

Aula 16 – Árvores Patricia e Árvores B

Prof. Aléssio Miranda Jr
alessio@cefetmg.br

2Q-2017

TRIE originado de 'Information reTRIEval'

TRIE = digital tree = radix tree = prefix tree

Trie Memory*

EDWARD FREDKIN, *Bolt Beranek and Newman, Inc., Cambridge, Mass.*

Introduction

Trie memory is a way of storing and retrieving information.¹ It is applicable to information that consists of function-argument (or item-term) pairs—information conventionally stored in unordered lists, ordered lists, or pigeonholes.

The main advantages of trie memory over the other memory plans just mentioned are shorter access time, greater ease of addition or up-dating, greater convenience in handling arguments of diverse lengths, and the ability to take advantage of redundancies in the information stored. The main disadvantage is relative inefficiency in using storage space, but this inefficiency is not great when the store is large.

In this paper several paradigms of trie memory are described and compared with other memory paradigms, their advantages and disadvantages are examined in detail, and applications are discussed.

is simply the binary variable of which the admissible values are *member* and *nonmember*.

At the outset, before a storage is begun, the trie is merely a collection of *registers*. Except for two special registers, which we may call α and δ , every register has a *cell* for each member (type) of the ensemble of alphabetic characters. If we let that ensemble include a "space" to indicate the end of a word (argument), each register must have 27 cells.

Each cell has space for the address of any register in the memory. Cells in the trie that are not yet being used to represent stored information always contain the address of the special α register. A cell thus represents stored information if it contains the address of some register other than α . The information it represents is its own name, "A" for the A cell, "B" for the B cell, etc., and the address of the next register in the sequence.

Storage of words of alphabetic characters is illustrated in Fig. 1. For the sake of simplicity the ensemble of



Published in:



Magazine

Communications of the ACM [CACM Homepage](#) [archive](#)

Volume 3 Issue 9, September 1960

Pages 490-499

[ACM](#) New York, NY, USA

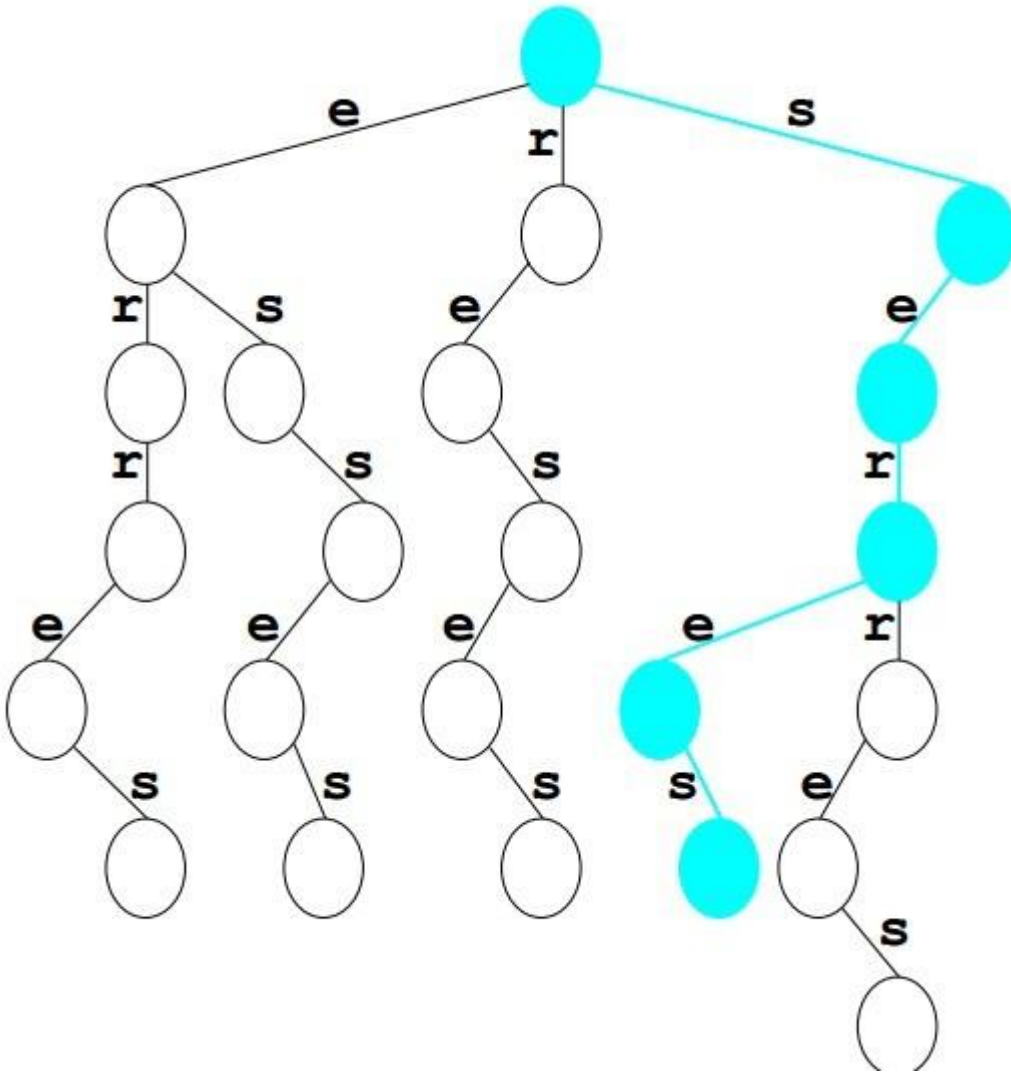
[table of contents](#) doi > [10.1145/367390.367400](#)

Edward Fredkin

Born	October 2, 1934 (age 80) ^[citation needed] Los Angeles
Residence	Brookline, MA
Citizenship	USA
Nationality	USA
Fields	Computer Science, Physics, Business,
Institutions	Massachusetts Institute of Technology Carnegie Mellon University Capital Technologies, Inc.
Alma mater	(Caltech)
Known for	Trie data structure, Fredkin gate
Notable awards	Dickson Prize in Science 1984

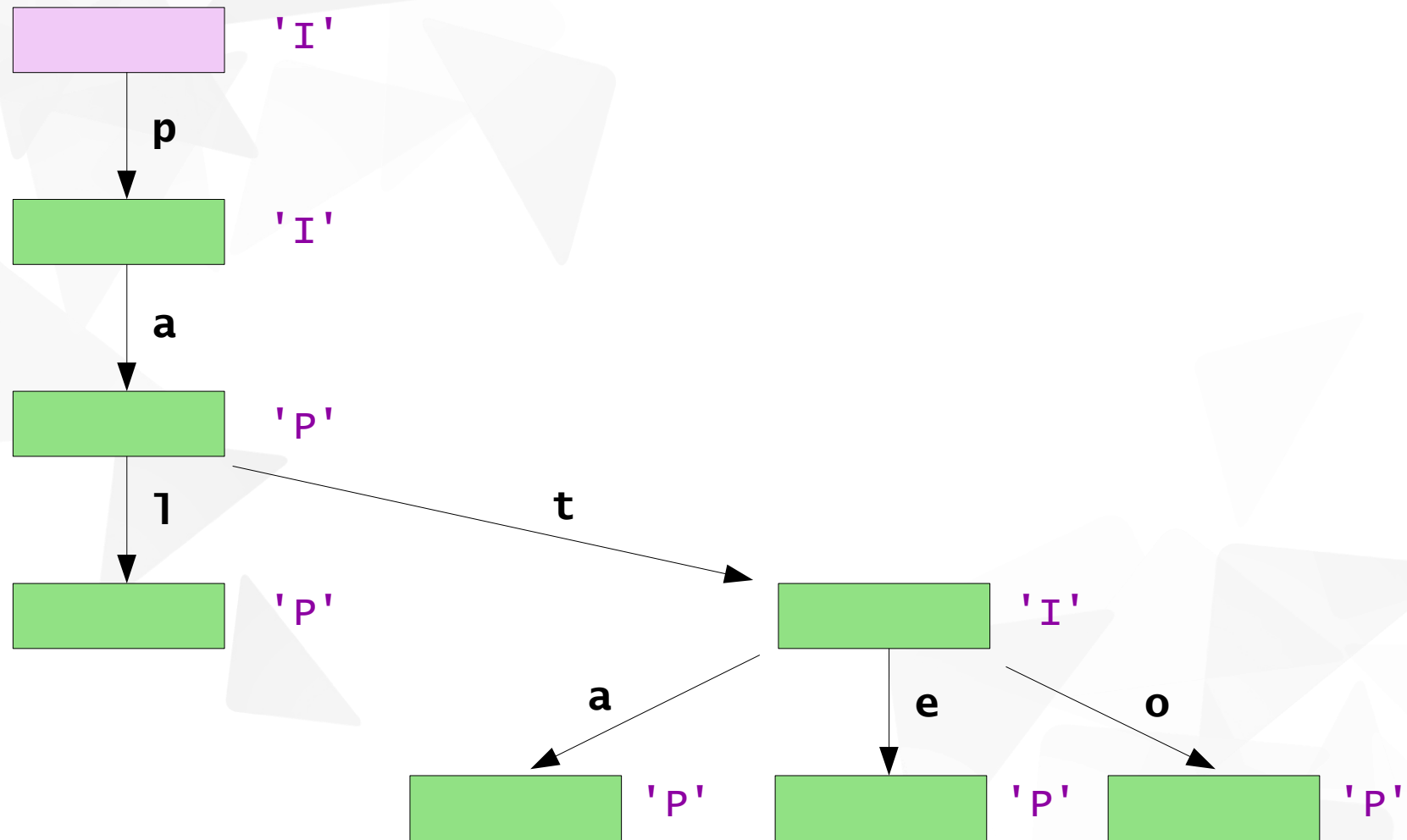
Exemplo de árvore TRIE (árvore m-ária)

Alfabeto:
 $\{e, r, s\}$



erre
erres
es
esse
esses
se
ser
serre
re
res
rese
reses
serres
seres

Trie



Complexidade computacional

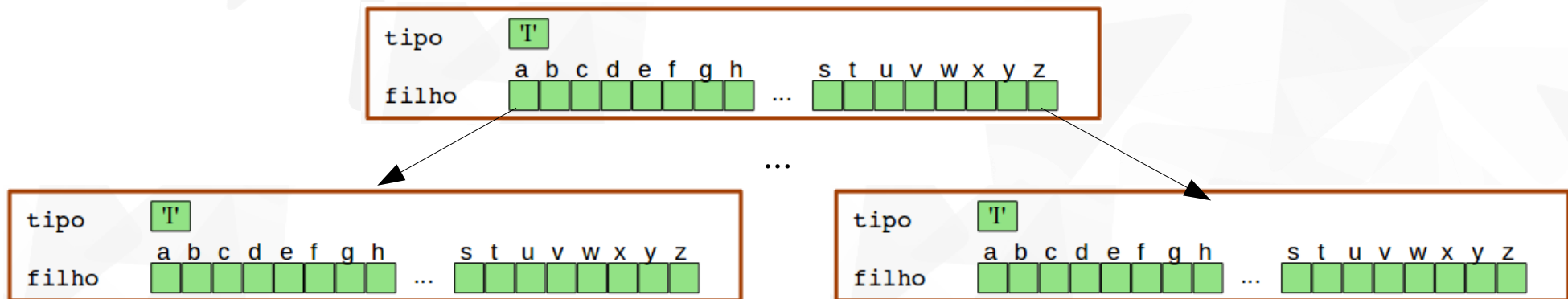
- **Busca, Inserção e Remoção:**

Para uma palavra p de tamanho $|p|$: $O(|p|)$

Independente do número total de palavras.

- **Uso de memória:**

Ineficiente para armazenar todas as palavras.



