Danilo Araujo de Oliveira Romeira - 13725823
Gabriel Alexo Rocha - 13727636
Otavio Silva Gonçalves - 13672384

Exercício de programação
Algoritmos e Estrutura de Dados I

relatório do exercício de programação apresentado
na disciplina de Algoritmos e estrutura de dados

São Paulo
2022

ÍNDICE

1.RESUMO

O objetivo desse trabalho de programação é saber manusear strings de um texto de forma que é possível encontrar a localização de palavras quantas vezes se repete de forma rápida e minimamente eficiente a metodologia usada consiste em utilizar estrutura de dados para manusear as informações de forma fluida e tornar o objetivo uma realidade. Dessa forma, esse feito foi atingido utilizando a lista ligada e a árvore de busca binária.

2.INTRODUÇÃO

A principal missão desse ep foi a aplicação de estruturas de dados para manipular informações, principalmente com a finalidade de acessá-las de forma eficiente e as comparar quando for necessário. Em um primeiro momento e, como passo inicial, nos foi apresentado previamente um algoritmo capaz de separar as linhas e as palavras de um texto , assim, a missão foi em como manipular essas palavras com o intuito de cumprir o objetivo desse trabalho,ou seja criar uma estrutura que permita verificar a existência ou não de uma palavra sem que seja necessário verificar o texto linha a linha sendo que o programa deve exibir as linhas do texto nas quais a palavra ocorre.Os dois tipos de estruturas escolhidos foi a lista ligada e uma árvore de busca binária os motivos serão apresentados a seguir.

Lista ligada: a maior característica da lista ligada é que cada elemento armazena um dado e uma referência para o elemento seguinte da lista. Dessa forma, é permitido a grande vantagem desse tipo de lista ser dinâmica, assim, pode ser alterado tamanho durante a execução do programa.
Um dos motivos pela escolha desse tipo de lista é que será proporcionado facilidades na hora de inserir e remover elementos em qualquer posição da lista, uma vez que não é necessário mover os elementos subsequentes da lista.

Diante de tal contexto, a aplicação dessa estrutura proporciona ao código enviado facilidades na hora de inserir e remover as palavras do texto, já que poderá ser feito em uma complexidade temporal O(1).

Árvore de busca binária: o que sustenta a escolha deste algoritmo para esse ep são 3 pilares em relação às vantagens de sua aplicação. é fornecido inserções e remoções fáceis e também é possível realizar a busca de forma binária, dessa forma, a complexidade desse algoritmo é O(n log n).

3.MÉTODO

O programa parte da leitura de um arquivo texto e usa um dos índices que pode ser lista ligada ou árvore de busca binária. O algoritmo segue incluindo dois arquivos de cabeçalho, que são as funções da lista ligada e árvore de busca binária
Dentro da função “main” existe uma variável do tipo inteiro chamada “índice” que é atribuída inicialmente o valor zero, será utilizada para indicar o tipo de índice que deve ser criado pelo programa. Assim, se o usuário indicou “lista”, a variável índice é definida como 1 e uma nova lista ligada é criada.Por outro lado, se o usuário indicou"árvore",a variável "índice" é definida como 2 e uma árvore binária de busca é criada. No próximo passo o programa segue lendo linha a linha o arquivo texto passado pelo usuário.
Então, é alocado memória para a variável “linha” e é declarado a variável ”contador_linha” na qual será responsável por contar quantas linhas terá o arquivo texto. Dessa forma, o mecanismo do algoritmo segue armazenando as linhas do arquivo texto na “linha” e a função “strchr” localiza o(‘\n’) na string “linha”, caso encontre, ele é substituído por um caractere nulo, a variável “copia_ponteiro_linha tem a funcionalidade de garantir a integridade do conteúdo original do arquivo texto no funcionamento do algoritmo”e uma outra variável auxiliar é a “palavra”, que armazena as palavras separadas das strings do “copia_ponteiro_linha”.
Assim, conforme solicitado pelo usuário o programa usa a lista ligada ou árvore de busca para encontrar a palavra e, então, se prepara para buscar uma nova palavra ou encerrar e, é importante lembrar que as duas maneiras guardam as palavras em ordem alfabética e a busca é sequencial ou binária dependendo da estrutura

4.RESULTADOS

Os testes foram feitos com 4 textos, cada um com tamanhos de linhas e quantidade de palavras diferentes. Durante os testes, foi observado quanto tempo demorava para carregar o arquivo e construir o índice de palavras, além de observar o tempo que levava para buscar uma palavra dentro do texto, isso comparando os dois tipos estrutura de dados, a árvore e a lista.

