Implementação de um indexador de arquivos

Isauflânia Suelen Ribeiro Timóteo
Leandro Furlam Turi

Considerações iniciais

Sobre os Tipos Abstratos de Dados utilizados

- Palavra: aglomerado de caracteres alfa-numérico e *case sensitive*;
- Maior palavra:

PNEUMOULTRAMICROSCOPICOSSILICOVULCANOCONIÓTICO

46 letras

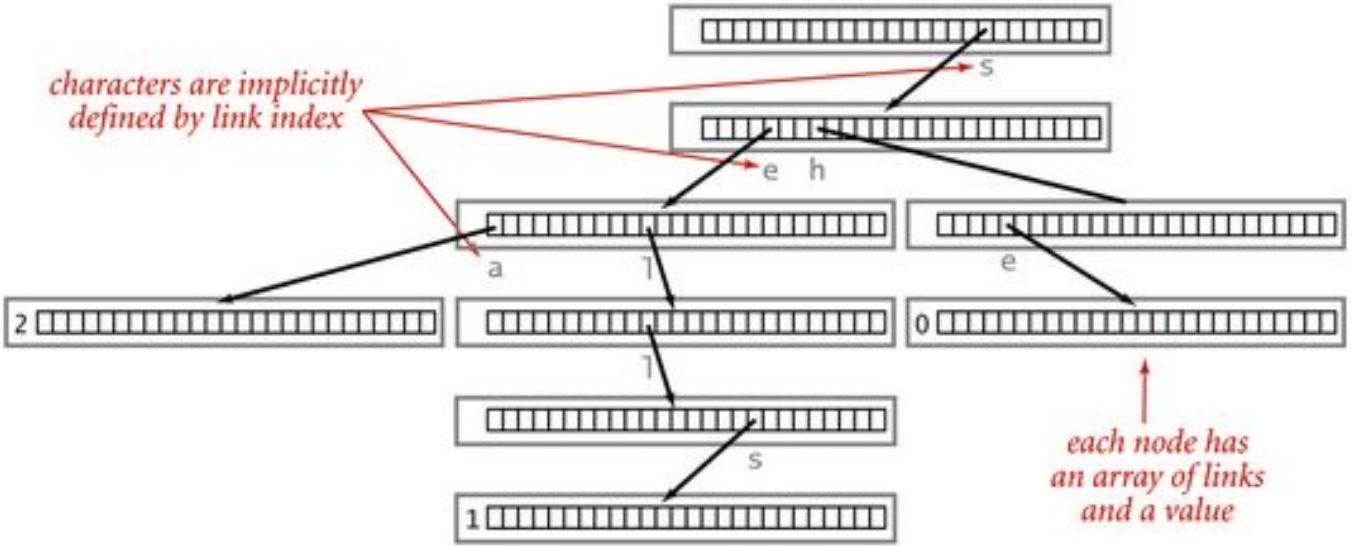
Tipo Palavra

- Palavra;
- Quantidade;
- Tipo Arquivo.

Tipo Arquivo

- Identificador;
- Quantidade de ocorrências da palavra;
- Posições de ocorrência.

- Lista Encadeada;
- Árvore Binária;
- Árvore Binária Balanceada;
- Árvore Trie;
- Tabela Hash.



Trie representation ($R = 26$)

Árvore Trie

INSTITUTO DE MATEMATICA E ESTATISTICA - USP. Tries (árvores digitais). 2002. Disponível em: <https://www.ime.usp.br/~pf/estruturas-de-dados/aulas/tries.html>.

Tabela Hash

É recomendável que M seja um número primo!

k	2^k	M
7	128	127
8	256	251
9	512	509
10	1024	1021
11	2048	2039
12	4096	4093
13	8192	8191
14	16384	16381
15	32768	32749
16	65536	65521
17	131072	131071
18	262144	262139

