TRIE

Universidade Federal de Sergipe
Disciplina: Estrutura de Dados
Docente: Alberto Costa Neto
Discentes:
Cainã Castro Aquino
Edcarlos dos Santos Ramos
Leandro Santos Lima
Riquelme Prado Leite

UNIVERSIDADE FEDERAL DE SERGIPE

Situação Problema: Como buscar uma palavra em um dicionário ?

Utilizando a busca sequencial, no pior caso, onde a palavra está no final do dicionário, seriam percorridas todas as palavras até chegar na palavra desejada (que também pode ser a última).

Utilizando a busca binária seria possível eliminar 50% das opções a cada operação de busca, mas não seria o método mais eficiente.

Como buscar de uma forma mais eficiente ?

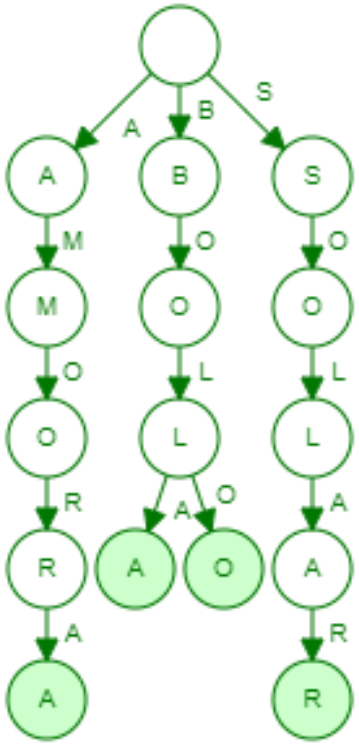
- Procurando cada palavra utilizando seu prefixo é uma forma eficiente de se realizar essa busca!
- A cada letra percorrida são eliminadas todas as outras com suas palavras diferentes.
- Por exemplo ao ler a primeira letra haveria uma porcentagem de Redução = (1 / 26) * 100 ≈ 3.846% na opção de primeiro nível (considerando o alfabeto com letras minúsculas e sem pontuações)

TRIE

Estrutura de Dados de pesquisa , utilizada com strings que é representada como uma árvore de múltipla ramificação onde seus índices são caracteres de um alfabeto com um nó terminal sinalizando o fim de uma determinada palavra.

TABELA ASCII

DEC	OCT	HEX	BIN	Símbolo
65	101	41	01000001	A
66	102	42	01000010	B
67	103	43	01000011	C
68	104	44	01000100	D
69	105	45	01000101	E
70	106	46	01000110	F