Árvores P.A.T.R.I.C.I.A.

acrônimo de

Practical **A**lgorithm **T**o **R**etrieve
Information **C**oded **I**n **A**lphanumeric

PATRICIA

PATRICIA—Practical Algorithm To Retrieve Information Coded in Alphanumeric

DONALD R. MORRISON

Sandia Laboratory, Albuquerque, New Mexico*

ABSTRACT. PATRICIA is an algorithm which provides a flexible means of storing, indexing, and retrieving information in a large file, which is economical of index space and of reindexing time. It does not require rearrangement of text or index as new material is added. It requires a minimum restriction of format of text and of keys; it is extremely flexible in the variety of keys it will respond to. It retrieves information in response to keys furnished by the user with a quantity of computation which has a bound which depends linearly on the length of keys and the number of their proper occurrences and is otherwise independent of the size of the library. It has been implemented in several variations as FORTRAN programs for the CDC-3600, utilizing disk file storage of text. It has been applied to several large information-retrieval problems and will be applied to others.

KEY WORDS AND PHRASES: indexing, information retrieval, keys

CR CATEGORIES: 3.7

1. Context and Purpose

Libraries and files are constructed to store information. Examples include personnel files, telephone directories, part lists, dictionaries (scientific or technical), and general libraries. The user of a large library or file approaches the library in search of information. We call a piece of information sought by a user a *target*. Typically, a target is a book, a poem, a chapter, an article, a definition, a theorem, a biography,



PATRICIA—Practical Algorithm To Retrieve Information Coded in Alphanumeric

Full Text: PDF Buy this Article

Author: [Donald R. Morrison](#) Sandia Laboratory, Computer Science, Division 5256, Albuquerque, New Mexico



1968 Article

Published in:

· Journal
· Journal of the ACM (JACM) [JACM Homepage](#) [archive](#)
Volume 15 Issue 4, Oct. 1968
Pages 514-534
[ACM](#) New York, NY, USA
[table of contents](#) doi> [10.1145/321479.321481](#)

[Bibliometrics](#)

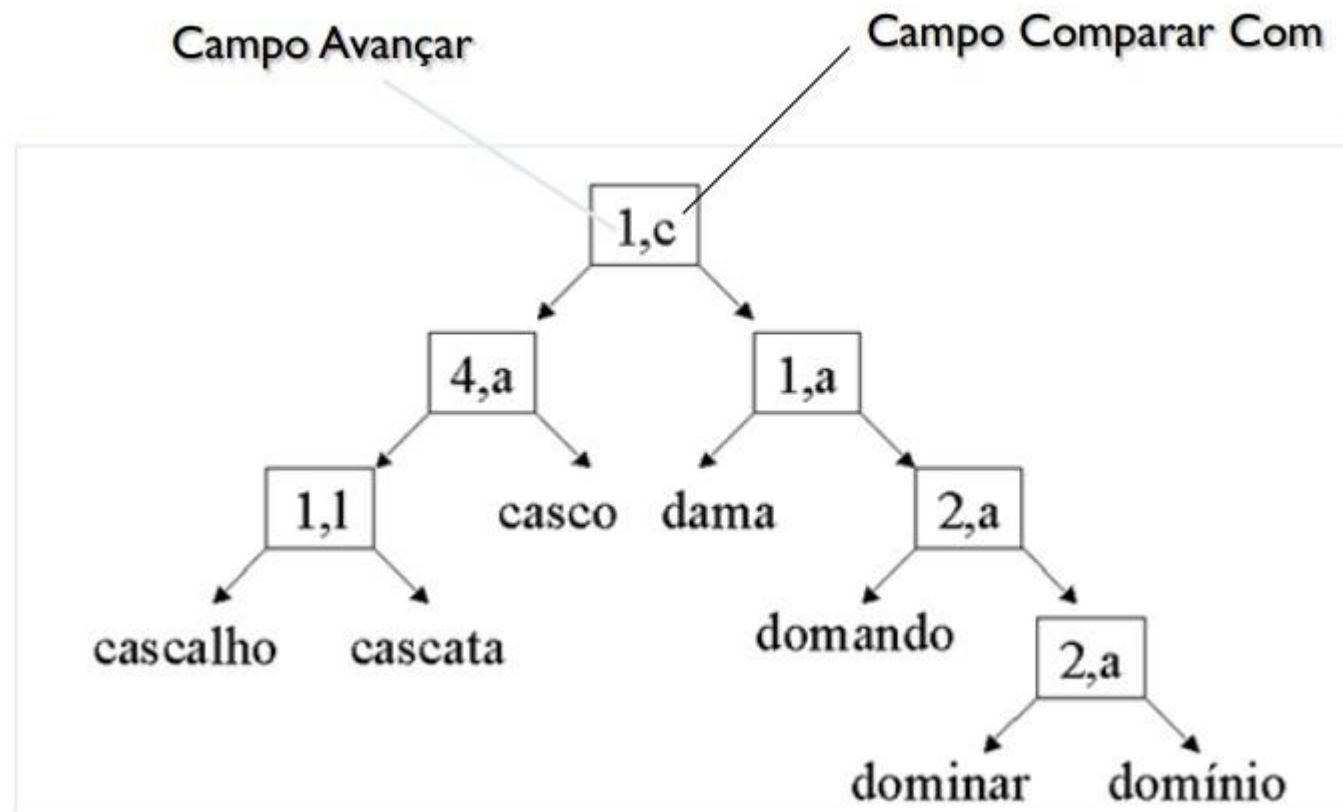
· Downloads (6 Weeks): 52
· Downloads (12 Months): 295
· Downloads (cumulative): 5,495
· Citation Count: 180

Cited by 946

Árvores PATRICIA

- TRIE compactada binária.
- **Caminhos que possuem nós com apenas 1 filho são agrupados em uma única aresta.**
- Diferente das TRIE, não armazena informações nos nós internos, apenas contadores e ponteiros para cada sub-árvore descendente.
Ou seja, nenhuma chave é prefixa de outra.

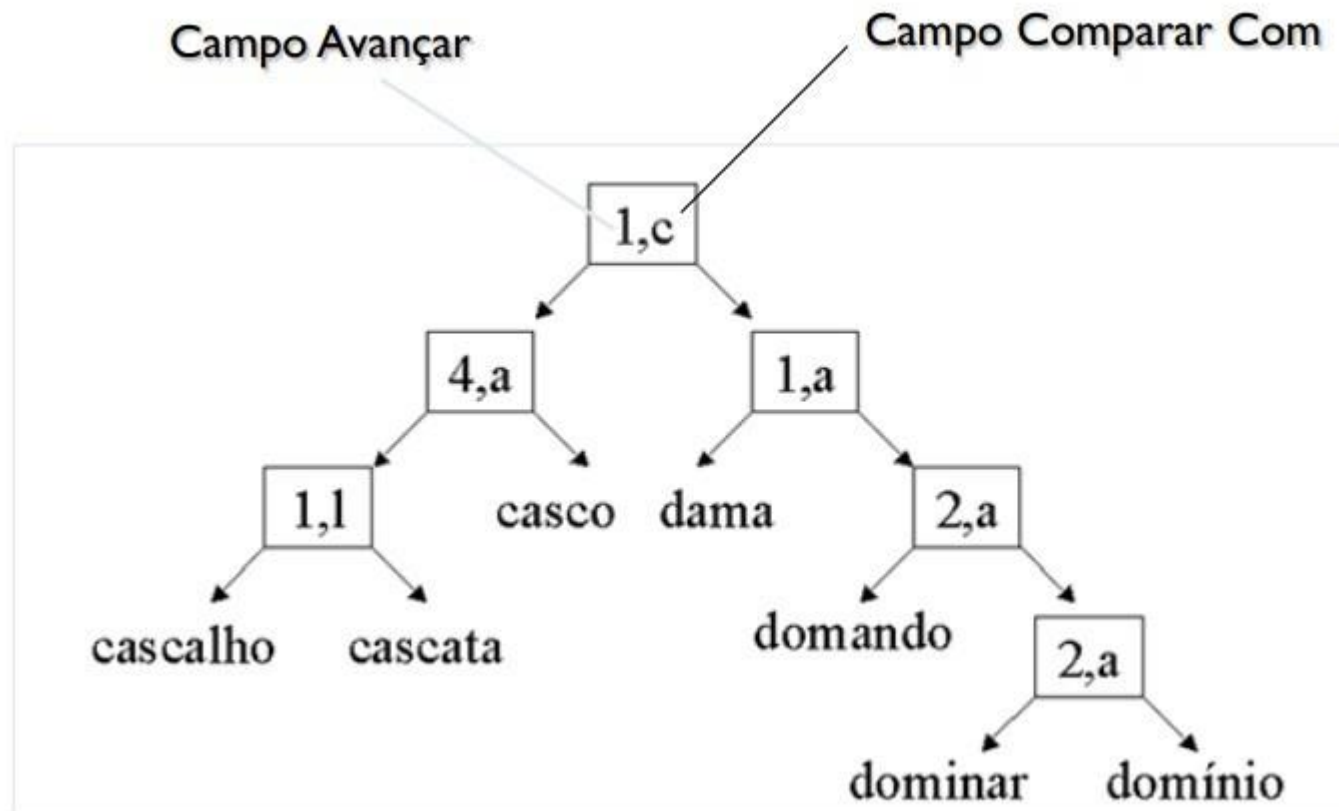
PATRICIA: Exemplo de representação



PATRICIA: Exemplo de representação

Registro acumulativo que integra todos os nós exeto nas folhas.

Identifica qual a posição do caractere da chave informada que deve ser analisado



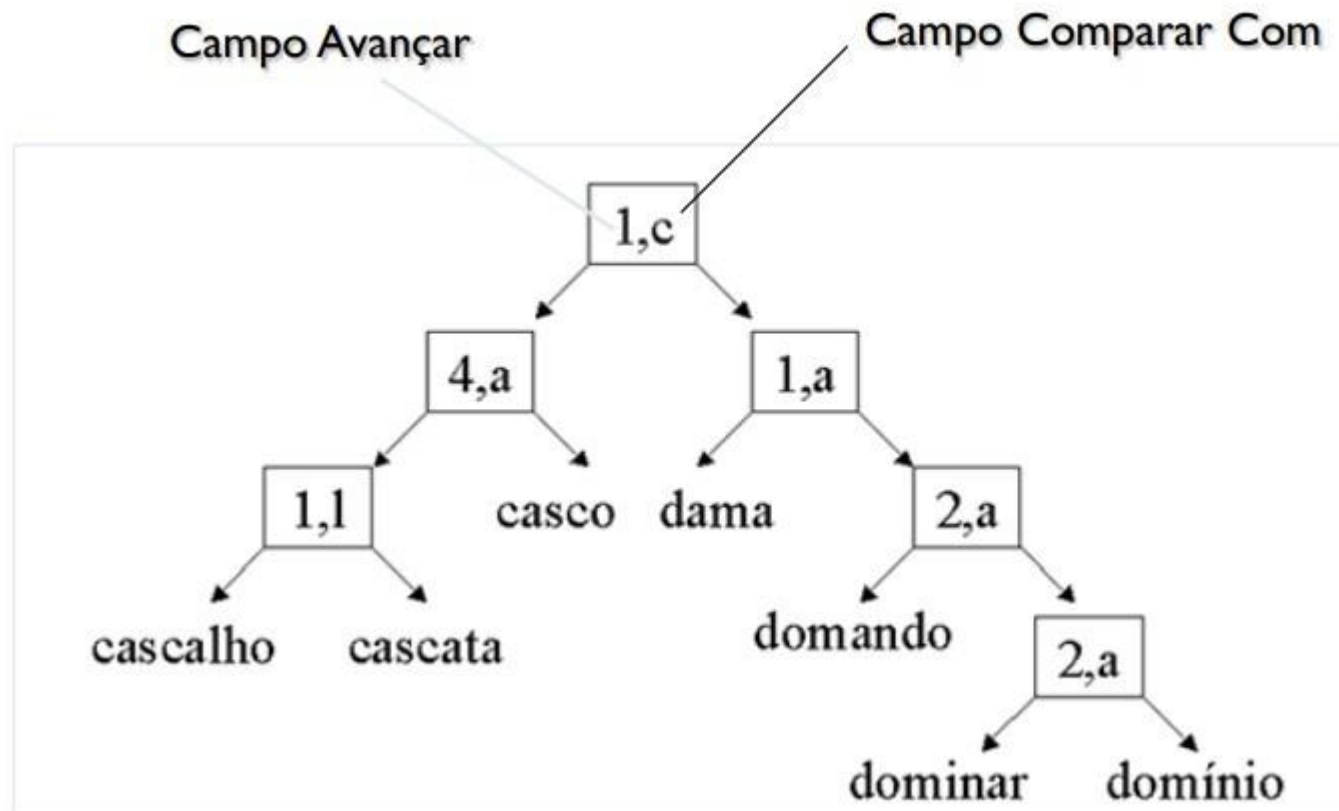
PATRICIA: Exemplo de representação

Registro acumulativo que integra todos os nós exeto nas folhas.