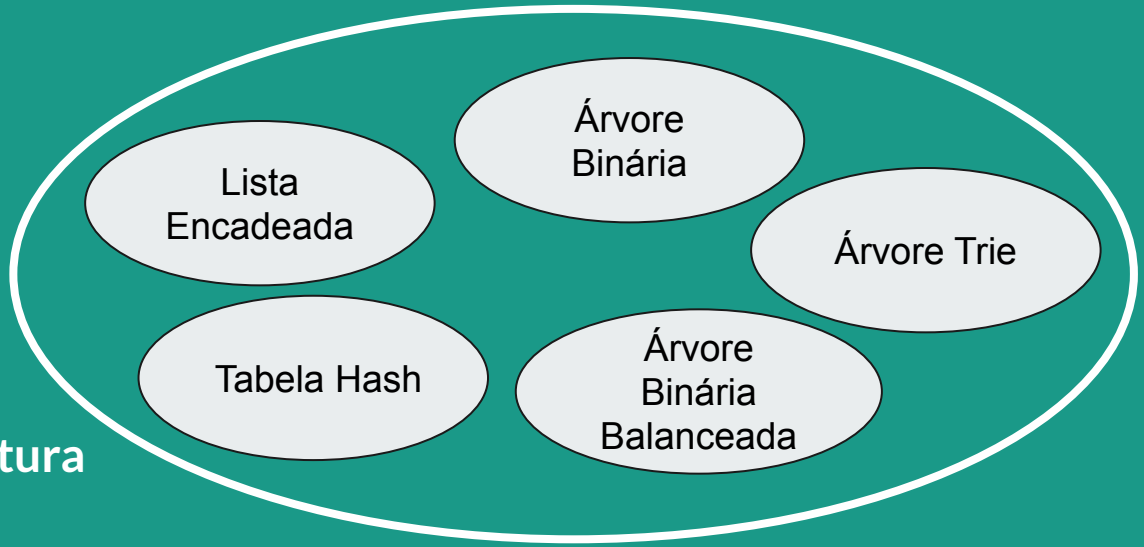
INSTITUTO DE MATEMATICA E ESTATISTICA - USP. Tabelas de dispersão (hash tables). 2002. Disponível em: <https://www.ime.usp.br/~pf/mac0122-2002/aulas/hashing.html>.

Tabela Hash

```c
int hash (char *palavra)
{
    int h;
    h = 1;
    for (int i = 0; palavra[i] != '\0'; i ++)
    {
      h = (h * 251 + palavra[i]) % M;
    }
    return h;
}
```

INSTITUTO DE MATEMATICA E ESTATISTICA - USP. Tabelas de dispersão (hash tables). 2002. Disponível em: <https://www.ime.usp.br/~pf/mac0122-2002/aulas/hashing.html>.

ENCAPSULAMENTO



Tipo Estrutura

Avaliação de Desempenho

```
void avaliaDesempenho (char** caminhosArq, int qtd, int n);
```

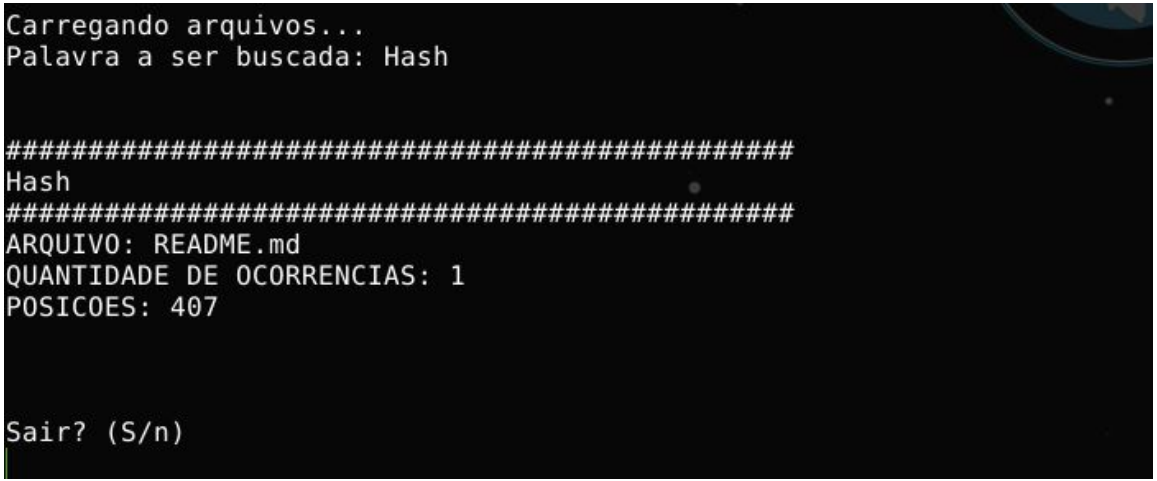
```
lfurlam@relampago:~/Dropbox/Ufes/ED1/trab2-ed1$ ./indexador README.md  10

NUMERO DE BUSCAS: 10
                ENCADEADA   ARVORE      AVL         TRIE        HASH
CARREGAMENTO    0.000858    0.000496    0.001100    0.000319    0.000099
BUSCA           0.000000    0.000013    0.000005    0.000005    0.000004
```

