Universidade Federal de Pernambuco – UFPE
Centro de Informática
Sistemas de Informação
IF969 – Algoritmos e estruturas de dados
Professor: Renato Vimieiro
Projeto de Implementação **Entrega: 05/09/16**

Introdução

O objetivo deste trabalho é praticar o desenvolvimento de programas usando as técnicas estudadas em sala de aula, em particular, o uso de algoritmos de busca (busca binária) e estruturas de dados lineares (listas duplamente encadeadas).

Você deve gastar pelo menos uma hora lendo este documento para se certificar que entendeu completamente o que é exigido, de forma a não perder tempo com idas e vindas ao documento para entender o que deve ser feito, e também não entregar algo que você acredite estar correto mas que depois descubra que se enganou com alguma especificação. Leia atentamente o documento destacando partes importantes, anotando as dúvidas para saná-las o quanto antes.

Autocompletar

Este projeto é uma adaptação do que foi proposto pelo Prof. Kevin Wayne na Universidade de Princeton (http://www.cs.princeton.edu/courses/archive/spr14/cos226/assignments/autocomplete.html).

A funcionalidade de autocompletar está presente em muitos sistemas atuais. Por exemplo, o Google sugere possíveis buscas quando o usuário começa a digitar aquilo que deseja procurar. O mesmo pode ser observado em aplicativos para celular, processadores de texto (Word), entre outros.

O tempo de processamento para esses sistemas é crítico, uma vez que a sugestão será útil para o usuário somente se for apresentada em um curtíssimo espaço de tempo. Estudos apontam que, em geral, esses sistemas possuem apenas 50ms para retornar uma lista de sugestões. Além disso, deve-se ter em mente que o sistema deve executar uma busca todas as vezes que uma tecla é pressionada.

Sistemas de autocompletar podem ser implementados de diversas formas. Nesse projeto você irá implementá-lo através de vetores, listas encadeadas e algoritmos de busca (busca binária). Você deverá carregar um conjunto de consultas; ordená-las em ordem lexicográfica; usar a busca binária para determinar um subconjunto de consultas que começam com um dado prefixo; e retornar uma lista duplamente encadeada com os k elementos mais relevantes em ordem descendente.

Resumo: como o programa funciona

A Figura 1 mostra a interface gráfica do sistema. O usuário deve digitar o caminho do arquivo no campo "Arquivo" e clicar em "Ok" para carregar o arquivo de consultas no programa. Tendo carregado, o usuário deve digitar a busca no campo "Palavra". O sistema usará o prefixo no campo "Palavra" para buscar as consultas mais relevantes, retornando as "Qtd" melhores em ordem. A lista será exibida no campo "Resultado".

O que fazer?

Um esqueleto do programa final está disponível na página da disciplina. Ele contém uma implementação completa da interface, e outras classes não implementadas que fornecem as funcionalidades discutidas acima.

Seu trabalho é implementar funções de comparação de termos de consulta (palavra e peso, cuja estrutura de

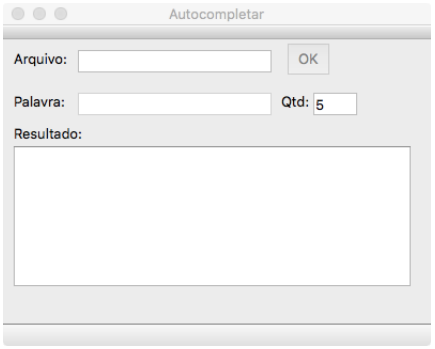

Figura 1: Interface gráfica do sistema de autocompletar.

dados está descrita em Palavra). Você deve sobrecarregar o operador $<$ para comparar dois objetos da classe Palavra em ordem lexicográfica dos termos. Além disso, você deve implementar uma função de comparação por prefixo, e por peso. Essas funções devem retornar um inteiro negativo, zero, ou positivo no caso do primeiro parâmetro ser menor, igual ou maior que o segundo.

Você também deve implementar a classe Controle, que é responsável por fornecer as funcionalidades de *carregarDados* e *find* para carregar o arquivo de consultas e ordená-las, e encontrar a lista de sugestões que casam com o prefixo informado. Você também deve implementar dois métodos privados, *firstIndexOf* e *lastIndexOf* que retornam as posições da primeira e última consultas que possuem o prefixo desejado. Essas funções devem ser usadas para encontrar a parte do vetor de consultas que deve ser processado para encontrar as k consultas mais relevantes.

Finalmente, as k consultas mais relevantes devem ser inseridas numa lista duplamente encadeada (classe Lista). Você deve implementar as funções *inserirOrdenado* e *removerFim*. O método *inserirOrdenado* deve inserir os elementos em ordem de acordo com a função de comparação passada como segundo parâmetro. Você também deve implementar as funções que permitam imprimir uma lista usando as funções de saída disponíveis (arquivo ou console).

Arquivo de entrada

O pacote com o esqueleto do projeto contém três arquivos de teste. Os arquivos de entrada possuem o seguinte formato: a primeira linha contém somente um inteiro com a quantidade de consultas no arquivo; as consultas aparecem a partir da segunda linha, sendo o primeiro termo um peso associado à consulta que deve ser usado como atributo de relevância (maior peso mais relevante), e o segundo os termos (palavras) da consulta. Os campos são separados por um caracter de tabulação. Os campos podem conter espaços em branco no início e fim que devem ser removidos.

Outros arquivos de teste podem ser obtidos em `http://www.cs.princeton.edu/courses/archive/spring15/cos226/checklist/autocomplete.html#data`. Teste o desempenho do seu programa, especialmente, com arqui-

vos grandes. Procure verificar se variações no tamanho/quantidade de consultas causa grandes diferenças no desempenho do programa.

Logística

- O trabalho poderá ser feito em grupos de no máximo dois componentes
- Os grupos devem se inscrever por **email**, enviando para o professor o nome e matrícula dos integrantes até o dia **12/09**.
- O código completo do programa deverá ser entregue por **email** até às **23h59 do dia 25/09/16**. Os códigos não entregues no horário serão penalizados com a perda de pontos. Os códigos devem ser compactados em formato zip ou tgz. **Arquivos em formato rar ou qualquer outro formato não serão aceitos e o grupo perderá pontos por atraso**.
- Trabalhos copiados de colegas ou da internet, seja trechos ou totalidade, serão prontamente anulados (todos os envolvidos).
- O professor tem o direito de pedir que o grupo apresente o projeto. Nesse caso, ele também tem direito de escolher a quem dirigirá a pergunta durante a apresentação.