Informações sobre os textos usados para teste:

Texto	Quantidade de Linhas	Total de palavras	Total de palavras distintas
Texto Padrão	13	119	69
Beatles - Hey Jude	26	202	81
Queen - Bohemian Rhapsody	94	369	156
Neil Gaiman - CTHULHU	233	2431	815

Cada texto possuía uma certa quantidade de linhas, um número total de palavras, e um número total de palavras distintas. O fator que mais importou para o desempenho da criação do índice foi o total de palavras diferentes, já que isso significava uma maior quantidade de inserções na lista/árvore, o que consumia mais tempo e memória do programa.
Podemos ver isso na seguinte tabela de tempo de processamento do arquivo e criação do índice em milissegundos:

Texto	Tempo Índice (Árvore)	Tempo Índice (Lista)
Texto Padrão	0,242	0,067
Beatles - Hey Jude	0,284	0,089
Queen - Bohemian Rhapsody	0,452	0,221
Neil Gaiman - CTHULHU	2,806	12,24

Conforme os textos possuíam mais palavras distintas o tempo de processamento (em milissegundos) aumentava.
Lista x Árvore: Em textos menores a lista obteve um melhor desempenho, já em textos grandes a árvore se mostrou muito mais eficiente que a lista.

Agora tabela de busca de palavras em milisegundos:

Texto	Tempo Busca (Lista)	Tempo Busca (Árvore)
Texto Padrão	0,018	0,004
Beatles - Hey Jude	0,019	0,002
Queen - Bohemian Rhapsody	0,032	0,004
Neil Gaiman - CTHULHU	0,229	0,003

Para o tempo de busca de uma palavra nas linhas do texto, o resultado foi que o número de linhas afetava diretamente o tempo de processamento.

Lista x Árvore: a árvore obteve o melhor resultado em todos os diferentes tamanhos de texto, sendo a opção mais rápida de busca, grande parte disso devido a sua característica de busca binária, que a deixa muito mais eficiente.

5.DISCUSSÃO

Com os dados obtidos podemos dissertar sobre a eficiência dos dois tipos de estrutura de dados em relação às distintas operações feitas. É possível perceber que as duas estruturas possuem suas vantagens e desvantagens e se sobressaem em diferentes aspectos.
Ainda que no geral a Árvore tenha obtido melhores resultados nos testes do nosso trabalho, os testes foram apenas para criação de índice e operação de busca, ainda existem outras operações que não foram usadas nesse trabalho mas que também devem ser avaliadas, como exclusão, classificação de itens, entre outros.
Conforme o objetivo do programa muda, uma estrutura de dados pode ser uma opção melhor em relação a outra, tanto no desempenho e eficiência dela, mas também como na complexidade, já que também deve ser avaliado o custo benefício da solução de um determinado problema.

6.CONCLUSÃO

Dessa forma, usando como base o conhecimento teórico sobre as estruturas de dados utilizadas nesse trabalho, e também os testes realizados, podemos tirar algumas conclusões.
Na operação de criação de índice das palavras, foram armazenadas todas as palavras distintas presentes no texto para facilitar a realização da busca, essa operação quando feita com as palavras sendo armazenadas em uma lista ligada havia um melhor desempenho para os textos com menor quantidade de palavras distintas, já conforme o número de palavras diferentes aumentava, a árvore se tornava progressivamente mais eficiente do que a lista.
Na operação de busca de palavras, os dados mostraram que árvore de busca binária tinha um desempenho muito melhor do que a lista, isso se aplicava a todos os tamanhos de texto testados, de forma que demonstra que a busca binária utilizada melhorava a rapidez em que a palavra buscada era encontrada e a linha que ela estava presente.

Assim, os resultados obtidos nos testes condizem com a teoria estudada durante as aulas de algoritmos e estruturas de dados, a complexidade da árvore de busca binária tem como caso médio $O(\log n)$ e seu pior caso $O(n)$, já a lista ligada apresenta seu caso médio e pior caso como $O(n)$. Ainda assim, provavelmente devido à complexidade da estrutura, a lista ainda se sai melhor na operação de criação com pequenas quantidades de palavras, em todos os outras situações a melhor opção é a árvore de busca binária.

7.REFERÊNCIAS BIBLIOGRÁFICAS