Busca de palavras legais

```
void buscaPalavra (char **caminhosArq, int qtd);
```

Busca de palavras legais



Dúvidas sobre como utilizar?

README.md

Trabalho 2 de Estruturas de Dados 1.
Indexador de arquivos.

O objetivo deste trabalho é utilizar diversas estruturas de dados para indexar o conteúdo de arquivos, além de analisar o desempenho de buscar palavras.
Possui funções de análise geral e de busca comum.

Estruturas utilizadas:
* Lista Encadeada;
* Árvore Binária;
* Árvore Binária Balanceada (AVL);
* Árvore Trie;
* Tabela Hash.

Execução (Depende):
1. Buscar uma palavra legal:

`$ ./indexador <caminho do arquivo0> <caminho do arquivo1> ...`

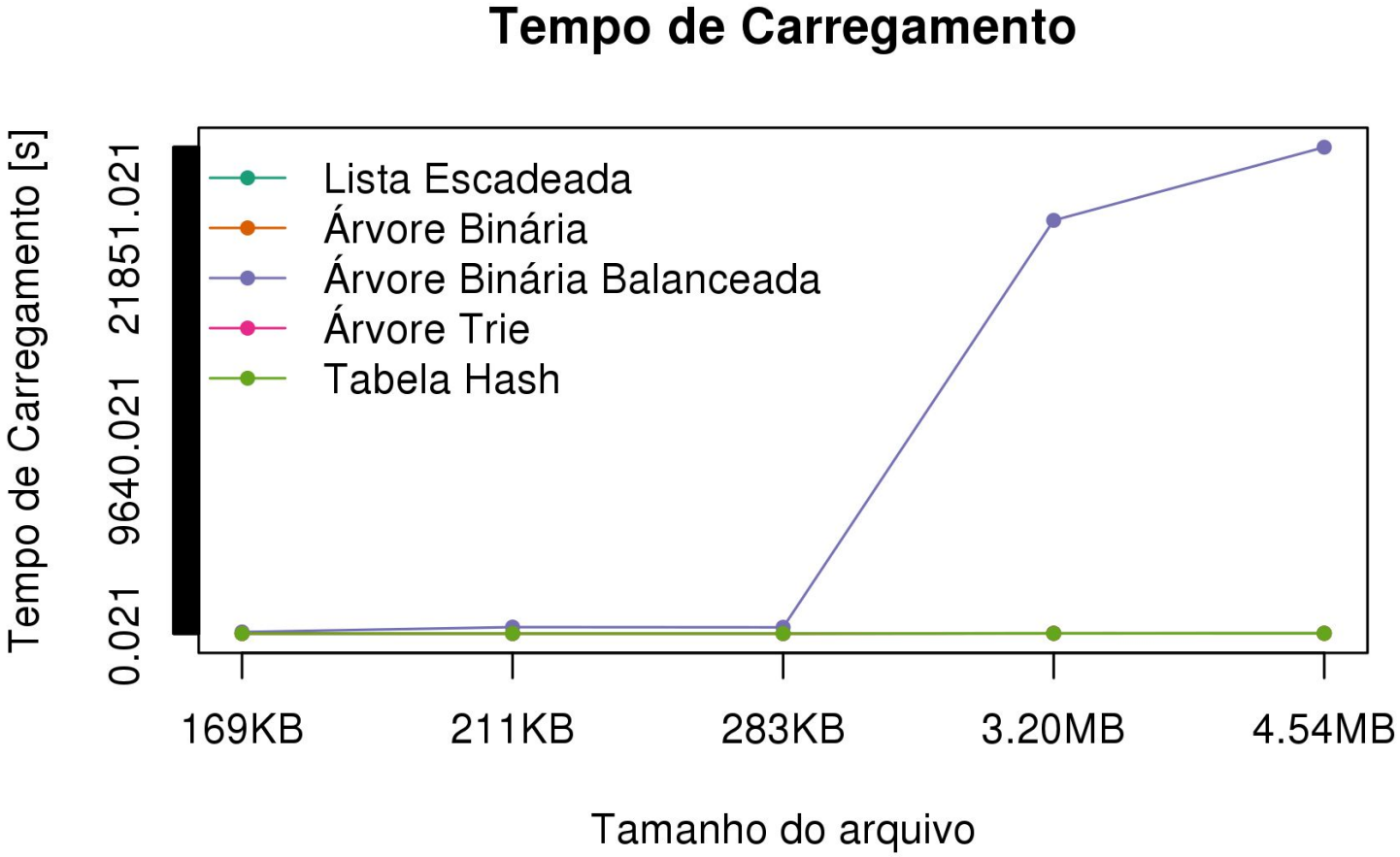
Após a execução será pedida a estrutura que deseja carregar os arquivos e a palavra a ser buscada.

