Book Depository Dataset

BookDepository.com é um site internacional de vendas online de livros, com mais de 20 milhões de obras cadastradas. Uma grande coleção de dados sobre os livros vendidos pelo BookDepository.com pode ser obtido em https://www.kaggle.com/sp1thas/book-depository-dataset. Este conjunto de dados contém uma lista de metadados como título, descrição, dimensões, categoria e outros.

Seu grupo ficou encarregado de processar esse conjunto de dados, analisando o desempenho de algoritmos de ordenação e da busca em estruturas balanceadas, além de utilizar tabelas hash para determinar a lista de autores mais frequentes. Cada uma destas etapas é descrita nas seções a seguir. Para as etapas 1 e 3, deverá ser feita uma comparação dos algoritmos considerando três métricas de desempenho: número de comparações de chaves, o número de cópias de registros realizadas, e o tempo total gasto (tempo de processamento e não o tempo de relógio).

Você deverá utilizar um conjunto real de registros para os experimentos. Como dito anteriormente, os arquivos disponibilizados no Google Classroom são uma versão simplificada do dataset no link original. Há em torno de 1 milhão de registros de livros armazenados em um arquivo CSV: “dataset.csv”, contendo metadados desses livros. Os arquivos "authors.csv" e "categories.csv" se relacionam com o arquivo "dataset.csv" pelos campos identificadores. Por exemplo, o campo *author_id* de “authors.csv” está relacionado com o campo *author* de "dataset.csv".

Entrega

O grupo deverá ser formado por no máximo 4 alunos e as responsabilidades de cada aluno deve ser documentada e registrada. A distribuição das responsabilidades deve ser feita de maneira uniforme, de modo que cada membro do grupo se envolva com o trabalho na mesma proporção que os demais. **O trabalho está dividido em 2 partes com os seguintes prazos de entrega:**

- **PARTE 1** (referente à Seção 1): **20/10.**
- **PARTE 2** (referente às Seções 2 e 3): **25/11**

Deverá ser agendada uma data para apresentação do trabalho para o(a) professor(a) no final do período. A data máxima para apresentação é o dia 27/11.

Todos os itens abaixo devem ser entregues:

1. Código-fonte completo;

2. Arquivo executável (preferencialmente Linux), com instruções de execução, incluindo a informação de em que diretório o dataset deve estar localizado para a correta execução;
    a. **Não incluir o dataset na submissão via Google Classroom.**
3. Relatório (google docs) contendo as seguintes informações:
    a. Descrição das atividades realizadas por cada membro do grupo;
    b. Análises dos resultados obtidos em cada uma das seções;
    c. Explicação sobre as estruturas de dados implementadas.

1 – Análise dos Algoritmos de Ordenação

- Os elementos a serem ordenados são registros/objetos armazenados em um vetor de tamanho N. Cada registro contém:

    ➔ `authors`: lista de ids de autores entre colchetes e separados por vírgulas (os nomes dos autores associados aos ids estão em `authors.csv`)
    ➔ `bestsellers-rank`: ranking na lista de mais vendidos (`int`)
    ➔ `categories`: lista de ids de categorias de livros (os nomes das categorias estão em categories.csv)
    ➔ `description`: descrição do livro (`str`)
    ➔ `edition`: edição (`str`)
    ➔ `id`: id único do livro atribuído pelo Bookdepository.com (`int`)
    ➔ `isbn10`: ISBN-10 (`str`)
    ➔ `isbn13`: ISBN-13 (`str`)
    ➔ `rating-avg`: avaliação média 0-5
    ➔ `rating-count`: número de avaliações
    ➔ `title`: título (`str`)

Note que os arquivos estão no formato CSV (*Comma-separated values*). Isso significa que cada campo listado acima é separado do outro por uma vírgula.

Você deverá instrumentar os algoritmos para contabilizar o número de comparações de chaves, o número de cópias de registros e o tempo total gasto na ordenação. Estes números deverão ser impressos ao final de cada ordenação para posterior análise.