Identifica qual a posição do caractere da chave informada que deve ser analisado

Indica o caractere que deve ser comparado ao caractere da chave informada.

Similar às ABBs: Se a chave é menor ou Igual ao nó, ela é consultada à esquerda, caso contrário, à direita.



PATRICIA: Exemplo de inserção

Palavra 1 = Consultório

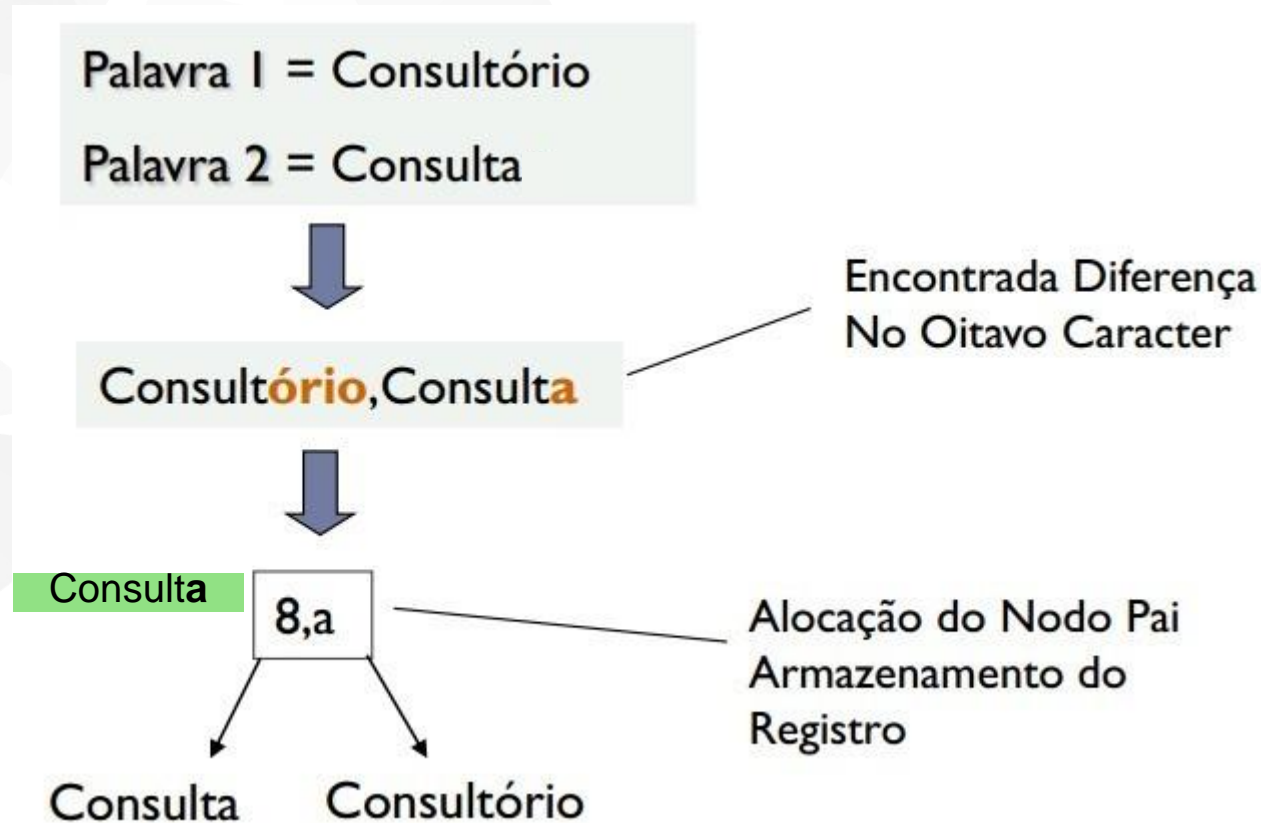
Palavra 2 = Consultar



Consultório, Consulta

Encontrada Diferença
No Oitavo Character

PATRICIA: Exemplo de inserção



PATRICIA: Exemplo de inserção

Palavra I = Consulado



Consulta

8,a

Consulta

Consultório

PATRICIA: Exemplo de inserção

Palavra I = Consulado



Consulta

8,a

Consulta

Consultório

Consulado
Consulta

PATRICIA: Exemplo de inserção

Palavra I = Consulado



Consulta

8,a

Consulta

Consultório

Consulado
Consulta



Consula

7,a

Consulado

Consulta

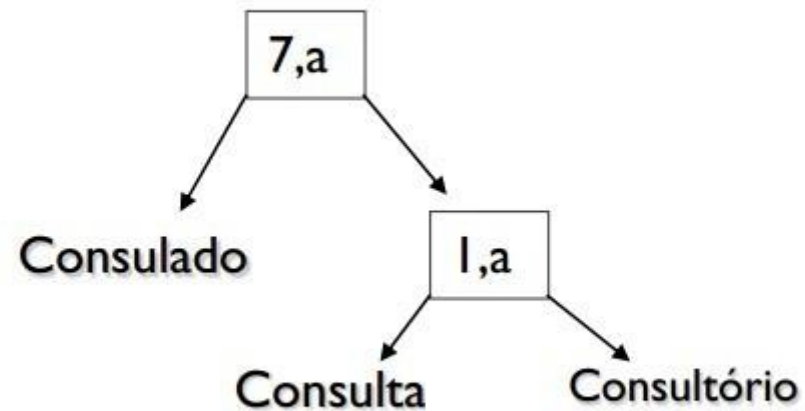
I,a

Consultório

Alocação do Nodo para
Alocação da Nova Palavra

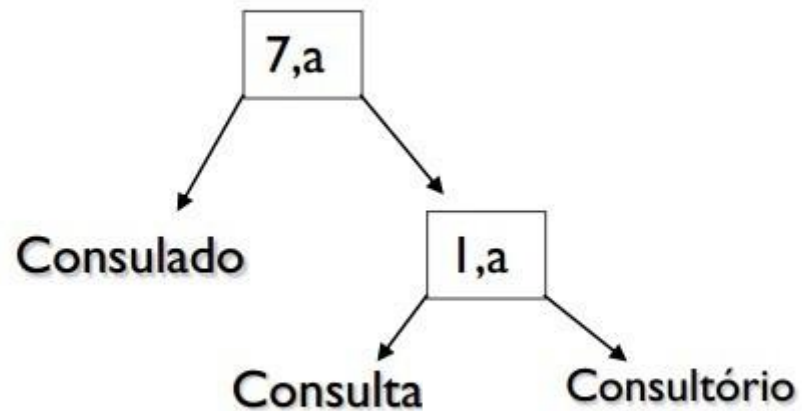
Acumulador Atualizado

PATRICIA: Exemplo de busca



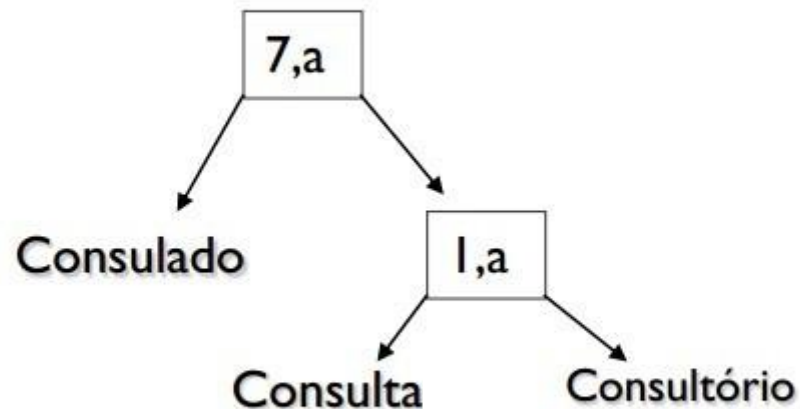
PATRICIA: Exemplo de busca

Busca por “**Consultório**”
■ ■



PATRICIA: Exemplo de busca

Busca por “Consultório”
■ ■

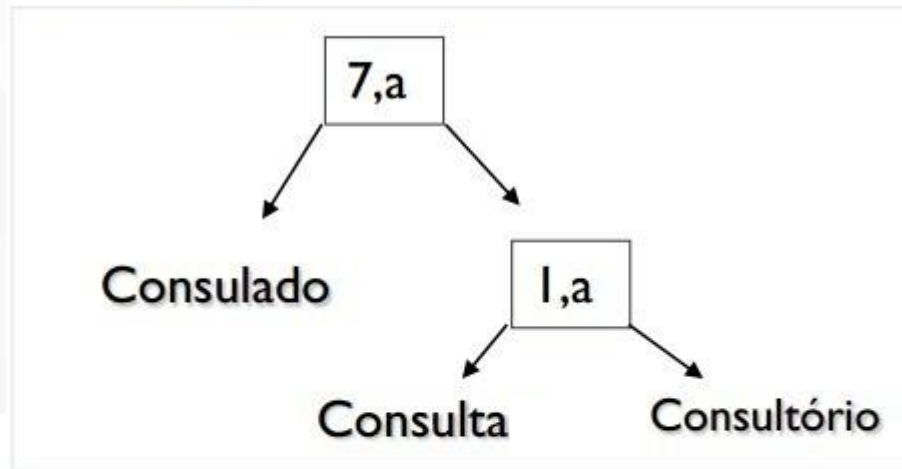


Etapas:

- 1) Primeira consulta:
 - Posição 7, letra 'a'.
 - Como 't' é maior que 'a', Desloca-se para direita
- 2) Segunda consulta:
 - Posição 8, letra 'a'.
 - Como 'o' é maior que 'a', Desloca-se para direita

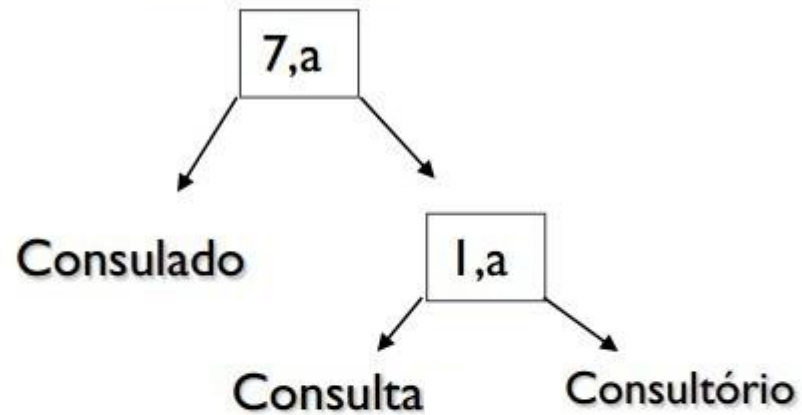
Se a palavra procurada é igual à palavra armazenada no nó, então acha-se a palavra.