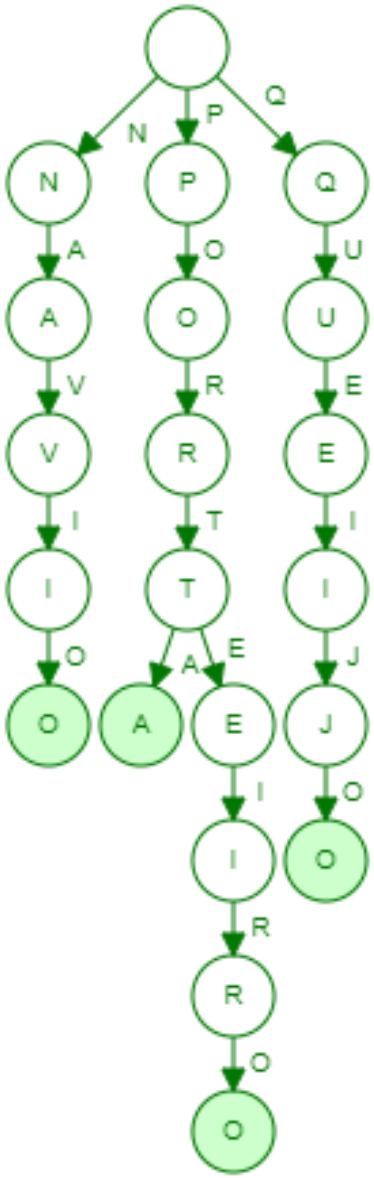
TRIE

Navio

Porta

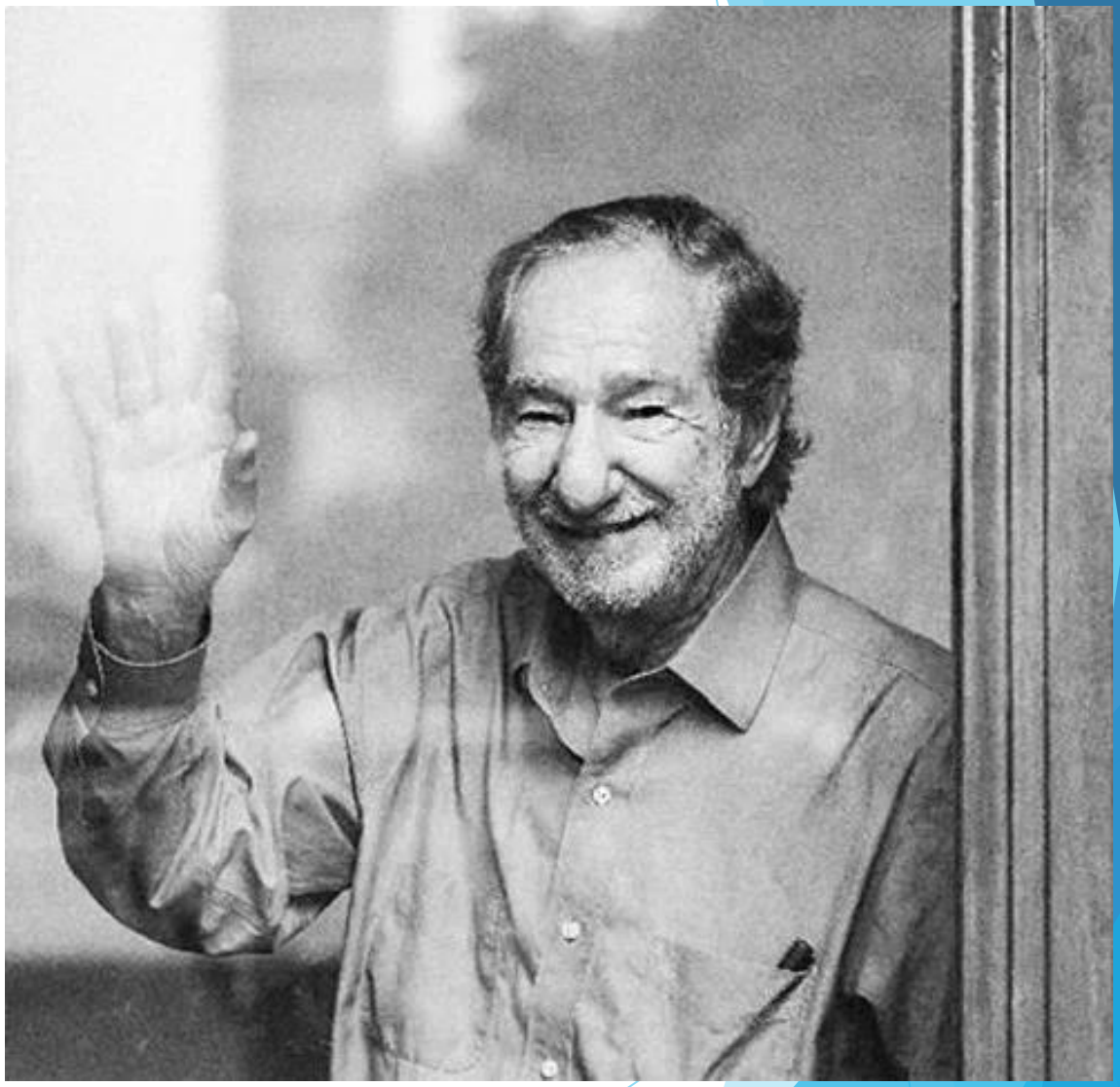
Porteiro

Queijo



TRIE

Trie (pronunciado "trai" ou "try") vem do termo "retrieval" e foi sugerido por Edward Fredkin na decada de 1960.
Trie é uma árvore de busca digital mais específica pois seus nós não armazenam diretamente os valores das chaves, mas sim caminhos que seguem por elementos da chave até o seu último elemento.



