Relatório

Indexador e Buscador de Palavras

Júlia Du Bois Araújo Silva (14584360)

Lucas Kaufmann (14591100)

Thiago Vieira Calile Costa(14609529)

As estruturas foram implementadas de acordo com aquilo que o grupo acreditou ser mais interessante.

A lista foi implementada unicamente ligada, além de sua indexação sendo análoga a um dicionário empregado por estudiosos de línguas, obedecendo a ordem alfabética. Não foi mantido um nó cabeça ou um ponteiro para o último item, pois acelerariam apenas levemente a indexação da lista dado seu método de organização.

A árvore teve uma implementação programaticamente direta: ao utilizar a função strcmp da biblioteca string.h, a indexação se deu ao utilizar o retorno da função para decidir se o nó seria inserido à esquerda ou à direita.

As buscas também obedeceram lógicas similares, com a lista unicamente ligada tendo uma busca linear, enquanto a árvore binária tem, por padrão, a busca binária.

Os seguintes testes foram executados em um computador com um processador Intel i3 6100U.

Foram realizados testes com seis textos diferentes, para benchmarking, utilizamos alguns livros disponíveis em formato de texto plano pelo Projeto Gutenberg. Esses textos foram “Carmilla”, “Dom Casmurro”, um trecho curto de Lorem Ipsem, “A Metamorfose”, “Moby Dick” e “Os Três Porquinhos”.As informações sobre os resultados brutos estão presentes no Anexo. No resultado, foram analisados os resultados de Moby Dick, o texto mais longo analisado.

Resultados

Como esperado, a indexação varia em tempo entre as duas implementações. Isso é melhor visto nos casos de alto volume. No caso, com os testes dos programas sobre a versão disponível de Moby Dick, a implementação de lista indexou a obra em 166 segundos, enquanto a árvore conseguiu realizar a mesma tarefa em 0,6 segundo.

Porém, ambas as implementações, concluídas as indexações, tinham suas buscas extremamente rápidas, não demorando mais que 1 milissegundo.

A razão principal pela qual a indexação de textos é mais lenta na lista quando comparada com a árvore é por conta da necessidade de percorrer a lista sequencialmente. A inserção exige uma varredura completa para adicionar uma nova palavra, enquanto na árvore binária de busca é permitida uma inserção mais ágil, pois é possível desconsiderar partes da árvore com base nas comparações, assim tendo menos operações para inserir uma nova palavra.

A complexidade de memória não foi estudada como um todo, mas podemos inferir que ambas as implementações são iguais, já que o objeto de maior consumo de memória é o token da palavra, tendo apenas uma pequena variação entre o nó da árvore e da lista com a primeira guardando dois ponteiros e a primeira, apenas um.

Não acreditamos que textos com caracteres de várias línguas venham a causar problemas, desde que haja compatibilidade com a função strcmp. Testes em cirílico, por exemplo, com o texto dos Irmãos Karamazov por Fyodor Dostoievski, resultaram em falha de segmentação, provavelmente por conta disso. Como a necessidade de caracteres de múltiplas línguas não foi especificada no escopo do projeto, esse problema não foi investigado mais a fundo.

Não houveram testes sobre arquivos compostos apenas com os múltiplos caracteres de texto branco e controle presentes nos padrões internacionais. Também não foram testados

caracteres sem qualquer tipo de ordem, como “`:><:}{^ºªº¬²³£”, mas é possível que poderiam ser organizados segundo sua aparição na tabela ASCII, caso não fossem removidos do texto indexado como proposto pelo exercício.

A teoria se replica para caracteres que não são limpos pelo passo de “normalização” do texto.

Há um porém: caracteres especiais cujo valor ASCII ou Unicode são muito mais altos que suas versões não especiais. Por exemplo, o caractere “o” pode ter essa forma mais simples, assim como “ô”, “ó” e “õ”. Se a lista tivesse uma implementação idêntica a de um dicionário da língua portuguesa, a ordem provável das palavras seria: “poá → põe → pôr → porém → pormenor → português”, basicamente ignorando a acentuação e tomando o valor base da letra como a informação importante para a classificação.

Mas, ao tomar a função strcmp como sendo apenas uma comparadora de valores de cada símbolo individual segundo sua posição nas tabelas internacionais de caracteres, a ordem pode se dar como: “pormenor → português → porém → poá → pôr → põe”, dependendo de como cada convenção atribui prioridade a grafemas diferentes.

Anexo

A seguinte tabela inclui diferentes textos e tempo de indexação em ms em um computador com processador Intel® Core™ i5-8250U × 8. As palavras foram contadas utilizando a função wc 9.1 em um terminal Linux.

Arquivo	Número de palavras	Lista	Árvore
carmilla.txt	12.183	00971	00068
domcasmurro.txt	69.646	04800	00123
loremipsum.txt	70	00001	00000
metamorphosis.txt	25.058	00547	00058
mobydick.txt	215.693	78755	00302
threelittlepigs.txt	4.107	00032	00020

A seguinte tabela inclui o tempo de busca em ms de 3 palavras presentes em certo texto e uma não presente, para todos os 6 textos, com o índice em lista e em árvore, em um computador com processador Intel® Core™ i5-8250U × 8.

carmilla.txt			
Palavra buscada	**Ocorrências**	**Lista**	**Árvore**
carmilla	59	<00001	<00001
shadow	4	<00001	<00001
vampire	20	<00001	<00001
inexistente (pao)	0	<00001	<00001
domcasmurro.txt			
Palavra buscada	**Ocorrências**	**Lista**	**Árvore**
olhos	157	00002	<00001
casmurro	10	00001	<00001
capitú	331	00001	<00001

inexistente (ballet)	0	<00001	<00001
loremIpsum.txt			
Palavra buscada	**Ocorrências**	**Lista**	**Árvore**
lorem	1	<00001	<00001
laborum	1	<00001	<00001
ut	3	<00001	<00001
inexistente (amem)	0	<00001	<00001
metamorphosis.txt			
Palavra buscada	**Ocorrências**	**Lista**	**Árvore**
apple	5	<00001	<00001
couch	14	<00001	<00001
gregor	178	<00001	<00001
inexistente (insect)	0	<00001	<00001
mobydick.txt			
Palavra buscada	**Ocorrências**	**Lista**	**Árvore**
ishmael	18	00002	<00001
moby	77	00004	<00001
whale	1051	00002	<00001
inexistente (larva)	0	00003	<00001
threelittlepigs.txt			
Palavra buscada	**Ocorrências**	**Lista**	**Árvore**
pig	35	<00001	<00001
blow	4	<00001	<00001
wolf	18	<00001	<00001
inexistente (animal)	0	<00001	<00001