CCF 212 - Algoritmos e Estruturas de Dados II Trabalho Prático 01 - Árvores Digitais

Artur Papa - 3886, Vinicius Mendes - 3881, Jhonata Miranda - 3859

Setembro 2021

1 Introdução

O trabalho em questão tem como objetivo abordar os conteúdos trabalhados na disciplina de Algoritmos e Estrutura de Dados II, em específico, como visto nas aulas teóricas e práticas, a inserção e busca de elementos utilizando árvores digitais. A priori, os métodos de pesquisa digital são particularmente vantajosos quando as chaves são grandes e de tamanho variável. Um aspecto interessante quanto aos métodos de pesquisa digital é a possibilidade de localizar todas as ocorrências de determinada cadeia em um texto, com tempo de resposta logarítmico em relação ao tamanho do texto [2].

Além disso, cabe ressaltar que foi utilizado nesse trabalho dois tipos de árvores digitais, a TST(Ternary Search Tree) e a PATRICIA(Practical Algorithm to Retrieve Information Coded In Alphanumeric). Vale notar que a primeira se trata de um tipo especial de estrutura de dados Trie e é amplamente utilizada como uma alternativa de pouca memória para testar em uma vasta gama de aplicações, como verificação ortográfica e pesquisa em sites. Já a segunda se trata de uma representação compacta de uma Trie onde os nós que teriam apenas um filho são agrupados nos seus antecessores [1].

2 Desenvolvimento

2.1 Execução e Organização das Tarefas

De início, optamos por desenvolver a implementação da Árvore PATRICIA que possuía a capacidade de inserir palavras ao invés de caracteres, como foi demonstrado em sala. Essa decisão foi tomada pois acreditavamos que seria a parte essencial do projeto e a que poderia nos gerar mais problemas. Baseamos na implementação do Professor Ziviani, visto que, tinhamos mais material disponível para consulta, além de maior familiaridade com os algoritmos implementados por ele na matéria de Algoritmo e Estrutura de Dados I.

Posteriormente, pela necessidade da leitura de um arquivo molde e da busca por um item dentro da árvore, foi priorizado o desenvolvimento das funções de pesquisa e leitura de arquivo. No primeiro caso, percorremos a PATRICIA comparando uma palavra passada pelo usuário com uma chave encontrada dentro da estrutura "TipoPatNo". Já no segundo, utilizamos o "fscanf"para ler o arquivo e salvar em um vetor de char, a biblioteca ctype.h, que possui funçoes essenciais para o desenvolvimento do código, transformando caracteres maiúsculos em minusculos, identificando tipos de caracteres, entre outros.

Depois da árvore PATRICIA, optamos por implementar a Árvore Trie do tipo TST, que por sua vez é utilizada na função de autopreenchimento. Para esse ponto, utilizamos uma lista de palavras em português como dicionário, que foi salva dentro da pasta ***data***. A TST insere todas as palavras e o autopreenchimento é responsável por buscar os prefixos passados e retornar as possíveis palavras disponíveis.

2.2 Implementação da PATRICIA

Por conseguinte, tivemos que pensar em como adaptar os algoritmos da árvore PATRICIA para inserir palavras ao invés de apenas armazenar caracteres. A princípio, tivemos a ideia de fazer uma árvore compressa, em que os nós iriam armazenar os termos ao invés de apenas uma letra, entretanto não encontramos muitos materiais sobre esta forma de implementação. Dessa maneira, foi decidido apenas que seria mudado a comparação bit a bit para valores inteiros.

```c
int diferenca(tipoChave k, tipoChave r)
{
    int count = 0;
    while (k[count] != '\0' || r[count] != '\0')
    {
        if (k[count] != r[count])
            return count;
        else
            count++;
    }
    return count;
}
```

Figura 1: Função diferença da PATRICIA

Assim, com essa função mostrada acima conseguimos descobrir em qual posição se encontra o caracter diferente da palavra que estamos inserindo com a que já está na árvore. Além disso, usamos uma letra de desvio para decidir se tal chave seria inserida pela esquerda ou pela direita, desta maneira, caso o índice do termo a ser inserido seja maior do que a letra de desvio, chamamos a função insereEntre recursivamente pela direita, caso contrário, inserimos recursivamente pela esquerda.

```c
else
{
    if (k[(*t)->NO.NInterno.indice] > (*t)->NO.NInterno.desvio)
        (*t)->NO.NInterno.dir = insereEntre(k, &(*t)->NO.NInterno.dir, i, d);
    else
        (*t)->NO.NInterno.esq = insereEntre(k, &(*t)->NO.NInterno.esq, i, d);
    return (*t);
}
```

Figura 2: Chamada recursiva para inserir

Vale ressaltar que a mesma lógica foi utilizada para fazer a busca de uma chave que está na PATRICIA.

2.3 Considerações finais

2.3.1 Dificuldades e aprendizado

Nossa dificuldade inicial consistiu no desenvolvimento de uma forma de ler aquivos eficientes, pois, de início tentamos ler diversos arquivos ao mesmo tempo, o que não obteve um bom

resultado, sobrepondo alguns itens. Essa mesma dificuldade foi responsável por diversos outros erros em outros pontos essenciais para a evolução do projeto, influênciando diretamente no indíce invertido, em que, pela leitura estar defeituosa, a contagem e identificação das palavras tornou-se extremamente tabalhosa.

Adiante, enfrentamos alguns problemas com nossos Sistemas Operacionais ao tentar instalar a biblioteca GTK. Ao tentar baixá-la, o Linux de um integrante do grupo corrompeu e perdeu todos os arquivos que já possuía. Outro membro buscou outras formas de realizar esse passo, utilizando, inclusive de um pen-drive, já que não tinha memória suficiente em seu notebook.

Outrossim, obtemos certa experiência ao trabalhar com os tipos de árvores desenvolvidos em sala de aula, aperfeiçoamos o desenvolvimento de projetos na linguagem C e, acima de tudo, a capacidade de solução de empecilhos em equipe.

3 Conclusão

Posto isso, conclui-se que o trabalho em questão foi desenvolvido arduamente, atingindo algumas especificações requeridas na descrição do mesmo em construir o índice invertido para máquinas de busca.

Por conseguinte podemos ressaltar que o livro-texto da disciplina [2] foi de suma importância para o desenvolvimento do projeto, haja vista que as explicações dadas pelo autor nos ajudaram bastante a entender as etapas a serem executadas, a construção dos códigos e a disponibilização do algoritmo da PATRICIA. Além disso podemos ressaltar que vídeos como os disponibilizados na disciplina de Algoritmos e Estrutura de Dados II, bem como a interação no fórum foram de grande ajuda para o entendimento dos conceitos necessários para o desenvolvimento do projeto.

Em adição, é válido dizer que apesar das dificuldades na implementação do código, o trio foi capaz de superar e corrigir alguns erros no desenvolvimento do algoritmo. Por fim, verificou-se a assertiva para o objetivo do projeto em implementar os métodos de pesquisa digital utilizando a árvore TST e PATRICIA.

Referências

[1] digital:`https://iq.opengenus.org/ternary-search-tree/`.

[2] Nivio Ziviani. *Projeto de Algoritmos com implementações em Pascal e C*, volume 3. 2010.