Edward Fredkin, Físico Estadunidense Criador da TRIE.

Por que usar uma TRIE ?

Vantagens:

- Busca extremamente eficiente
- Inserção Rápida
- Eficiente para armazenar strings longas

Desvantagens:

- Grande uso de memória
- Ineficiente com palavras muito diferentes
- Possui uma implementação complicada

Complexidade nas operações em uma Trie :

O cálculo de complexidade das funções da Trie é O(n), onde n é o tamanho da cadeia que está sendo analisada.

Já o cálculo da complexidade do espaço de uma Trie é O(N*M*C) onde N é a quantidade total de cadeias, M é o comprimento máximo da cadeia e C o tamanho do alfabeto.

Onde é possível encontrar utilizações de TRIES na programação ?

Mecanismos de auto complementar

Corretores ortográficos

Sistemas de busca de textos

Como é ordenada ou localizada uma determinada String ?

- Diferente de alguns outros tipos de estrutura de dados a Trie não ordena seus elementos baseando-se na comparação entre as strings, isso por que a ordenação é realizada olhando especificamente os elementos das strings na hora da inserção.
- Partindo desse preceito, o custo de inserção ou de busca em uma Trie é o mesmo independentemente da quantidade de strings que já estão inseridas.
- O que realmente muda o custo é o tamanho da string que se está tentando inserir.

Inserção em uma TRIE :

Passo a Passo:

- 1° : Inicializa uma raiz vazia
  e com possíveis derivações para todos caracteres do alfabeto.
- 2° : Caso não exista,
  adiciona um nó na direção dos caracteres a serem inseridos.
- 3° : Caso exista,
  utiliza os prefixos já existentes na árvore até chegar a um ponto onde não existe.
- 4° : Ao chegar no último caractere
  também é adicionado valor terminal (ou final) ao nó.

Busca em uma TRIE :

Passo a Passo:

- 1º : Comece na raiz da Trie, que é o nó que representa a string vazia ou o prefixo vazio.
- 2º : Observe o primeiro caractere da string de busca: Pegue o primeiro caractere da string que você deseja buscar e procure o nó filho correspondente.
- 3º : Caso encontre o nó filho correspondente repita o processo para os filhos até o final da string, caso não encontre em algum momento significa que a string não pertence a trie.
- 4º : Ao chegar ao final da string deve checar se o último caractere é terminal.

Remoção em uma TRIE :

Passo a Passo:

- 1º : Comece fazendo uma busca para garantir que a string a ser apagada pertence a trie.
- 2º : Checar se o nó terminal possui filho, se não possuir, o estado deixa de ser terminal e apagamos o caractere, se possuir, o estado apenas deixa de ser terminal e interrompemos a remoção.
- 3º : Repetir o processo para todos os nós a cima que deram origem ao nosso nó terminal checando também desta vez se o nó é final.
- 4º : Para o caso desses nós a cima, serem terminais ou possuírem filhos interrompemos a remoção.

Listagem em uma TRIE :

Passo a Passo:

- 1º : Visitar cada nó da subárvore mais à esquerda da trie, guardando a informação de cada caractere.
- 2º : Checar se o nó é terminal, caso seja, concatenar os caracteres que deram origem a esse nó e exibir a string final.
- 3º : Repetir o processo para todas as subárvores da raiz, dá mais à esquerda para mais à direita, para que seja respeitada a ordem definida.

Fontes de Pesquisa:

1 - Definição Trie : https://datastructures.maximal.io/tries/

2 - Código e cálculo da complexidade das funções de uma TRIE/ Aplicações : Implementação Trie em C – Inserir, Pesquisar e Excluir (techiedelight.com)

3 – Imagens exemplos:
https://www.cs.usfca.edu/~galles/visualization/Trie.html