PATRICIA: Exemplo de remoção



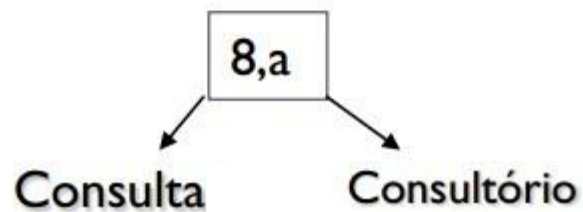
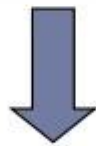
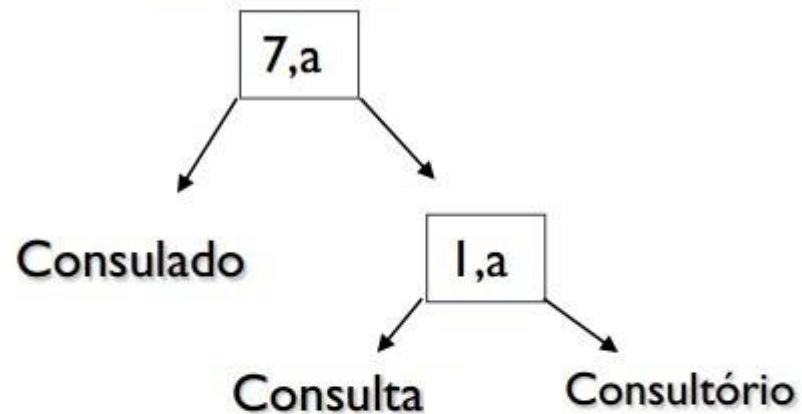
PATRICIA: Exemplo de remoção

Apagar “**Consulado**”



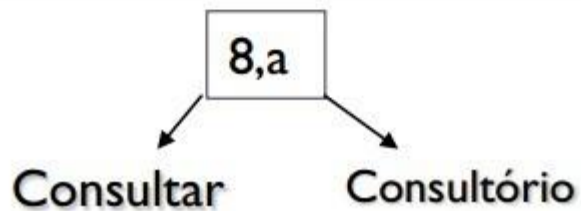
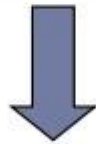
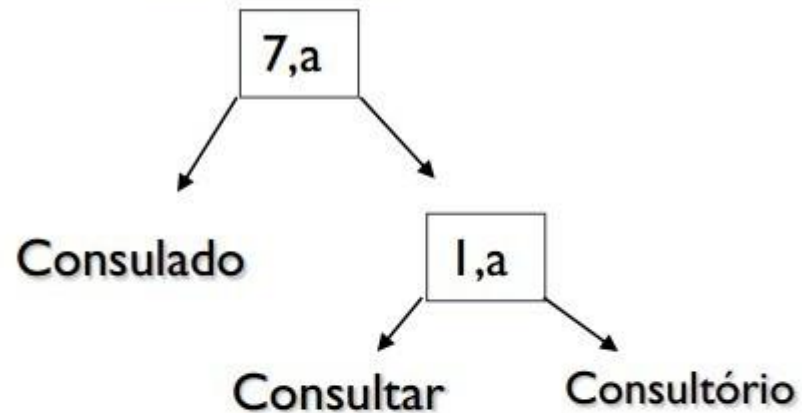
PATRICIA: Exemplo de remoção

Apagar “**Consulado**”



PATRICIA: Exemplo de remoção

Apagar “**Consulado**”



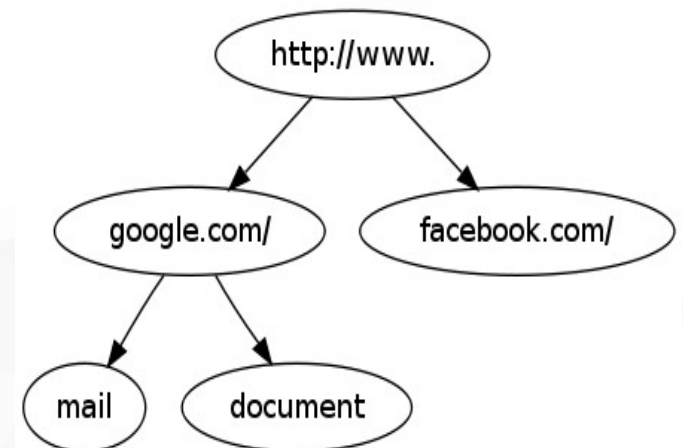
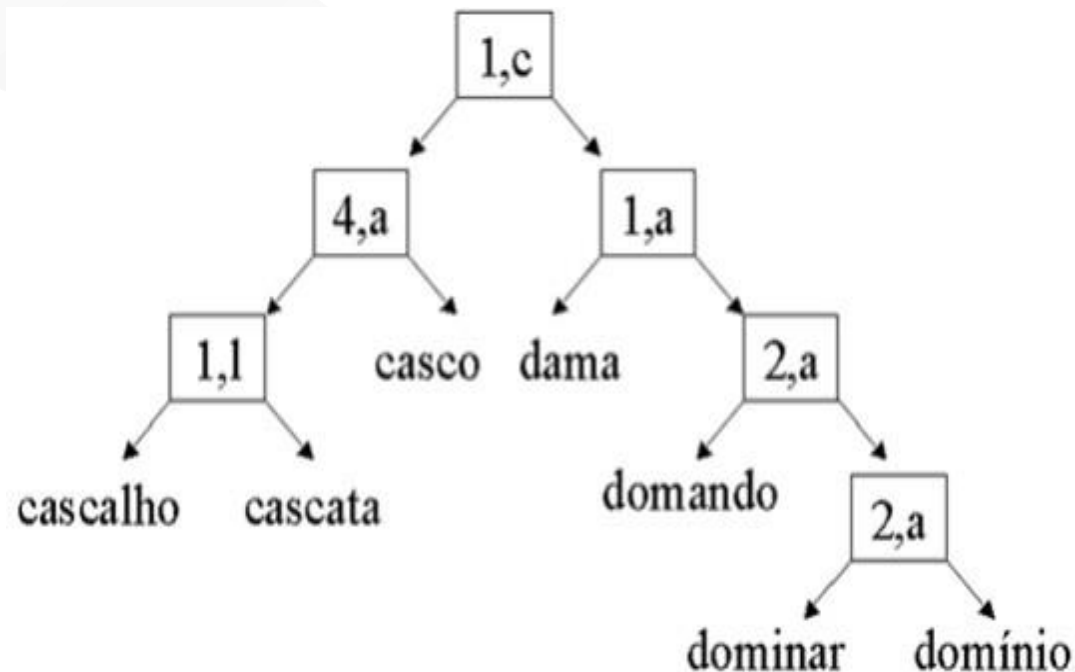
Etapas:

- 1) Busca-se e apaga-se a palavra “consulado”
- 2) Soma-se o valor do campo avançar do nó pai a todos os filhos.

- **Vantagem**

Permite armazenar um número de posições para qual é movido para frente antes de fazer a próxima comparação.

[elimina comparações desnecessárias → melhora o desempenho]



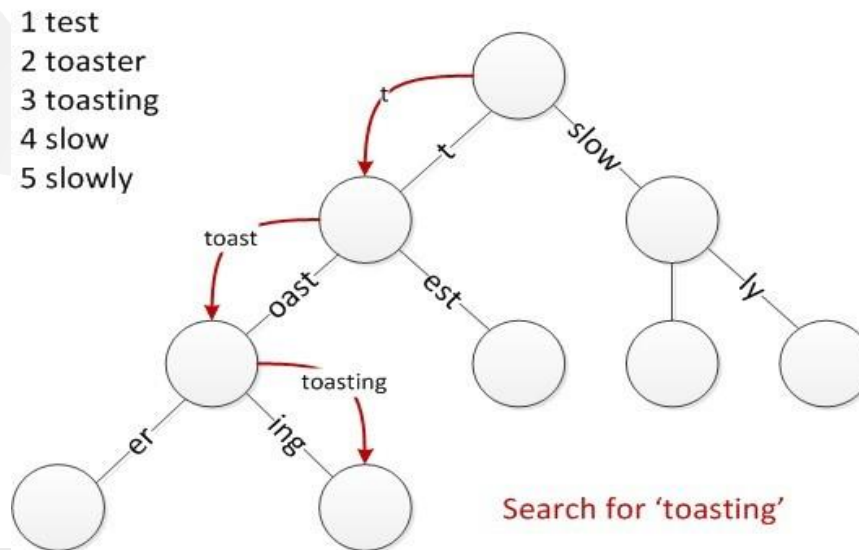
- **Desvantagem**

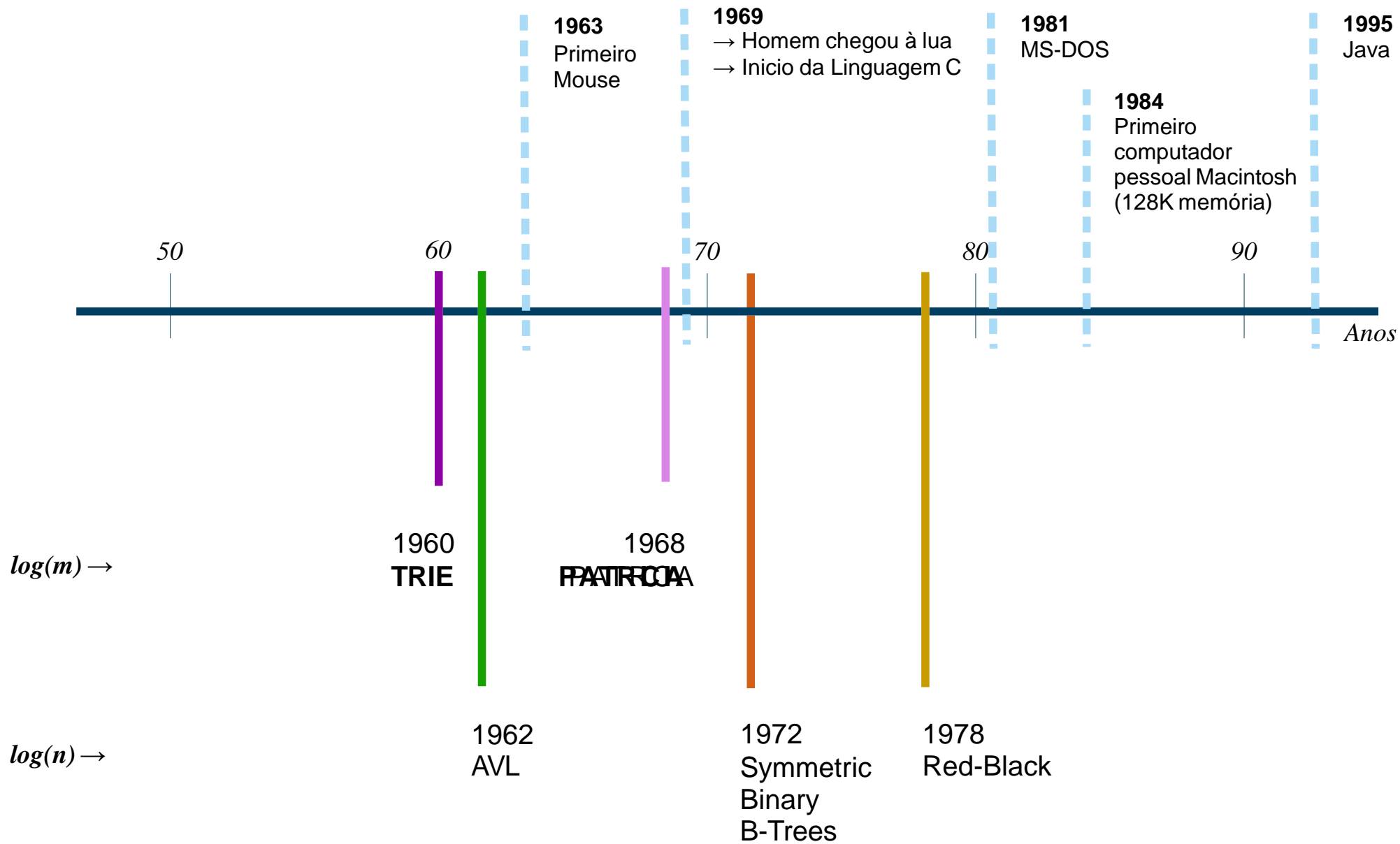
Considera apenas 2 subárvores.

Se mais do que duas chaves são distintas na mesma posição do caractere, é necessário adicionar nós extras ao índice para separá-lo.

[Se n chaves são distintas na mesma posição, então serão necessários $n-1$ nós para separá-los.

