Universidade Federal do Ceará - Campus Quixadá
QXD0115 – Estrutura de Dados Avançada
Curso de Ciência da Computação
Prof. Atílio Gomes Luiz

Contador de frequências usando Estruturas de Dados Avançadas

1. Nesse trabalho, deverá ser desenvolvida uma aplicação que utilize as árvores binárias de busca balanceadas vistas em aula (AVL e Rubro-Negra) e também as tabelas hash.

   Faça um programa que leia um texto qualquer (arquivo no formato .txt) e imprima, em ordem alfabética, as palavras e a sua frequência no texto. Por exemplo, no texto "quatro mais quatro são oito" o seu programa deverá imprimir:

mais	1
oito	1
quatro	2
são	1

   Observação: Como o arquivo-texto dado como entrada para a aplicação pode ser muito grande, a tabela de frequências pode vir a ser gigantesca e a sua impressão no terminal pode ser inviável. Logo, quando falo "imprimir", com isso estou querendo dizer que a tabela de frequências deve ser impressa em algum lugar para posterior análise. Por exemplo, você pode gravar a saída em um arquivo.

   A leitura do arquivo .txt deverá desprezar espaços em branco e sinais de pontuação. Os espaços em branco serão considerados separadores de palavras. Sinais de pontuação tais como: sinal de interrogação, de exclamação, vírgulas, ponto-e-vírgulas, travessões, chaves, colchetes, dentre outros, não devem ser considerados como palavras e devem ser descartados. A única exceção é para palavras compostas com hífen: super-homem, pré-história, sub-hepático, auto-hipnose, neo-humanismo, etc.

   Além disso, a leitura deverá converter todas as letras maiúsculas em minúsculas.

   A pesquisa e a inserção das palavras do texto deverão ser implementadas **com cada uma das** seguintes **estruturas de dados**:

   (a) Dicionário (usando como base uma árvore AVL)

   (b) Dicionário (usando como base uma árvore Rubro-Negra)

   (c) Dicionário (usando como base uma tabela Hash com tratamento de colisão por encadeamento exterior)

   (d) Dicionário (usando como base uma tabela Hash com tratamento de colisão por endereçamento aberto)

   Coloque contadores no seu programa para determinar o **número de comparações de chaves** necessárias para montar a tabela de frequências em cada uma das estruturas acima (Apenas o número de comparações entre chaves para montar a estrutura. Não é necessário considerar as comparações para a impressão ordenada).

   Calcule também o tempo que cada estrutura leva apara montar a tabela.

Você deve propor pelo menos uma outra métrica adicional e analisá-la na sua aplicação. Por exemplo, no tocante aos dicionários implementados por meio de árvores balanceadas, você poderia contar o número de rotações realizadas. Essa é só uma sugestão, se você quiser pode pensar em outra métrica.

Analise, através dos dados coletados, o desempenho/eficiência de cada estrutura.

Use parâmetros de linha de comando para fazer as chamadas do seu programa. Esse tipo de execução é bastante comum em sistemas Linux. Por exemplo, se o seu programa executável chama-se `freq` e você quiser contar a frequência do arquivo `texto.txt` utilizando o dicionário com AVL, sua chamada pode ser feita assim na linha de comando do Linux (ou prompt de comando, no caso do Windows):

> `freq dictionary_avl texto.txt`

Descubra como fazer para o seu programa ler os parâmetros da linha de comando e defina (e documente) a sintaxe dos comandos e dos parâmetros a serem utilizados pelo usuário. Veja, nesse trabalho não estou pedindo uma aplicação com menu, mas sim uma aplicação de linha de comando.

Como falei acima, a tabela de frequências gerada na saída pode ser impressa diretamente num arquivo texto de saída. Se quiser, pode dar também ao usuário a possibilidade de nomear o arquivo de saída na linha de comando, fica a seu critério.

Também faz parte do trabalho descobrir como fazer a leitura de um arquivo texto e a manipulação adequada de strings.