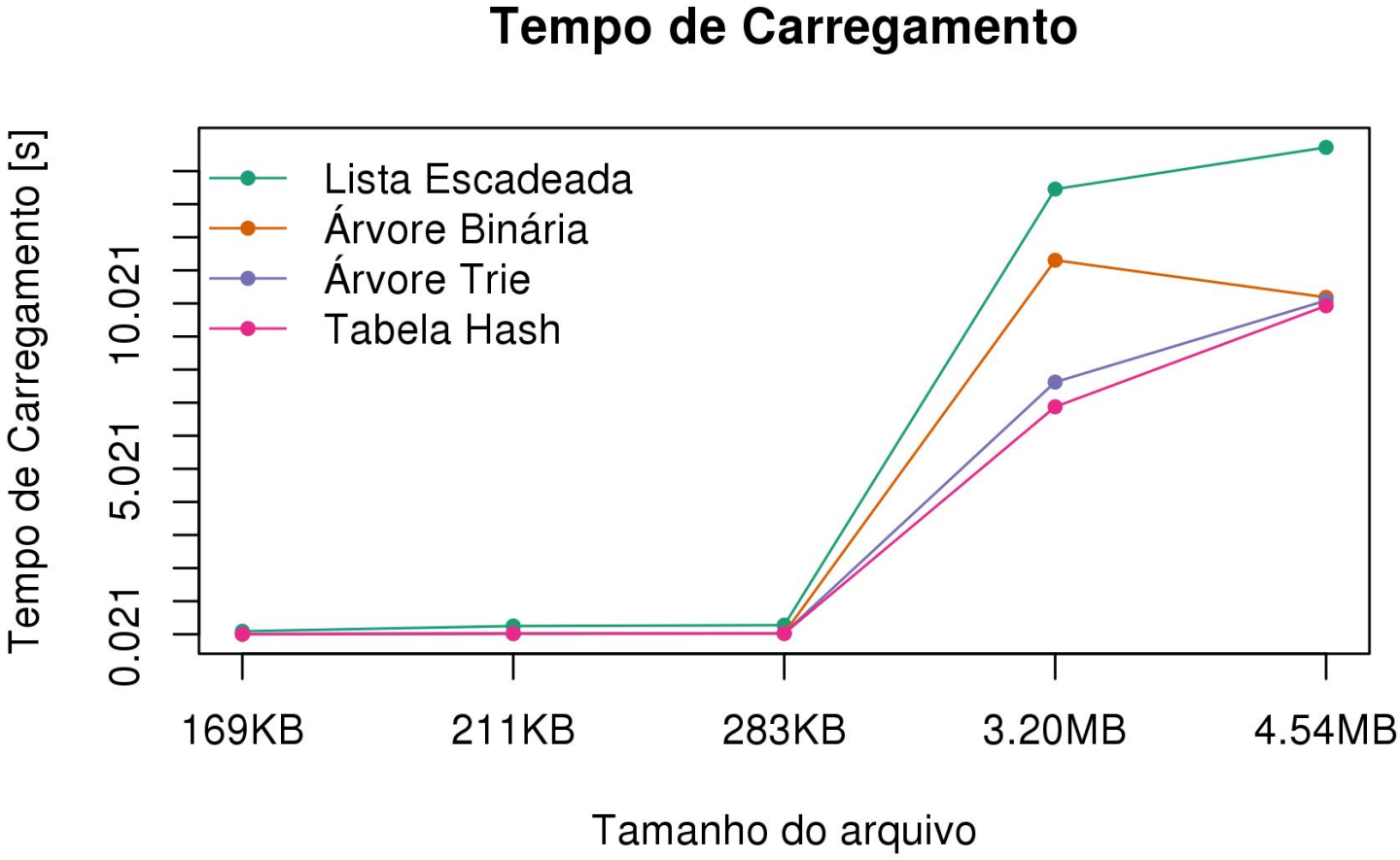
2. Analisar o desempenho:

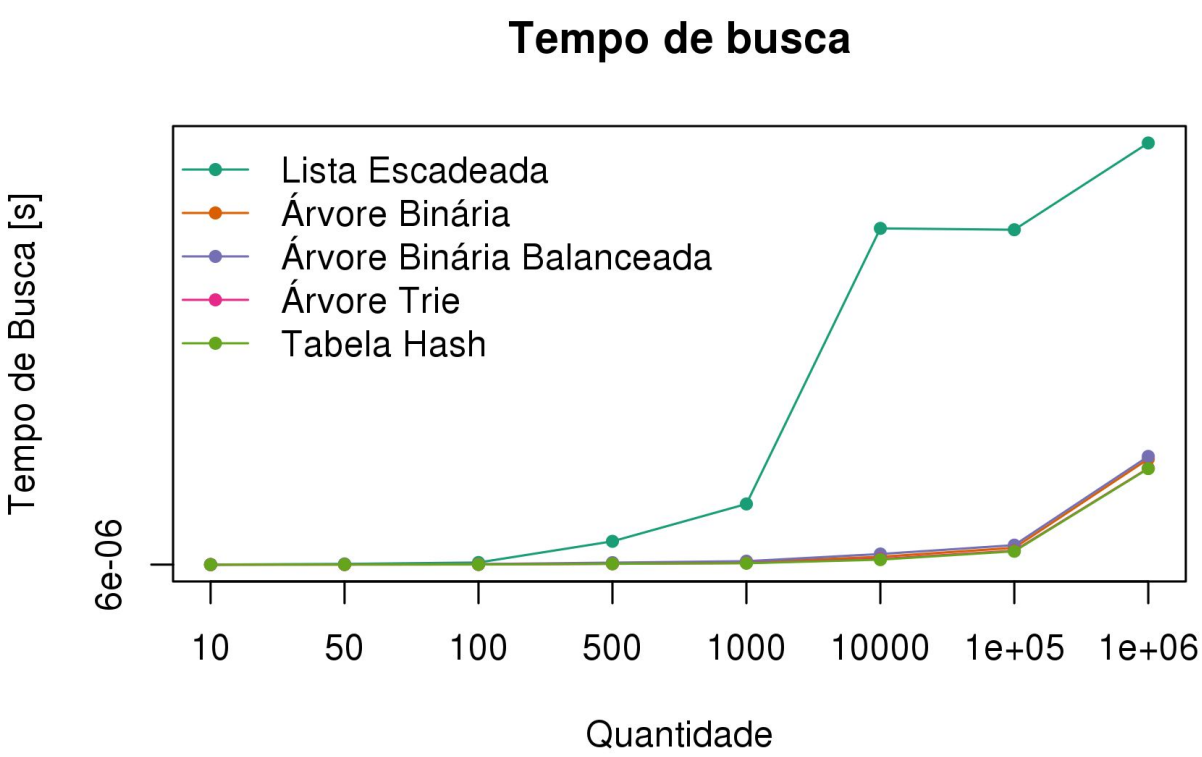
`$ ./indexador <caminho do arquivo0> <caminho do arquivo1> ... <tamanho da busca>`

