado.

Para realizar a implementação do menu interativo, inicialmente foi tentado utilizar a ferramenta de design de interface de usuário Glade juntamente com a biblioteca GTK da linguagem C. Porém, devido aos inúmeros problemas referentes à manipulação das estruturas e passagens de parâmetros para o menu, principalmente devido a nossa falta de conhecimento da biblioteca e do software Glade, optamos por não utilizar tal ferramenta. Assim sendo, o menu foi construído utilizando apenas os códigos da biblioteca GTK com o auxílio do Chat GPT quando

nos deparávamos com questões específicas da biblioteca e que não constavam na documentação. Deixamos as funções do menu em arquivos .c e .h separados do restante do código para melhor organização. As funções que estavam relacionadas a exibir e pesquisar precisaram ser adaptadas para o funcionamento na interface gráfica, mas ainda mantêm a mesma lógica e utilizam de funções que foram anteriormente implementadas.

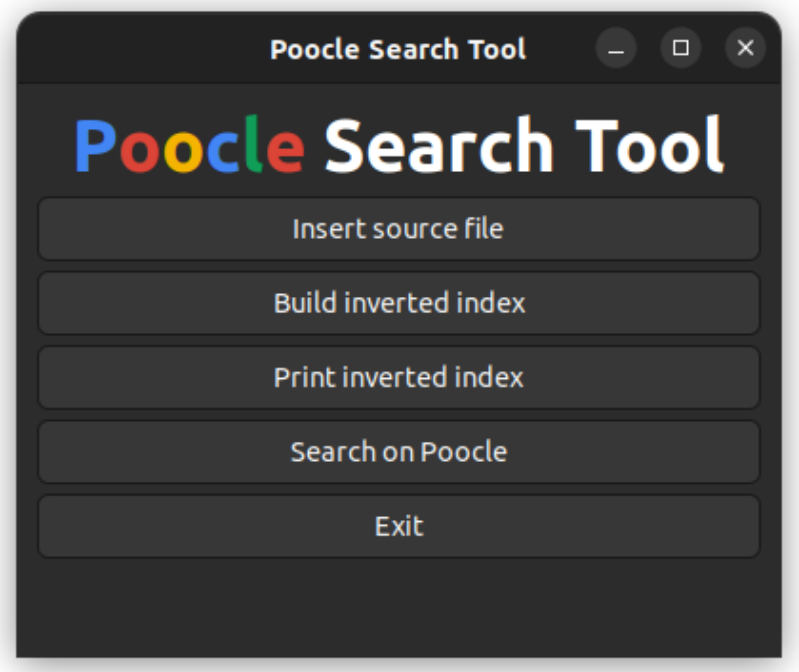

Imagem 3: Interface principal do Menu Interativo

3.6 Makefile e Meson Build

Para a melhor utilização da biblioteca GTK, criamos um arquivo intitulado *“meson.build”* que possibilita e facilita a implementação dos comandos da biblioteca e passa as dependências da execução do programa para a interface. A utilização do Meson Build substitui de certa forma o arquivo *“makefile”*, porém como na especificação do trabalho foi solicitado a presença desse arquivo, fizemos um *“makefile”* que executa os comandos de compilação do *“meson.build”* que por sua vez executará o restante do projeto.

4. Resultados

Como resultado, obtivemos um programa com menu interativo que disponibiliza opções ao usuário para exercer as funções requisitadas. Assim como visto nas imagens abaixo, conforme o usuário clique nos botões do menu, as funções vão sendo executadas de modo com que o usuário não veja esse processo acontecendo.

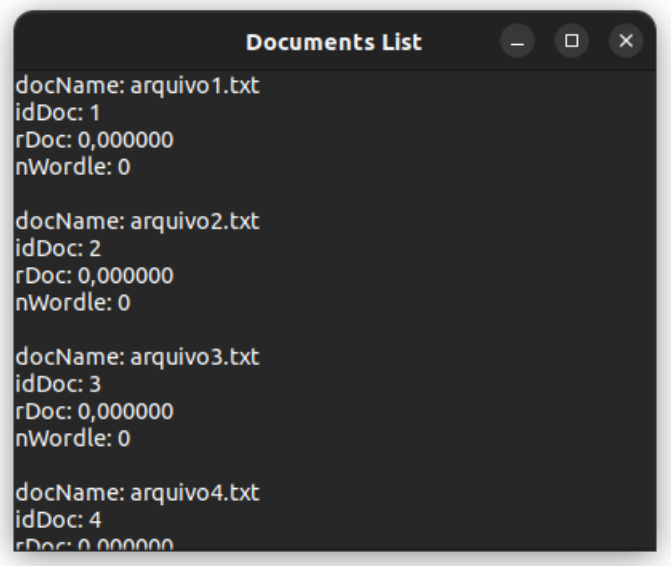

Imagem 4: Opção “Insert source file”