Restrições com relação às EDs

- As estruturas de dados usadas neste projeto devem ser de uso geral, ou seja, elas devem ser genéricas. Elas não devem estar atreladas a esse projeto específico. Deve ser possível guardar pares de chave-valor de qualquer tipo válido. Logo os tipos desses dados são genéricos.
- Sendo mais explícito, estou pedindo que você implemente cada uma das 4 estruturas de dados como classes com templates. Cada estrutura de dado deve ser capaz de guardar um par de **(Chave, Valor)**. O campo **Chave** será a palavra lida no livro, já o campo **Valor** será a quantidade de vezes que a palavra foi encontrada até o momento durante a leitura. Ao final da execução da aplicação, quando a aplicação tiver terminado de ler o livro, cada Chave terá associada a ela a quantidade total de vezes que ela apareceu no texto. Esse valor associado é a frequência da chave.
- A aplicação não deve ter acesso à estrutura interna da classe. As regras de Programação Orientada a Objetos devem ser seguidas à risca.
- Todas as principais operações de um dicionário devem ser implementadas, mesmo que não venham a ser utilizadas. São elas: criação, inserção, acesso, atualização, remoção e destruição.
- A fim de imprimir os dados da ED na tela, isso pode ser feito com uma ou mais funções-membro públicas feitas especificamente para isso. Essa não é a melhor solução que existe para resolver esse problema. A solução mais correta e amplamente usada é implementar um iterador. Porém, não é obrigatório ter que implementar um iterador, dada a dificuldade de fazer isso e o tempo curto.

- **Dica**: O C++ oferece dois tipos de dicionário: `std::map` e `std::unordered_map`. O std::map é implementado usando árvore rubro-negra e o std::unordered_map é implementado usando tabela hash. Pesquise e estude sobre essas duas classes. Como funcionam no C++. Basicamente, uma parte do que estou pedindo neste trabalho, é que você implemente esses contêineres.

Algumas considerações com relação ao processamento do texto

- Sinais de pontuação (aspas simples, aspas duplas, sinal de interrogação, sinal de exclamação, vírgula, dois-pontos, ponto final, símbolos aritméticos tais como igualdade, soma, subtração e outros,) devem ser desconsiderados, ou seja, não são entradas do dicionário, nem fazem parte de palavras.
- O caso do hifen é mais especial. Ele pode ser usado num texto para indicar a fala de um personagem. Neste caso, ele pode ser descartado. Em português, o hífen é usado em palavras, tais como "mostrá-lo". Neste caso, acho prudente o hífen ser parte da palavra. Logo, neste caso, **mostrá-lo** seria uma entrada do dicionário.

O que deve ser entregue

- Código fonte do programa em C++ (bem indentado e comentado).
- Documentação **sucinta** do trabalho. Entre outras coisas, a documentação deve conter:
  1. Introdução: descrição do problema a ser resolvido e visão geral sobre o funcionamento do programa.
  2. Implementação: descrição sobre a implementação do programa. Descrever a estrutura de dados utilizada (de preferência com diagramas ilustrativos), o funcionamento das principais funções e procedimentos utilizados, o formato de entrada e saída de dados, bem como decisões tomadas relativas aos casos e detalhes de especificação que porventura estejam omissos no enunciado.
  3. Listagem de testes executados e Resultados encontrados: os testes executados devem ser simplesmente apresentados. Devem ser apresentados os resultados com relação às métricas que foram pedidas acima na descrição do trabalho: número de comparação de chaves, tempo de montagem da tabela, análise do desempenho de cada ED, entre outras que você inventar.
  4. Conclusão: comentários gerais sobre o trabalho e as principais dificuldades encontradas em sua implementação.
  5. Bibliografia: bibliografia utilizada para o desenvolvimento do trabalho, incluindo sites da internet se for o caso.
  6. Formato: obrigatoriamente em PDF.
  7. Em Latex: a documentação do trabalho pode ser feita em formato de artigo. De preferência, prepare esse documento em Latex. Atualmente, existe o site **Overleaf** que facilita a produção de documentos em latex. Você pode Latex usar o modelo de artigos da Sociedade Brasileira de Computação (link aqui).

Como deve ser feita a entrega final

A entrega deve ser feita pelo Moodle na forma de um único arquivo zipado, contendo o código, os arquivos e a documentação.