Se muitos casos destes acontecem, é preferível utilizar TRIE]



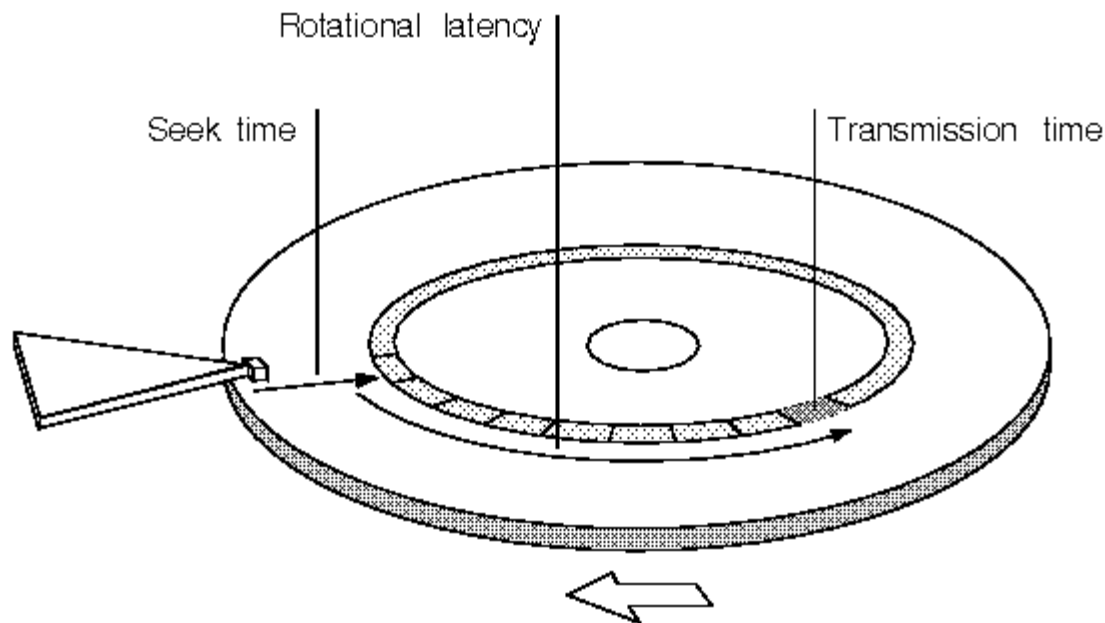


Custo de acesso a disco

Custo do acesso a disco

O custo do acesso a disco pode ser dividido em três operações distintas

- Tempo de busca (seek).
- Atraso rotacional no disco.
- Tempo de transferência dos dados.



Custo do acesso a disco

Tempo de busca (Seek time)

Tempo necessário para se mover o braço de acesso para o cilindro correto.

Atraso rotacional (Rotational delay)

É o tempo que o disco necessita para rotacionar de forma que o setor desejado fique sob a cabeça de leitura/escrita.

Tempo de transferência (Transfer time)

É dado pela seguinte fórmula:

Tempo de transferência = (Número de bytes transferidos / Número de bytes na trilha) x tempo de rotação.

Comparação memória RAM vs discos

❑ Discos são

lentos Um acesso na memória RAM (Random Access Memory) tipicamente pode ser feito em 30 a 60 nanosegundos (30×10^{-9} s).

- Obter a mesma informação no disco tipicamente leva de 7 a 10 milisegundos (7×10^{-3} s).
- Memória RAM é cerca de 1 milhão de vezes mais rápida que o disco.

❑ Mas provêm uma capacidade maior a um preço menor.

❑ Também são capazes de reter dados quando o computador é desligado.

Circuit Board/PCB

NAND Flash

SATA Controller

SATA Connector



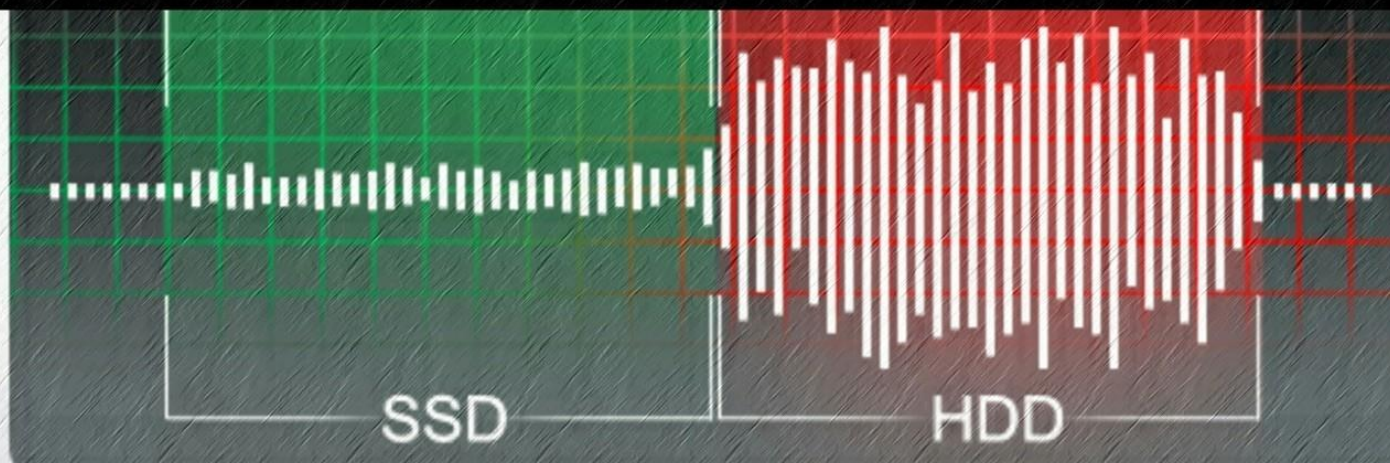
Platter

Actuator

Spindle

Actuator Arm

SATA Connector



SSD



HDD

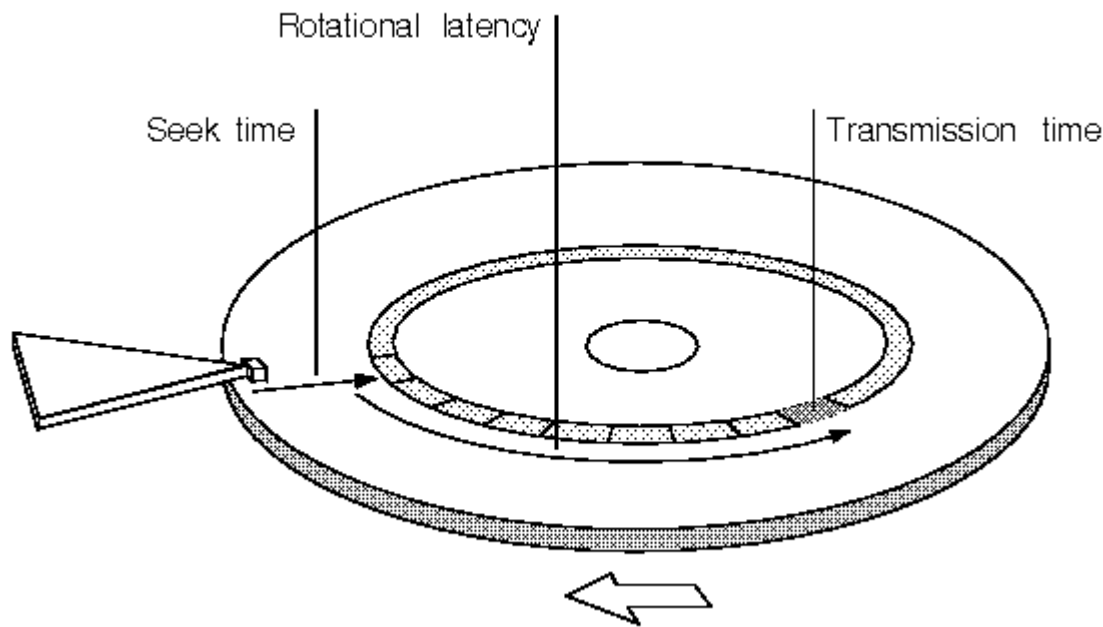




~ 8.0 ms	Access Times	~ .1 ms
4 to 6 Years	Reliability	14 to 36 Years
~ 500 ms	I/O Speed	~ 20 ms
~ 400 io/s	IOPS	~7000 io/s
18% over 4 Years	Failure Rates	3% over 10 Years
~ 15 Watts	Energy Usage	~ 5 Watts
~ 7% wait	CPU Usage	~ 1% wait

Estrutura de dados eficientes

- ❓ É importante considerar estrutura de dados que permitam **um número mínimo de leitura (acesso)** aos dados.



The background of the slide is white with a decorative pattern of light gray triangles of various sizes and orientations. These triangles are scattered across the page, with a higher concentration in the top-left and bottom-right corners, creating a modern, geometric aesthetic.

Árvores paginadas

Árvores binárias paginadas

Utilização do disco nas ABBs

Ler um nó desperdiça grande parte dos bytes lidos (sistemas atuais: mínimo de 512 bytes por página), já que só a chave e os ponteiros para os nós esquerdo e direito interessam

Page

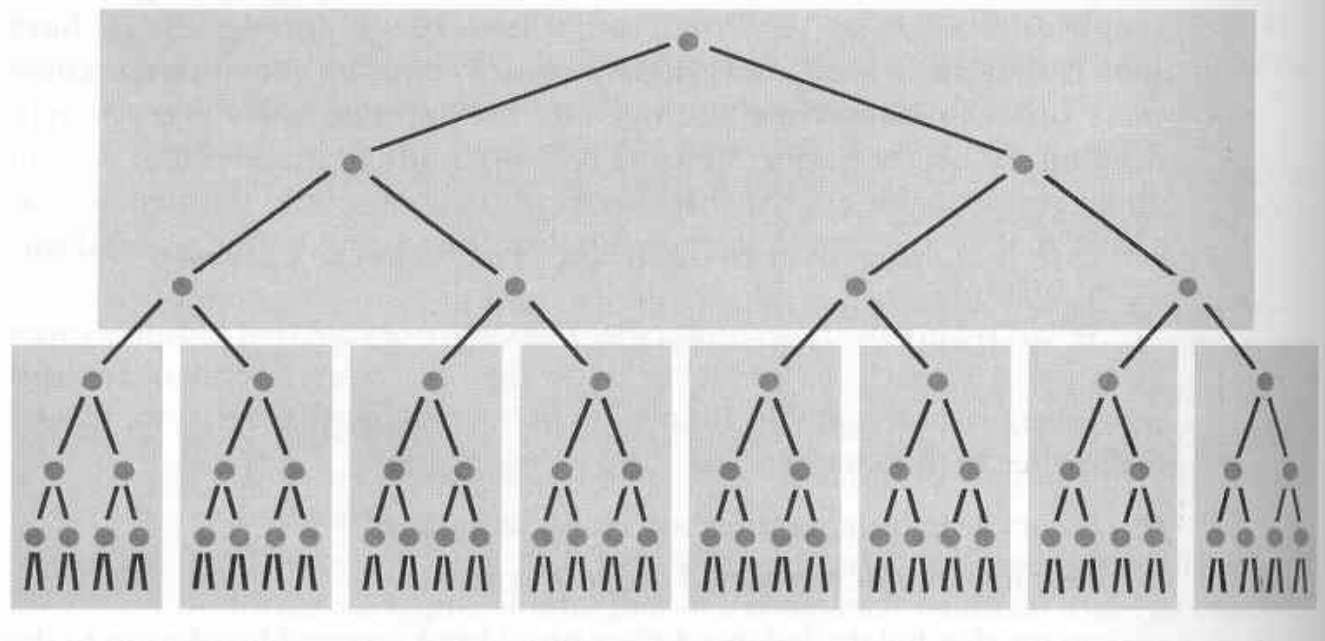
- A unit of disk I/O for handling seek and transfer of disk data
- Typically, 4k, 8k, 16k ...

Árvores binárias paginadas

- ❑ Vários nós na mesma página do disco.
- ❑ Divide a árvore em páginas
- ❑ **Armazena cada página em blocos contíguos no disco.** Assim, **é possível reduzir o número de buscas no disco.**

Árvores binárias paginadas

Exemplo: árvore de 63 nós (página de 7 nós).



$$\begin{aligned}(2^3-1) &= 7 \\ (2^6-1) &= 63 \\ (2^9-1) &= 511 \\ (2^{12}-1) &= 4095\end{aligned}$$

Com um nível adicional (3 seeks) é possível acessar 511 nós. Com mais outro nível (4 seeks), é possível acessar 4.095 nós.

Uma busca binária em 4.095 elementos necessita de **12 acessos**.

