Busca de Tweet

Na primeira parte do trabalho, foi implementada uma solução para exibir as *Trending Words* do Twitter. Porém, o Twitter agora precisa de uma funcionalidade para buscar um determinado tweet em seu conjunto de dados. Uma vez que são publicados milhares de *tweets* por dia no mundo todo, a sua solução deve ter um bom desempenho. Para que isso seja garantido, seu grupo deverá analisar o desempenho de algoritmos de busca em estruturas balanceadas e auto ajustáveis em diferentes cenários, descritos a seguir. Esta análise consistirá em comparar os algoritmos considerando três métricas de desempenho: número de comparações de chaves, o número de cópias de registros realizadas, e o tempo total gasto para busca (tempo de processamento e não o tempo de relógio).

Você deverá utilizar um conjunto real de *tweets* públicos para os experimentos. No link https://archive.org/details/twitter_cikm_2010, há mais de 5 milhões de *tweets* coletados entre Setembro de 2009 a Janeiro de 2010 e considerar a mesma estrutura de dados do Trabalho 1:

- `USERID (i.e., identificador do usuário que postou o tweet – tipo inteiro)`
- `TWEETID (i.e., identificador do tweet – tipo inteiro e` **`chave para ordenação`**`)`
- `TWEET (i.e., texto do tweet – tipo char, máximo de 140 caracteres)`
- `DATE (i.e., texto contendo a data do tweet – tipo char, formato AAAA-MM-DD HH:MM:SS)`

1 – Análise dos Algoritmos Utilizando Memória Interna

Você deverá avaliar o desempenho das árvores AVL, Vermelho e Preta, Splay e B ao inserir e remover um *tweet* e buscar um *tweet* pelo seu `TWEETID`.

Você ainda deverá implementar funções/métodos para importar os conjuntos de elementos aleatórios. Estes métodos/funções devem ser chamados uma vez para cada um dos N elementos a serem ordenados.

Análise:

Os algoritmos deverão ser aplicados a entradas com diferentes tamanhos (parâmetro N). Para cada valor de N, você deve gerar 5 (cinco) conjuntos de elementos diferentes, utilizando sementes diferentes para o gerador de números aleatórios. Você pode gerar um número aleatório com valores entre 1 e o número de linhas do seu arquivo de dados e importar o dado correspondente ao número da linha gerado. Experimente, no mínimo, com valores de N = 1000, 5000, 10000, 50000, 100000, 500000 e 1000000. Os algoritmos serão avaliados comparando os valores médios das 5 execuções para cada valor de N testado.

O seu programa principal deve ser receber três arquivos de entrada (`entradaInsercao.txt entradaBusca.txt entradaRemocao.txt`) com o seguinte formato:

7 → número de valores de N que se seguem, um por linha
1000

5000
10000
50000
100000
500000
1000000

Para cada valor de N, lido do arquivo de entrada `entradaInsercao.txt`:

- Gera cada um dos conjuntos de elementos, constrói a árvore, contabiliza estatísticas de desempenho para inserção na árvore analisada
- Armazena estatísticas de desempenho em arquivo de saída (`saidaInsercao.txt`).

Para cada valor de N, lido do arquivo de entrada `entradaBusca.txt`:

- Realiza a busca na árvore gerada pelas entradas da inserção, contabiliza estatísticas de desempenho para busca na árvore analisada
- Armazena estatísticas de desempenho em arquivo de saída (`saidaBusca.txt`).

Para cada valor de N, lido do arquivo de entrada `entradaRemocao.txt`:

- Realiza a remoção na árvore gerada pelas entradas da inserção, contabiliza estatísticas de desempenho para remoção na árvore analisada
- Armazena estatísticas de desempenho em arquivo de saída (`saidaBusca.txt`).

Ao final, basta processar os arquivos de saída referentes a cada uma das sementes, calculando as médias de cada estatística, para cada valor de N e para cada estrutura de dados considerados.

Resultados:

Apresente gráficos e tabelas para as três métricas pedidas, número de comparações, número de cópias e tempo de execução (tempo de processamento), comparando o desempenho das árvores e diferentes valores de N. Discuta seus resultados. Quais são os compromissos de desempenho observados?
Qual o impacto das variações nos valores da ordem para as Árvores B?

2 – Análise da Busca Utilizando Memória Externa

Implemente agora a busca por um *tweet* que considere todo o arquivo de dados. Você poderá usar a sua criatividade para resolver o problema da melhor forma, mas será avaliado de acordo com a qualidade da sua solução em dois aspectos: quanto o seu sistema é rápido e o quanto a estrutura de dados é interessante para resolver o problema.

Considerações

1) Todo código fonte deve ser documentado. A documentação inclui, dentre outros, a documentação de procedimentos, de funções, de variáveis, de partes do código fonte que realizam tarefas específicas. Ou seja, o código fonte deve ser documentado tanto

em nível de rotinas quanto em nível de variáveis e blocos funcionais.
2) A interface pode ser feita em modo texto (terminal) ou modo gráfico e deve ser funcional.
3) A implementação deve ser realizada usando a linguagem de programação C, C++ ou Java.

Entrega

O grupo deverá ser formado por 4 alunos e as responsabilidades de cada aluno deve ser documentada e registrada. O prazo final para entrega é dia **06/11.** Deverá ser agendada uma data para entrega e apresentação do trabalho para a professora.

Deve ser entregue os códigos implementados e um relatório com os seguintes itens:
1) Descrição das atividades realizadas por cada membro do grupo
2) Análises da Parte 1
3) A explicação sobre as estruturas de dados implementadas na Parte 2

Critérios de avaliação

Você não fechará o trabalho só tendo um "sistema que funciona". O sistema deve funcionar bem e o quão bem ele funcionar será refletido na sua nota. A nota poderá ser comparativa, então se esforce para ter uma solução melhor que a dos outros colegas. O objetivo do trabalho é testar a sua capacidade de fazer boas escolhas (e boas adaptações) de estruturas para resolver problemas. **Então usar classes prontas ou métodos prontos não são permitidos aqui.** Você poderá, se quiser, comparar sua solução com outras prontas. Mas deve perseguir o seu melhor sem usar recursos de terceiros.

Os membros da equipe serão avaliados pelo produto final do trabalho e pelos resultados individuais alcançados. Assim, numa mesma equipe, um membro pode ficar com nota 90 e outro com nota 50, por exemplo. Dentre os pontos que serão avaliados, estão:

- Execução do programa (caso o programa não funcione, a nota será zero)
- Código documentado e boa prática de programação (o mínimo necessário de variáveis globais, variáveis e funções com nomes de fácil compreensão, soluções elegantes de programação, código bem modularizado, etc)
- Testes: procure fazer testes relevantes como, por exemplo, aqueles que verificam casos extremos e casos de exceções
- Relatório bem redigido

Note que o grande desafio deste trabalho está na avaliação dos vários algoritmos nos diferentes cenários, e não na implementação de código. Logo, na divisão de pontos, a documentação receberá, no mínimo, 50% dos pontos totais.
Uma boa documentação deverá apresentar não somente resultados brutos mas também uma discussão dos mesmos, levando a conclusões sobre a superioridade de um ou outro algoritmo, para cada métrica avaliada.