Comentários Gerais:

- **Comece a fazer este trabalho com antecedência**, enquanto o problema está fresco na memória e o prazo para terminá-lo está tão longe quanto jamais poderá estar;
- Clareza, identação e comentários no programa também vão valer pontos;
- **O trabalho é individual** (grupo de UM aluno);
- Trabalhos copiados terão nota zero;
- Trabalhos entregue em atraso não serão aceitos.
- **Não existe segunda chamada para projetos. Logo, se você não fizer este projeto, automaticamente receberá a nota ZERO.**
- Eu devo disponibilizar alguns livros para teste. Um livro que vocês devem usar como teste é a bíblia: `https://openbible.com/textfiles/kjv.txt`

Dica 1: O autor de livros e professor Nívio Ziviani compôs algumas dicas de como deve ser feita uma boa implementação e documentação de um trabalho prático, que vale a pena ser consultada (link aqui).

Um olhar mais cuidadoso: O que são dicionários?

Um dicionário é uma estrutura de dados que armazena pares de chave-valor. Cada chave é única e é usada para acessar o valor associado a ela. Os dicionários são extremamente úteis para buscas rápidas, pois permitem acessar os valores diretamente através das chaves, em vez de iterar por todos os elementos.

Os dicionários são implementados de forma que a pesquisa, inserção e exclusão de pares de chave-valor sejam operações eficientes. Nos dicionários fornecidos pela maioria das linguagens de programação, como Python e C++, essas operações têm complexidade de tempo média $O(1)$ ou $O(\lg n)$, dependendo se foi implementado com um hash ou árvore balanceada, respectivamente.

Os dicionários são mutáveis, o que significa que você pode alterar suas entradas depois que elas são criadas, adicionando, removendo ou modificando pares de chave-valor.

Aplicações de Dicionários:

- Mapeamento de dados: Dicionários são ideais para armazenar e manipular dados que podem ser mapeados de maneira clara, como configurações, dados de usuário, resultados de buscas, etc.
- Contagem de frequências: Podem ser usados para contar a frequência de elementos em uma lista.

- Armazenamento de dados em cache: Permitem armazenar resultados de operações caras para acesso rápido subsequente.
- Representação de grafos: Dicionários podem ser usados para representar grafos, onde as chaves são nós e os valores são listas de nós adjacentes.

Essa flexibilidade e eficiência tornam os dicionários uma estrutura de dados muito poderosa e amplamente utilizada em diversas áreas da programação.

As operações básicas que um dicionário deve fornecer para o usuário incluem:

- Criação: Permitir a criação de um dicionário vazio ou com pares de chave-valor iniciais.
- Inserção: Adicionar um novo par chave-valor ao dicionário.
- Atualização: Modificar o valor associado a uma chave existente.
- Acesso: Recuperar o valor associado a uma chave específica.
- Remoção: Remover um par chave-valor do dicionário usando a chave.
- Verificação de existência: Verificar se uma chave existe no dicionário.
- Iteração: Percorrer os pares de chave-valor do dicionário.
- Tamanho: Obter o número de pares chave-valor no dicionário.
- Limpeza: Remover todos os pares chave-valor do dicionário.

Partes do Projeto

Esse trabalho pode ser dividido em duas partes principais:

- **Parte 1:** A primeira parte do trabalho lida com a implementação das estruturas de dados, que devem ser genéricas e devem funcionar independentemente da finalidade para a qual serão utilizadas. **Atenção:** A única exceção a isso que eu acabei de dizer é que eu estou pedindo para vocês colocarem contadores dentro das estruturas de dados. Tirando este pequeno detalhe, todo o restante da implementação das estruturas de dados pode ser realizado de forma independente da aplicação.
- **Parte 2:** A segunda parte do trabalho lida com a interface do programa, a manipulação de arquivos e, mais importante ainda, o processamento de strings.

Prazos e Entregas

Entrega da Parte 1 do Projeto: 30 de junho de 2025

Nesta data, todas as estruturas de dados devem estar completamente implementadas e funcionando corretamente, já com os devidos contadores. As métricas escolhidas devem estar documentadas em um arquivo pdf que deve ser enviado juntamente com as implementações.

Data de entrega do Projeto completo, com documentação finalizada, e aplicação funcionando, testes já realizados: 15 de julho de 2025.

A apresentação do projeto será feita logo em seguida.