RESULTADOS

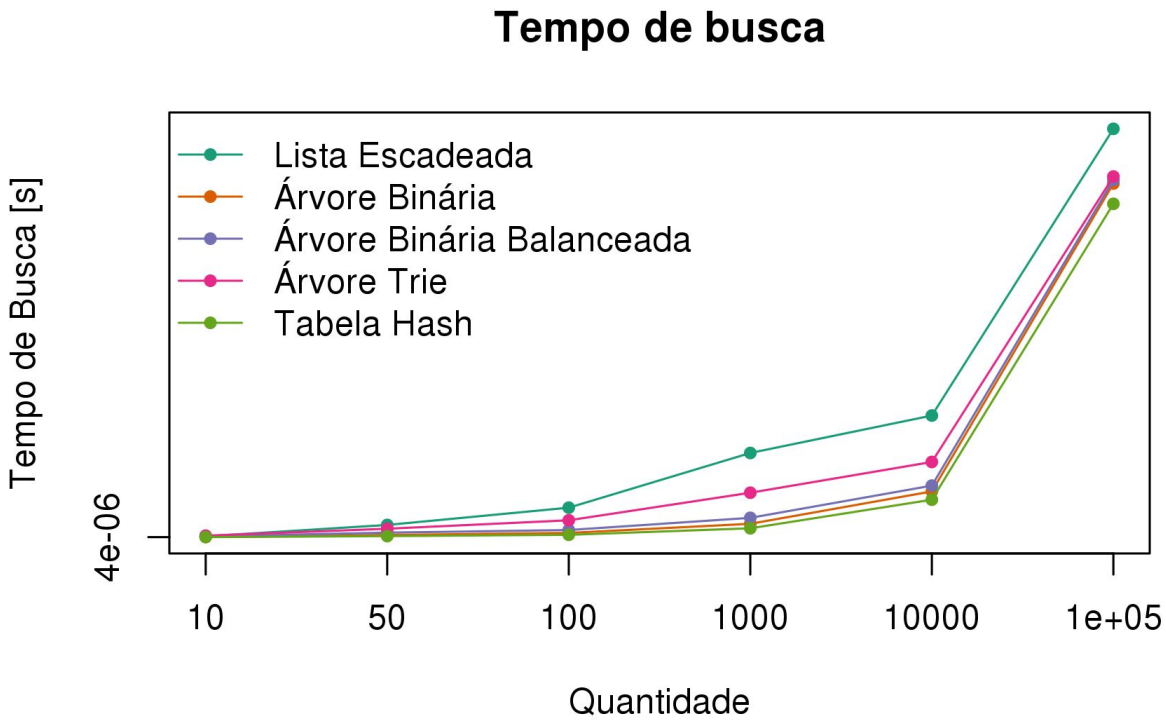
- 1000 palavras: Lorem Ipsum;
- 169KB: Alice's Adventures in Wonderland;
- 211KB: Quincas Borba;
- 283KB: Peterpan;
- 3.20MB: War and Piece;
- 4.54MB: Bibble.


Tempo de Carregamento
Tempo de Carregamento [s]
0.021
9640.021
21851.021
Lista Escadeada
Árvore Binária
Árvore Binária Balanceada
Árvore Trie
Tabela Hash
169KB
211KB
283KB
3.20MB
4.54MB
Tamanho do arquivo


Tempo de Carregamento
Tempo de Carregamento [s]
0.021
5.021
10.021
Lista Escadeada
Árvore Binária
Árvore Trie
Tabela Hash
169KB
211KB
283KB
3.20MB
4.54MB
Tamanho do arquivo



Alice's Adventures in Wonderlanderland


Tempo de busca
Lista Escadeada
Árvore Binária
Árvore Binária Balanceada
Árvore Trie
Tabela Hash
Tempo de Busca [s]
4e-06
10
50
100
1000
10000
1e+05
Quantidade

Lorem Ipsum

IT’S THAT ALL