Árvores binárias paginadas

Número de buscas, no pior caso, para a versão paginada:

$$\log_{k+1}(N+1)$$

onde k é o número de chaves guardadas em uma página.

Árvores binárias paginadas

Número de buscas, no pior caso, para a versão paginada:

$$\log_{k+1}(N+1)$$

onde k é o número de chaves guardadas em uma página.

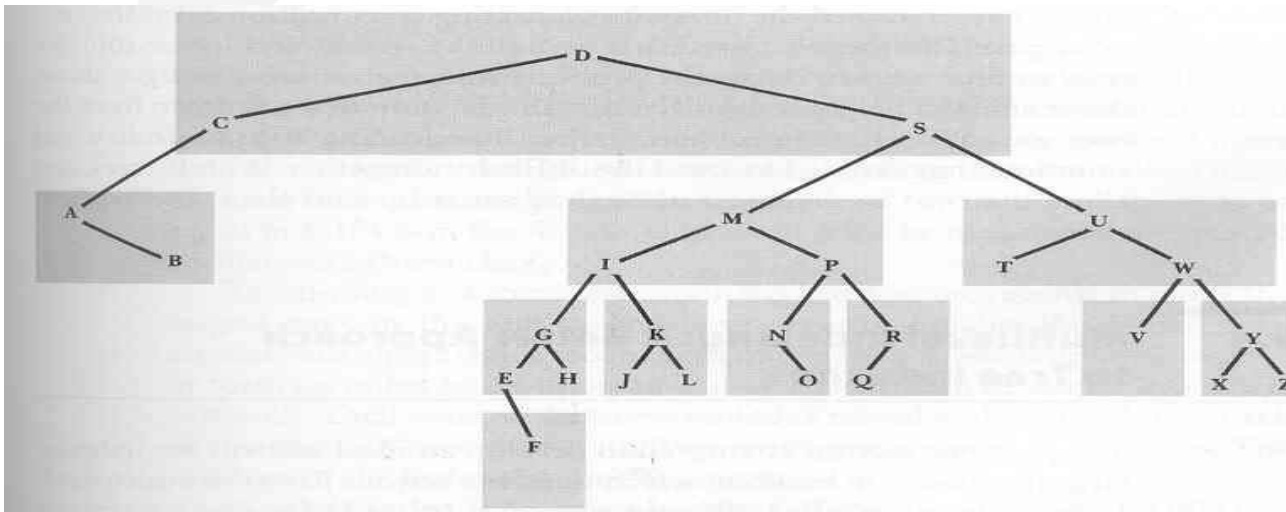
O uso do disco pelas árvores paginadas ainda é ineficiente.

- Não há razão para se usar uma árvore dentro de uma página!
 - ❑ Grande parte do espaço é desperdiçada em ponteiros.
 - Uma chave, 2 ponteiros
 - ❑ Possível busca binária em vetor, em memória: sem acessos ao disco.
 - Lembre-se: o caro é acesso ao disco

Árvores binárias paginadas

Um outro problema é como construir a árvore.

- Se temos as chaves de antemão, basta ordenar as chaves e construir a árvore com a lista ordenada (garante árvore balanceada).
- Se as chaves são recebidas de forma aleatória e inseridas ao chegar, a árvore é desbalanceada



sequência: C S D T A M P I B W N G U R K E H O L J Y Q Z F X V

Árvores binárias paginadas

Não se pode fazer uma rotação de uma página inteira como se faz uma árvore não-paginada.

- ❓ É necessário quebrar as páginas. Mas pode acontecer que o rearranjo se dissemine por grande parte da árvore. (***não local***)

Árvores binárias paginadas

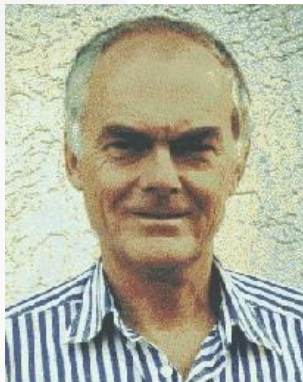
- Questões não-resolvidas - Como fazemos para:
 - ❓ Garantir que as chaves na página raiz sejam boas separadoras, dividindo o conjunto das outras chaves de maneira equilibrada?
 - Facilita o balanceamento global
 - ❓ Evitar que chaves muito distantes compartilhem a mesma página?
 - Mais chance de minimizar buscas (conceito de localidade)
 - ❓ Garantir que cada página contenha um número mínimo de chaves?
 - Desperdício fazer busca no disco para carregar poucas chaves

Árvores binárias paginadas

- **Questões não-resolvidas - Como fazemos para:**
 - ❑ Garantir que as chaves na página raiz sejam boas separadoras, dividindo o conjunto das outras chaves de maneira equilibrada?
 - Facilita o balanceamento global
 - ❑ Evitar que chaves muito distantes compartilhem a mesma página?
 - Mais chance de minimizar buscas (conceito de localidade)
 - ❑ Garantir que cada página contenha um número mínimo de chaves?
 - Desperdício fazer busca no disco para carregar poucas chaves
- **Solução:** construção da árvore de baixo para cima. A raiz emerge do conjunto de chaves.

Árvores B

Artigo: R. Bayer, E. McCreight “*Organization and Maintenance of Large Ordered Indexes*”. Acta-Informatica, 1:173-189, 1972.



Organization and maintenance of large ordered indices

Full Text:  Pdf

Authors: [R. Bayer](#) Mathematical and Information Sciences Laboratory
[E. McCreight](#) Mathematical and Information Sciences Laboratory

Published in:

· Proceeding
SIGFIDET '70 Proceedings of the 1970 ACM SIGFIDET (now SIGMOD)
Workshop on Data Description, Access and Control
Pages 107-141
[ACM](#) New York, NY, USA ©1970
[table of contents](#) doi > [10.1145/1734663.1734671](https://doi.org/10.1145/1734663.1734671)




 1970 Article



[Bibliometrics](#)

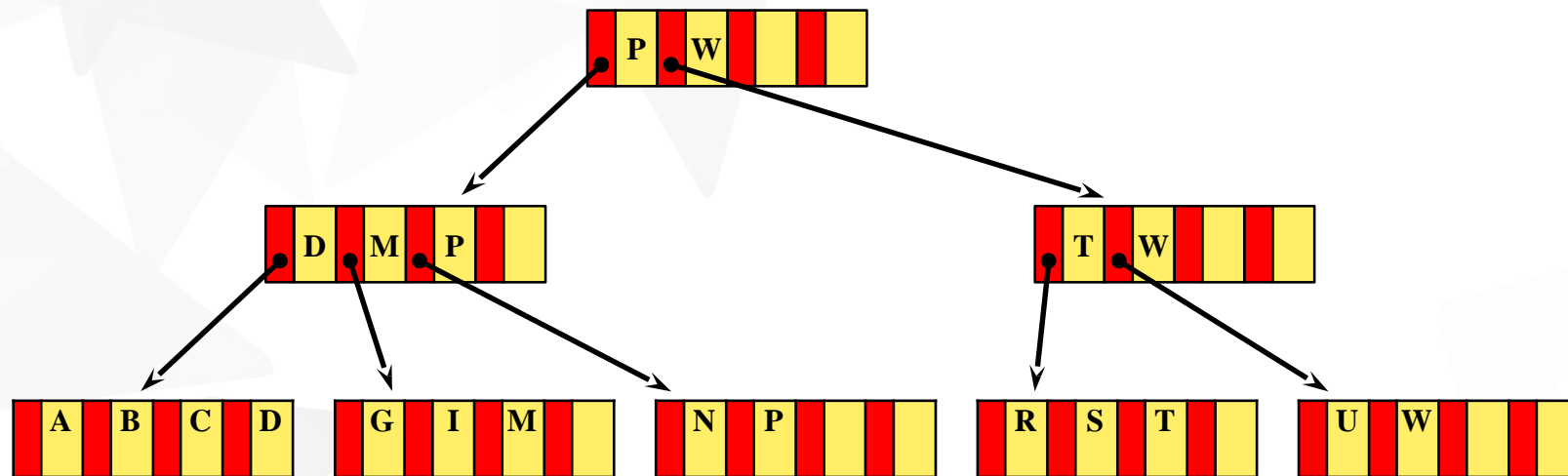
· Downloads (6 Weeks): 9
· Downloads (12 Months): 47
· Downloads (cumulative): 175
· Citation Count: 19

Rudolf Bayer

Born May 7, 1939 (age 75)
Nationality [German](#)
Institutions [Technical University Munich](#)
Alma mater [University of Illinois at Urbana–Champaign](#)
Thesis [Automorphism Groups and Quotients of Strongly Connected Automata and Monadic Algebras](#)  (1966)
Doctoral advisor [Franz Edward Hohn](#)^[1]
Known for [B-tree](#)
[UB-tree](#)
[red-black tree](#)
Notable awards [Cross of Merit, First class \(1999\)](#),
[SIGMOD Edgar F. Codd Innovations Award \(2001\)](#)

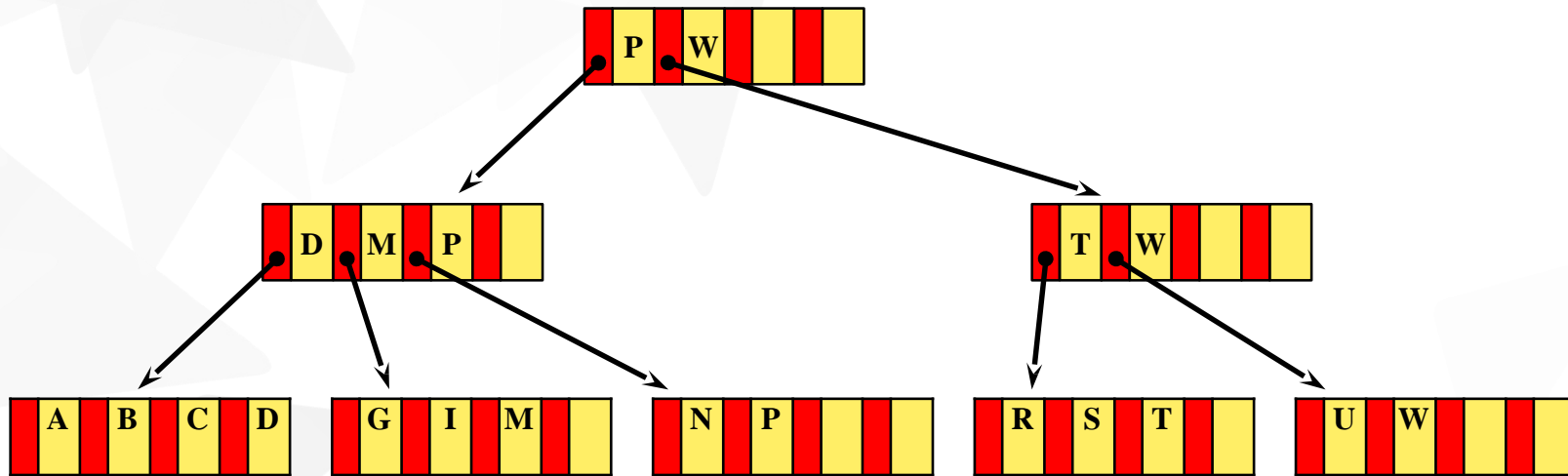
O nome árvores B não foi explicado:
Balanced, Broad, Boeing, Bayer.

Exemplo



- Uma árvore B com **tamanho de página = 64** ao indexar um arquivo com 1 milhão de registros possibilita achar a chave de qualquer registro em até **3 acessos** ao disco ($\log_{64} 1000000$).
- Busca binária = **20 acessos** ($\log_2 1000000$)

Exemplo



Ordem: Número máximo de chaves armazenadas em um nó.

Cada nó deve ter um número mínimo de chaves (exceção a raiz):
~metade da ordem.

Divide o nó cheio em 2 com metade das chaves.