Você irá comparar o algoritmo Quicksort com um outro algoritmo de ordenação de sua escolha para ordenar um conjunto de N elementos. Os elementos do vetor devem ser os registros do dataset, usando o título dos livros como chave de ordenação, importados aleatoriamente da sua base de dados. **Você deverá apresentar, no seu relatório, o algoritmo escolhido, explicitando sua fonte.**

Para fazer a ordenação, você deverá carregar um número de registros de forma desordenada inicialmente. Você deverá, portanto, 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.

Cada algoritmo deverá ser aplicado a entradas escolhidas aleatoriamente 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 selecionar um valor aleatório entre 1 e o número de bytes do arquivo e importar o registro na linha correspondente ao deslocamento em bytes dado pelo valor selecionado (ou na linha seguinte a esta). Experimente, no mínimo, com valores de N = 1000, 5000, 10000, 50000 e 100000. Os algoritmos serão avaliados comparando os valores médios de 5 execuções para cada valor de N testado.

O seu programa principal deve receber um arquivo de entrada (`entrada.txt`) com o seguinte formato:

5 → número de valores de N que se seguem, um por linha
1000
5000
10000
50000
100000

Para cada valor de N lido do arquivo de entrada:

- Gera cada um dos conjuntos de elementos, ordena, contabiliza estatísticas de desempenho
- Armazena estatísticas de desempenho em arquivo de saída (`saida.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 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 médio do Quicksort e do MeuSort para diferentes valores de N. Discuta seus resultados. Quais são os compromissos de desempenho observados?

2 – Implementação dos Autores mais Frequentes

Você deverá implementar um programa que leia N livros aleatórios e diferentes e conte quantas vezes um mesmo autor se repete dentro desses N livros, ou seja, determinar a quantidade de livros à venda de um determinado autor.

Você deverá utilizar uma tabela *hash* para armazenar livros e outra para pesquisar e inserir os **nomes** dos autores dos livros. **Lembre-se que não deve haver livros repetidos.** As funções *hash* devem apresentar duas propriedades básicas: seu cálculo deve ser rápido e deve gerar poucas colisões. Além disso, é desejável que ela leve a uma ocupação uniforme da tabela para conjuntos de chaves quaisquer. Escolha um dos métodos estudados para tratar as colisões.

Seu programa deve ler cada livro da tabela *hash* de livros. Cada autor encontrado deve ser armazenado na tabela de autores. No final da execução, seu programa deve imprimir os N (parâmetro definido pelo usuário) autores mais frequentes. A saída deve ser ordenada de forma decrescente de frequência. Para ordenação, escolha o algoritmo de ordenação com melhor desempenho na Seção 1. O grupo pode usar a sua criatividade para implementar a tabela de autores, mas será avaliado de acordo com a qualidade da sua solução.

3 – Busca em Estruturas Balanceadas e Auto-Ajustáveis

Você deverá avaliar o desempenho de estruturas balanceadas ao inserir um livro usando como chave o id do livro. Você também deverá analisar o desempenho dessas estruturas ao realizar a busca pelo livro. As estruturas que devem ser avaliadas são:

- Árvore Vermelho-Preto
- Árvore B (d=2)
- Árvore B (d=20)

Considere que a ordem d da árvore B representa o número **mínimo** de chaves em cada nó (exceto a raiz), conforme visto em sala de aula. Não há necessidade de se manter preso aos valores de d especificados acima, o grupo pode adotar valores diferentes (ou mais que dois valores) para os testes. Tenha o cuidado, no entanto, de escolher valores de d que permitam avaliar a diferença no desempenho para árvores de ordem baixa e árvores de ordem alta.

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 e 500000. Os algoritmos serão avaliados comparando os valores médios das 5 execuções para cada valor de N testado.

Para cada valor de N lido do arquivo de entrada:

- 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).
- 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).

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?

Considerações

1. Todo código fonte deve ser documentado. A documentação inclui, dentre outros, a documentação de procedimentos, de funções, de variáveis e 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.

A implementação deve ser realizada usando a linguagem de programação C, C++ ou Java.

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 em cada cenário considerado, para cada métrica avaliada.