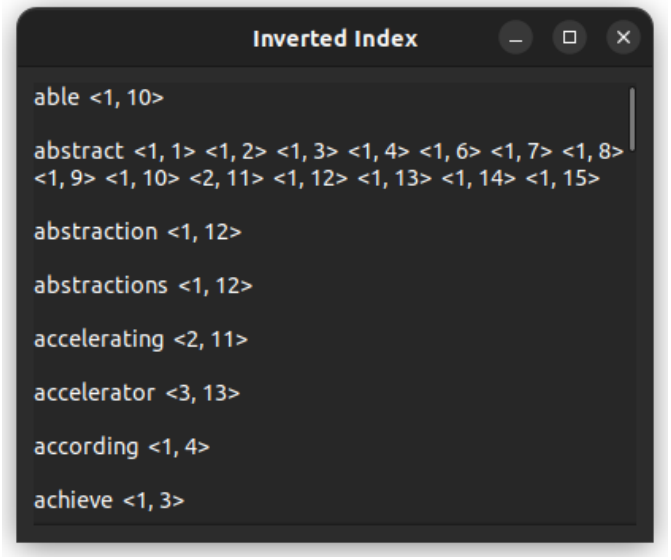

Imagem 5: Opção “Print inverted index”

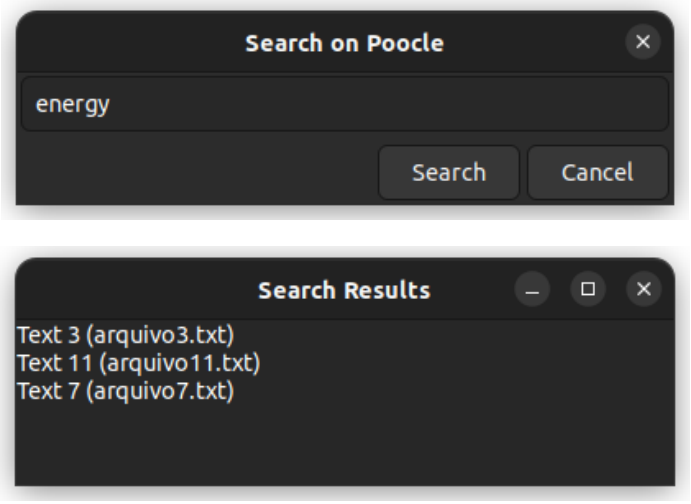

Imagem 6: Opção “Search on Poocle”

5. Considerações Finais

Por fim, ficamos bastante satisfeitos com os resultados obtidos do programa Poocle Search Tool, conseguindo implementar tudo que idealizamos ao longo do processo. Com o término do trabalho, acreditamos que entregamos uma aplicação, que ao nosso ver, atende aos requisitos anteriormente solicitados na especificação. A maior dificuldade foi em relação a PATRICIA e o GTK, mas com o auxílio de monitores e outros grupos foi possível concluir sua implementação. Além disso, a nossa experiência prévia com um trabalho semelhante na disciplina Algoritmos e Estruturas de Dados I possibilitou um melhor entendimento do processo de execução. Ademais, o trabalho trouxe novos conhecimentos, como por exemplo a técnica de estatística de índice invertido e seus cálculos e a implementação de uma árvore digital, contribuindo também para o melhor entendimento do funcionamento de buscadores que utilizamos no dia a dia.

6. Referências

[1] Github. Disponível em: <https://github.com/gabbrielol/PoocleSearchTool> Último acesso em: 29 de Maio de 2023.

[2] N. Ziviani, Projeto de Algoritmos com Implementações em Pascal e C - 3ª ed., Cengage Learning, 2010. Disponível em: <https://www.dcc.ufmg.br/~nivio/> Último acesso em: 23 de Maio de 2023.

[3] Python. Disponível em: <https://www.python.org> Último acesso em: 28 de Maio de 2023.

[4] Regular Expression. Disponível em <https://docs.python.org/pt-br/3/library/re.html> Último acesso em: 28 de Maio de 2023.

[5] Natural Language Toolkit (NLTK). Disponível <https://www.nltk.org> Último acesso em: 28 de Maio de 2023.

[6] GIMP Tool Kit. Disponível <https://www.gtk.org> Último acesso em: 26 de Maio de 2023.

[7] Meson. Disponível em: <https://mesonbuild.com> Último acesso em: 26 de Maio de 2023.

[8] Term frequency – Inverse Document Frequency. Disponível em: <https://www.geeksforgeeks.org/understanding-tf-idf-term-frequency-inverse-document-frequency/> Último acesso em: 17 de Maio de 2023.

[9] Chat GPT. Disponível em: <https://chat.openai.com> Último acesso em: 26 de Maio de 2023.