TRABALHO FINAL
Buscador de palavras-chave

1 Objetivo

O objetivo do trabalho é comparar o desempenho de diferentes estruturas de dados vistas na disciplina em uma aplicação de busca de palavras em tweets. Especificamente, a aplicação deve envolver pelo menos um tipo de árvore binária balanceada (AVL, Rubro-Negra ou Splay).

2 Especificação da Aplicação

Uma tarefa crucial na área de recuperação de informações é recuperar documentos que possuam as palavras-chave especificadas em uma consulta. A fim de resolver as consultas, os documentos precisam ser primeiro indexados. A tarefa a ser desenvolvida como trabalho final da disciplina simula, em pequena escala, o funcionamento de ***um buscador de palavras-chave***. A aplicação desenvolvida deverá ser composta por dois módulos: *indexação* e *consulta*. Durante a indexação, as palavras do texto dado como entrada serão carregadas em uma estrutura de dados, de forma que se possa identificar o documento (no caso o tweet) em que cada palavra ocorre. Durante a fase de consultas, a aplicação irá ler um arquivo com as palavras a serem consultadas e gerar um arquivo de saída com as palavras consultadas e os tweets nos quais elas ocorrem.

Uma **palavra** é uma sequência de letras. Todos os outros caracteres deverão ser considerados como separadores de palavras. Diferenças entre letras maiúsculas e minúsculas devem ser desprezadas (ex: a = A).

- Fase de indexação:
  - Entrada: Arquivo texto com o id do tweet (numérico) e seu texto a ser indexado. O formato das linhas é `id;tweet`.
  - Ao final da indexação, as palavras do texto juntamente com os ids dos tweets nos quais elas aparecem devem estar carregados na estrutura de dados e as estatísticas do processo devem estar computadas (número de nodos, número de comparações, número de rotações e altura da árvore). Veja o exemplo na Seção 3.
- Fase de consultas
  - Entrada: Arquivo texto com as palavras a serem consultadas (uma palavra por linha).
- Saídas: O arquivo de saída é composto por três partes (ver exemplo na Seção 3).
  - (i) as palavras consultadas e a lista de ids de tweets nos quais a palavra ocorre;
  - (ii) estatísticas da estrutura criada durante a fase de indexação (número de nodos, número de comparações, número de rotações e altura da árvore);
  - (iii) estatísticas da busca (número de comparações).

★ É necessário preparar **duas versões do trabalho** – sendo uma obrigatoriamente utilizando uma árvore binária balanceada (AVL, Rubro-Negra ou Splay). A outra versão pode ser com outra árvore (balanceada ou não) ou com listas encadeadas. Exemplos de comparações possíveis: AVL e ABP, Splay e LDE, AVL e Rubro-Negra, etc.

★ A ordem lexicográfica das palavras determinará a organização da estrutura de dados, *i.e.*, uma *p* palavra estará na subárvore esquerda de uma palavra *q* se ela aparecer antes de *p* na ordem lexicográfica (utilize a função `strcmp`).

★ Os textos dos tweets **não** devem ser armazenados em memória. Cada palavra do texto será usada apenas para consultar a estrutura para encontrar o os tweets em que ela ocorre.

★ A indexação consiste em percorrer o texto dos tweets, cada vez que uma palavra nova é encontrada, um novo nodo é gerado. Quando uma palavra já está na estrutura, basta acrescentar o id do tweet na lista de ocorrências da palavra. Como não sabemos em quantos documentos uma palavra aparece, a lista de ocorrências deve obrigatoriamente ser **uma lista encadeada**.

★ Mesmo que uma palavra ocorra múltiplas vezes no mesmo tweet, cada id deve aparecer **apenas uma vez** por consulta no arquivo de resultado.

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

Exemplo de chamada: `C:\minhaaplicacao entrada.txt consultas.txt saida.txt`
significa que é necessário indexar o texto de nome entrada.txt e a seguir processar as consultas do arquivo consulta.txt. O resultado será armazenado no arquivo saida.txt.
Há um código exemplo no Moodle que mostra como passar os parâmetros para a main e tokenizar o arquivo da entrada.

★ A eficiência da sua solução será medida pelo número de **rotações e comparações** realizadas com os elementos da estrutura. Por exemplo, para contar o número de comparações numa árvore a seguinte função pode ser usada (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->palavra,chave)==0)
             return a;
        else
        {
            if (strcmp(a->palavra,chave)>0)
                a = a->esq;
            else
                a = a->dir;
        }
    }
    return NULL;
}
```

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

★ É necessário elaborar **um relatório detalhado** com a **análise comparativa** do desempenho das duas estruturas para as operações de indexação e consulta. Utilize recursos como tabelas e gráficos para dar suporte às suas conclusões.

★ O trabalho deve ser feito, **preferencialmente**, em duplas mas também pode ser individual. A linguagem de programação aceita é C (Não é C++ nem C#).

3. Exemplo de funcionamento

Entrada: entrada.txt

```
1;Estou aqui com 2% de bateria e o carregador está no quarto ao lado :)
2;Demorou pacas fiquei até com a bateria em 4% hehe tá aí :)
3;Já temos muitas iniciativas bacanas, mas queremos novas ideias. Queremos ouvir você!
4;Amei te ver sorrindo
5;Frases do Livro Para Todos os Garotos que já Amei
6;pra quem ficar em duvida sobre oq me dar de aniversario, fica a dica!!!
7;Se você pudesse comer uma única coisa para o resto da sua vida, o que seria? - pipoca
8;me mandem perguntinhas
9;Atitudes simples que mudam o mundo
10;O outro mouse é o Logitech G700S, sem fio, com bateria interna recarregável
```

Entrada: consulta.txt

```
bateria
você
dado
amei
receptor
```

Comando: `C:\minhaaplicacao teste.txt consulta.txt saida.txt`

Saída: `saida.txt`

```
consulta: bateria  Palavra encontrada nos tweets 1, 2, 10,
consulta: você     Palavra encontrada nos tweets 3, 7,
consulta: dado     Palavra não encontrada
consulta: amei     Palavra encontrada nos tweets 4, 5,
consulta: receptor Palavra não encontrada

********** Estatísticas da Indexação **************
nodos = 84
comparações = 1497
rotações = 36
altura da árvore = 8

********** Estatísticas das Consultas **************
comparações = 29
```

4. Entrega e Apresentação

- **12 de maio de 2021** apresentação (horário da aula) e entrega pelo Moodle

5. Critérios de Avaliação

O trabalho deve ser realizado em duplas e deverá ser apresentado e defendido 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 incorrerá na divisão da nota obtida pelo número de alunos envolvidos. Para detectar o plágio, usaremos o software MOSS (http://theory.stanford.edu/~aiken/moss/).

É permitido reusar códigos vistos em aula (slides) e até mesmo funções de manipulação de árvores splay e rubro-negras encontradas na Internet. A fonte de todos os códigos reusados precisa ser identificada no trabalho.