TRABALHO FINAL
Gerador de Paráfrases

1 Objetivo

O objetivo do trabalho é comparar o desempenho das Árvores Binárias de Pesquisa e das árvores balanceadas (AVL, Rubro-Negra, ou Splay) em uma aplicação de geração de paráfrases.

2 Especificação da Aplicação

A geração de paráfrases consiste em reformular um texto trocando as palavras por outras de mesmo significado. Por exemplo, a sentença "O sujeito criminoso cometeu um roubo." pode ser parafraseada para "O indivíduo delitoso cometeu um assalto." A **geração automática de paráfrases** é um tema importante e desafiador na área de **Processamento de Linguagem Natural**, uma área de pesquisa da Ciência da Computação que tem o objetivo de fazer com que os computadores consigam processar/compreender a linguagem humana (falada e escrita).

A geração automática de paráfrases tem várias aplicações atuais como aumento de dados (*data augmentation*) para algoritmos de aprendizado de máquina, reescrita de consultas em sistemas de perguntas e respostas (*question answering*), recuperação de informação, sumarização de textos, entre outras.

As soluções do estado da arte para a geração automática de paráfrases são baseadas em algoritmos de deep learning. Contudo, para o nosso trabalho, utilizaremos uma abordagem simples baseada em um **dicionário de sinônimos**.

★ Sua tarefa é projetar uma aplicação que recebe um arquivo com um texto de entrada e um dicionário de sinônimos e gera uma versão parafraseada do texto de entrada. O dicionário deverá ficar armazenado em uma árvore. Você deve **criar duas versões da aplicação** cada uma utilizando utilizando uma as **árvores** vistas em aula (ABP, AVL, Rubro-Negra ou Splay).

★ A ordem lexicográfica das palavras determinará a organização da árvore, *i.e.*, a ordem em que aparecem no dicionário. Por exemplo, se a palavra *"feliz"* for a raiz da árvore, então a palavra *"bonito"* deve aparecer na subárvore esquerda da raiz (utilize a função `strcmp`).

★ O dicionário deve ser carregado na árvore na ordem em que as palavras estão no arquivo.

★ A aplicação **não é case sensitive,** letras maiúsculas e minúsculas devem ser consideradas iguais.

★ O texto de entrada não deve ser armazenado na árvore. Cada palavra do texto será usada apenas para consultar a árvore e encontrar o sinônimo.

★ Palavras que não constam no dicionário devem ser gravadas sem alterações no arquivo de saída.

★ Seu programa deverá ser chamado a partir da linha de comando (passando parâmetros para o main).

Exemplo de chamada: `C:\minhaaplicacao dicionario.txt entrada.txt saida.txt`

As entradas e saídas da sua aplicação são:

Entradas:

(i) o nome do arquivo com o dicionário de sinônimos e

(ii) o nome arquivo com o texto de entrada.

Saídas:

(i) arquivo de saída com o texto parafraseado e

(ii) estatísticas sobre o processo de geração da árvore e realização de consultas (número de nodos, altura da árvore, número de rotações e número de comparações realizadas durante as consultas).

★ O número de **comparações** realizadas com os elementos da árvore **obrigatoriamente** será calculado pela função a seguir (onde `comp` é uma variável global que acumula o número de comparações).

```
pNodoA* consulta(pNodoA *a, char *chave)
{
    while (a!=NULL)
    {
        comp++;
        if (!strcmp(a->info,chave) )
        {
            comp++;
            return a;
        }
        else
        {
            comp++;
            if (strcmp(a->info,chave)>0)
                a = a->esq;
            else
                a = a->dir;
        }
    }
    return NULL;
}
```

★ Arquivos de teste serão disponibilizados no Moodle. No dia da apresentação, novos conjuntos de arquivos serão fornecidos.

2 Requisitos

- É necessário elaborar um relatório **detalhado** com a análise comparativa do desempenho das duas árvores.
- Crie **diferentes versões** do dicionário variando o número de palavras e a ordenação (em ordem ou fora de ordem) para embasar as análises realizadas.
- Utilize recursos como **tabelas** e **gráficos** para dar suporte às suas conclusões.
- O trabalho deve ser feito, preferencialmente, em duplas. Também aceitaremos trabalhos feitos individualmente e de duplas cujos integrantes sejam de turmas diferentes.
- A linguagem de programação aceita é C (Não é C++ nem C#).

3. Entrega e Apresentação

- **13 de janeiro de 2025** apresentação **em aula presencial** e entrega pelo Moodle.

4. Critérios de Avaliação

O trabalho deve ser realizado preferencialmente em **duplas** mas trabalhos individuais também são aceitos. Todos os trabalhos devem ser apresentados por **ambos** os membros da dupla na data prevista.

Para a avaliação serão adotados diversos critérios:

- funcionamento (Peso: 30%);
- organização e documentação do código (Peso: 30%); e
- relatório (Peso: 40%).

Importante:

- Este trabalho deverá representar a solução da dupla para o problema proposto. O plágio é terminantemente proibido e a sua detecção irá zerar a nota do trabalho.
- É permitido reutilizar códigos dos slides das aulas.
- É permitido reutilizar códigos encontrados na web desde que devidamente refenciados.