Exemplo

Uso da seguinte sequência:

- **C S D T A M P I B W N G U R K E H O L J Y Q Z F X V.**
- Árvore B de ordem 4 ($m=4$).
- **As 26 chaves são inseridas em uma árvore de altura 3.**
 - ❑ O número de nós afetados por qualquer inserção nunca é maior do que dois nós por nível (um mudado e outro criado por uma divisão).
 - ❑ Com uma ordem maior a árvore é mais “baixa”

○ Input Sequence:

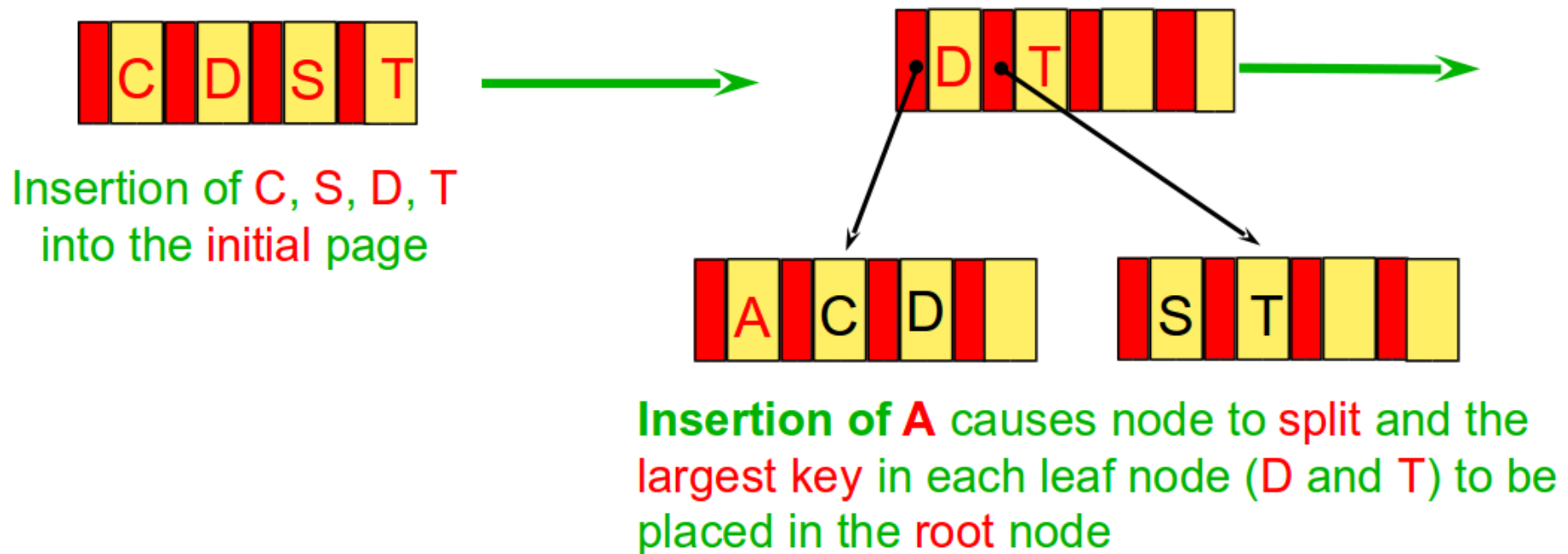
C S D T A M P I B W N G U R K E H O L J Y Q Z F X V



Insertion of C, S, D, T
into the initial page

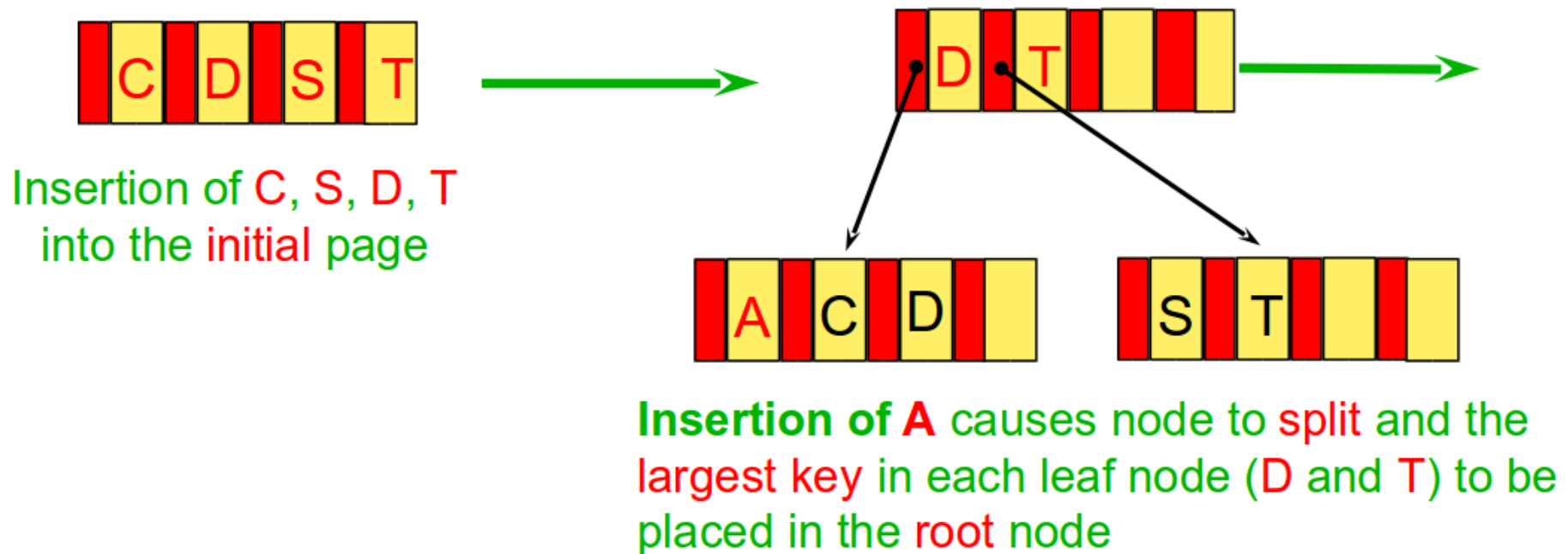
○ Input Sequence:

C S D T A M P I B W N G U R K E H O L J Y Q Z F X V



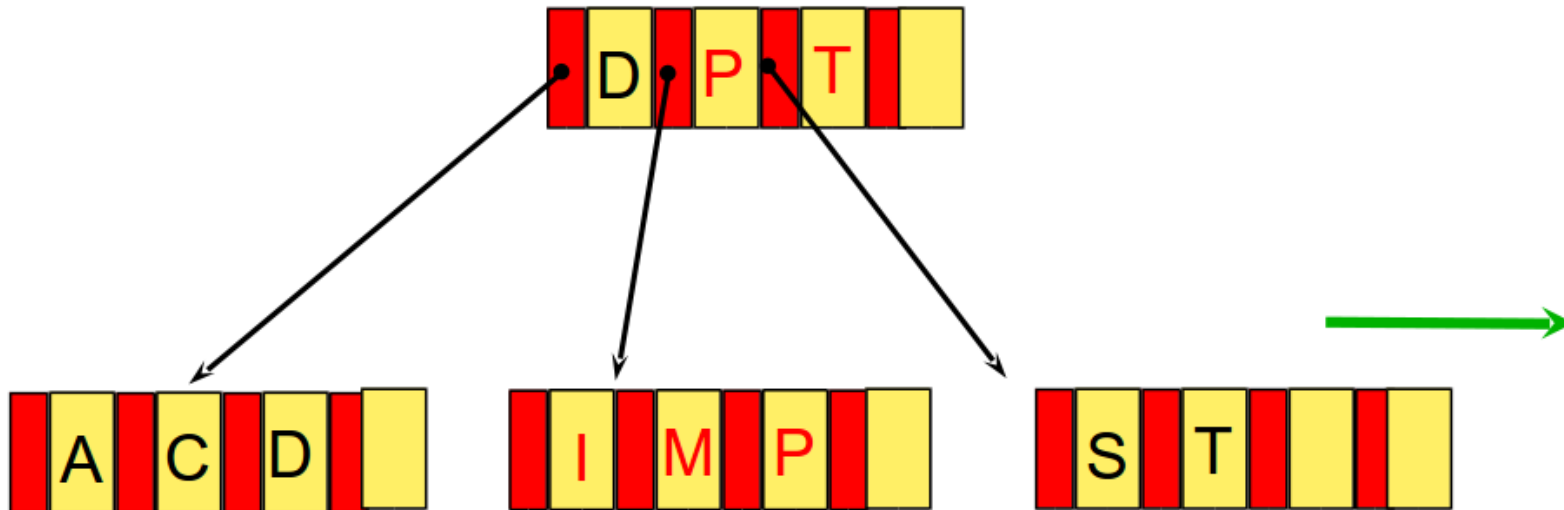
○ Input Sequence:

C S D T A M P I B W N G U R K E H O L J Y Q Z F X V



○ Input Sequence:

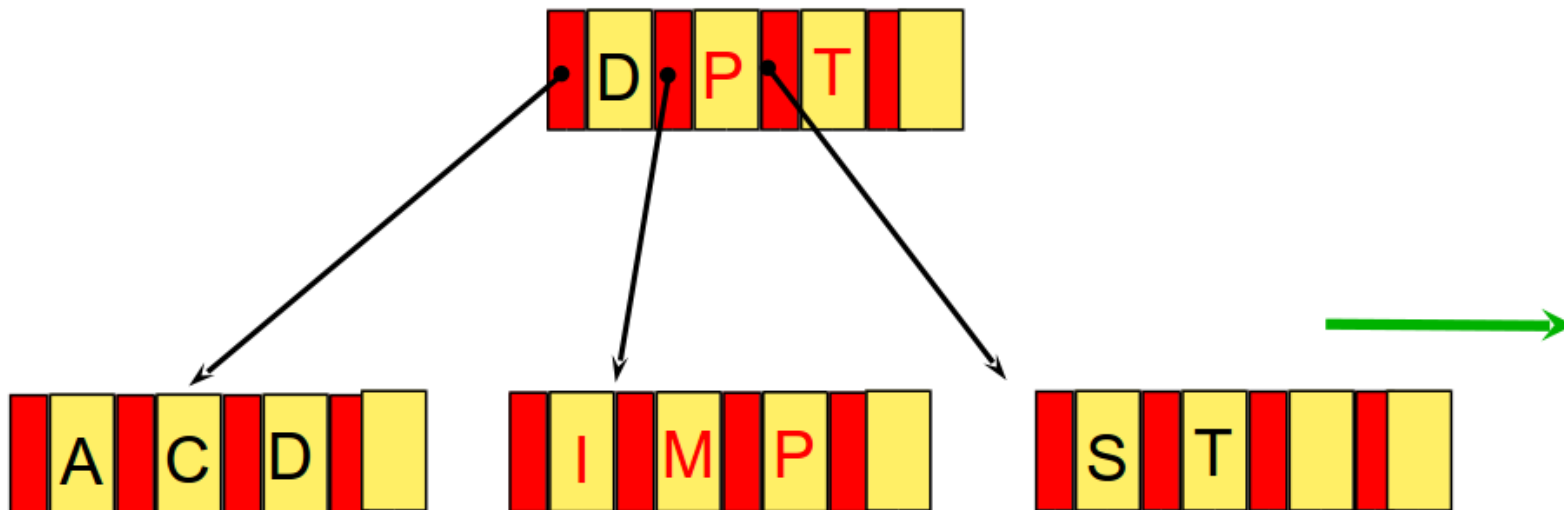
C S D T A **M** **P** **I** B W N G U R K E H O L J Y Q Z F X V



M and **P** are inserted into the rightmost leaf node,
then insertion of **I** causes it to split

○ Input Sequence:

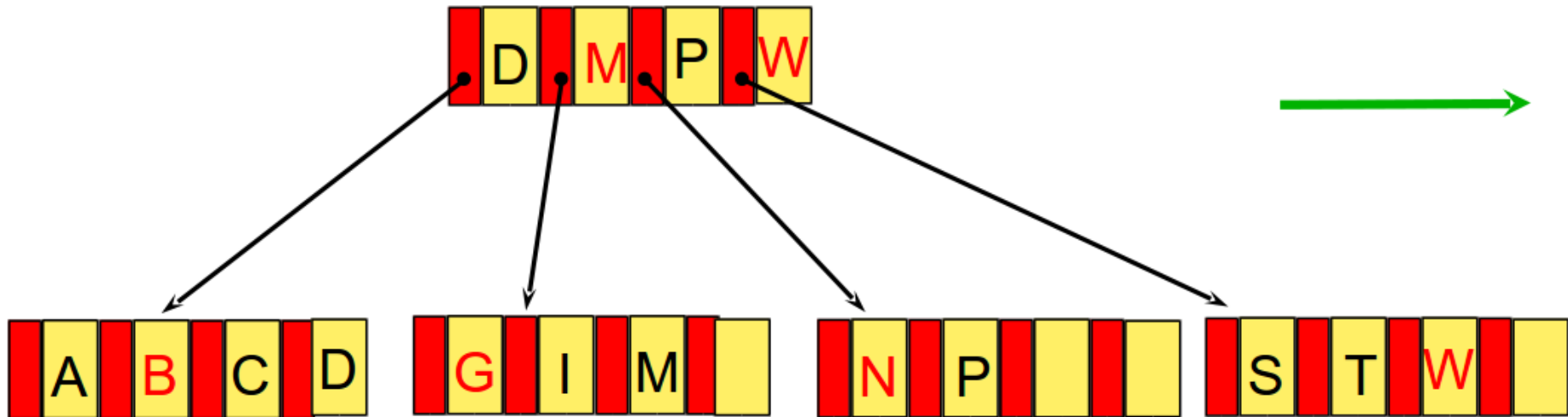
C S D T A **M P** B W N G U R K E H O L J Y Q Z F X V



M and **P** are inserted into the rightmost leaf node,
then insertion of **I** causes it to split

○ Input Sequence:

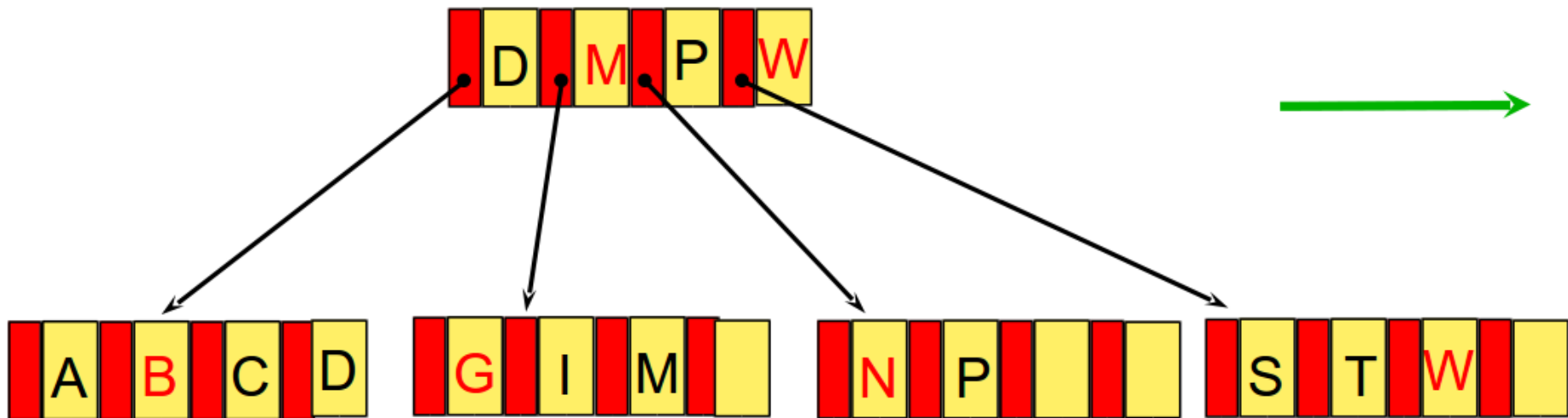
*C S D T A M P I **B W N G** U R K E H O L J Y Q Z F X V*



Insertions of B, W, N, and G into leaf nodes causes
another split and the root is now full

○ Input Sequence:

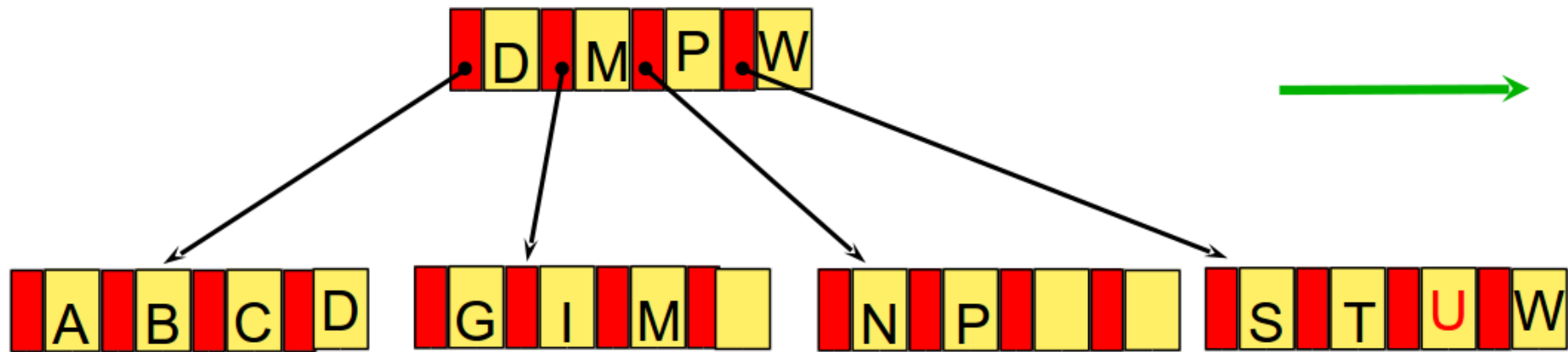
C S D T A M P I **B W N G** U R K E H O L J Y Q Z F X V



Insertions of B, W, N, and G into leaf nodes causes
another split and the root is now full

○ Input Sequence:

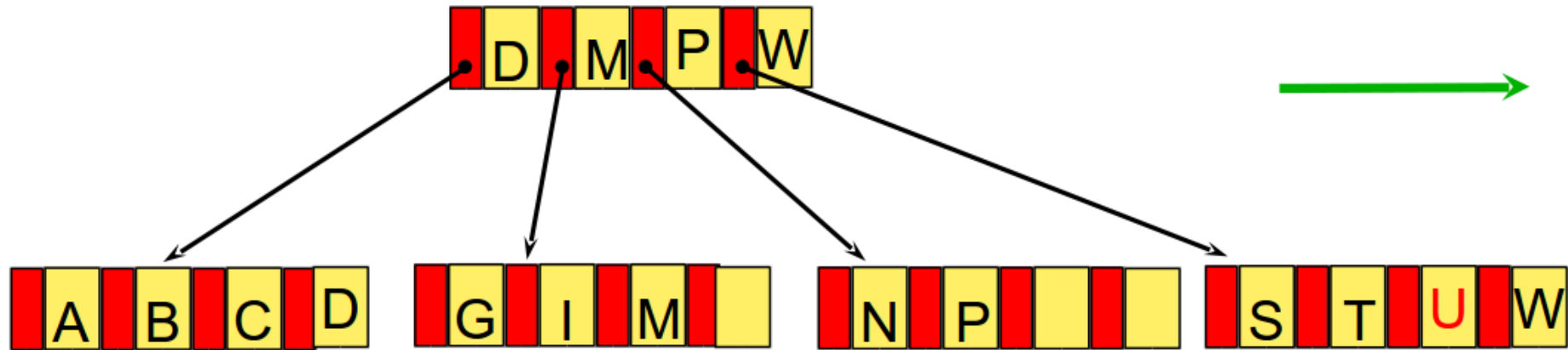
C S D T A M P I B W N G U R K E H O L J Y Q Z F X V



Insertion of **U** proceeds without incident, but **R** would have to be inserted into the **rightmost leaf**, which is **full**

○ Input Sequence:

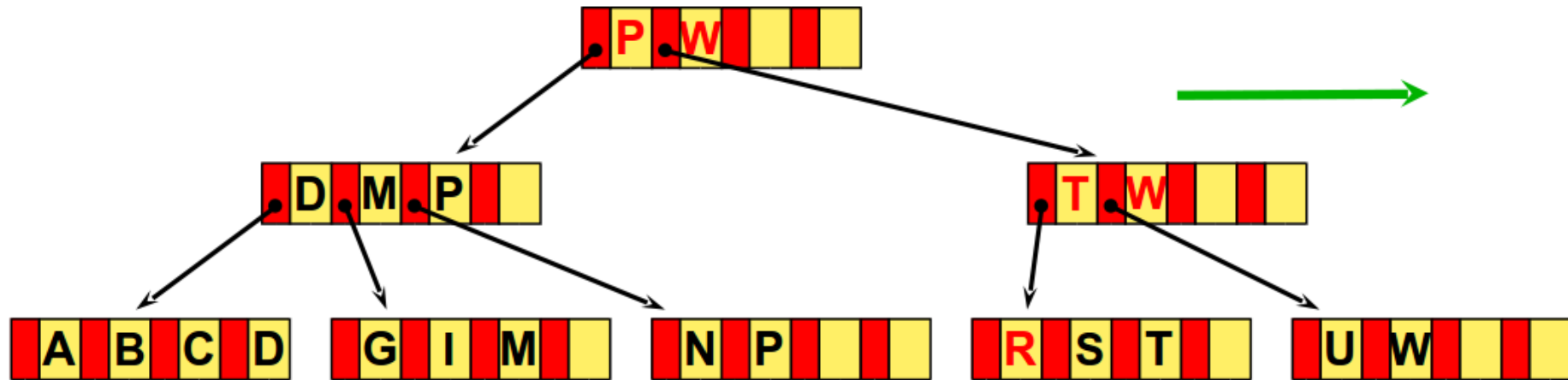
C S D T A M P I B W N G **U** **R** K E H O L J Y Q Z F X V



Insertion of **U** proceeds without incident, but **R** would have to be inserted into the **rightmost leaf**, which is **full**

○ Input Sequence:

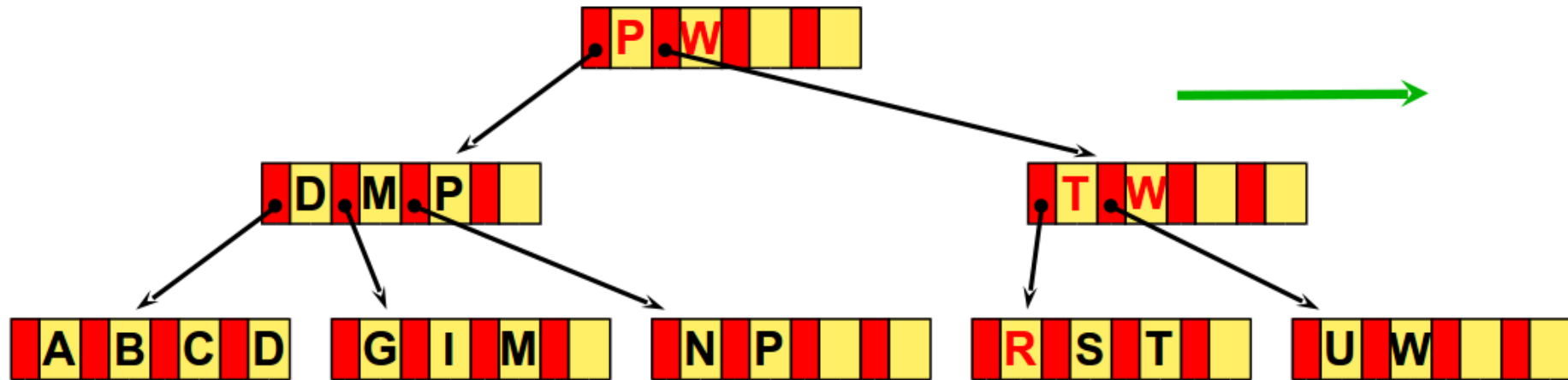
C S D T A M P I B W N G U **R** K E H O L J Y Q Z F X V



Insertion of **R** causes the **rightmost leaf node to split**, insertion into the root causes the **root to split** and the tree **grows to level three**

○ Input Sequence:

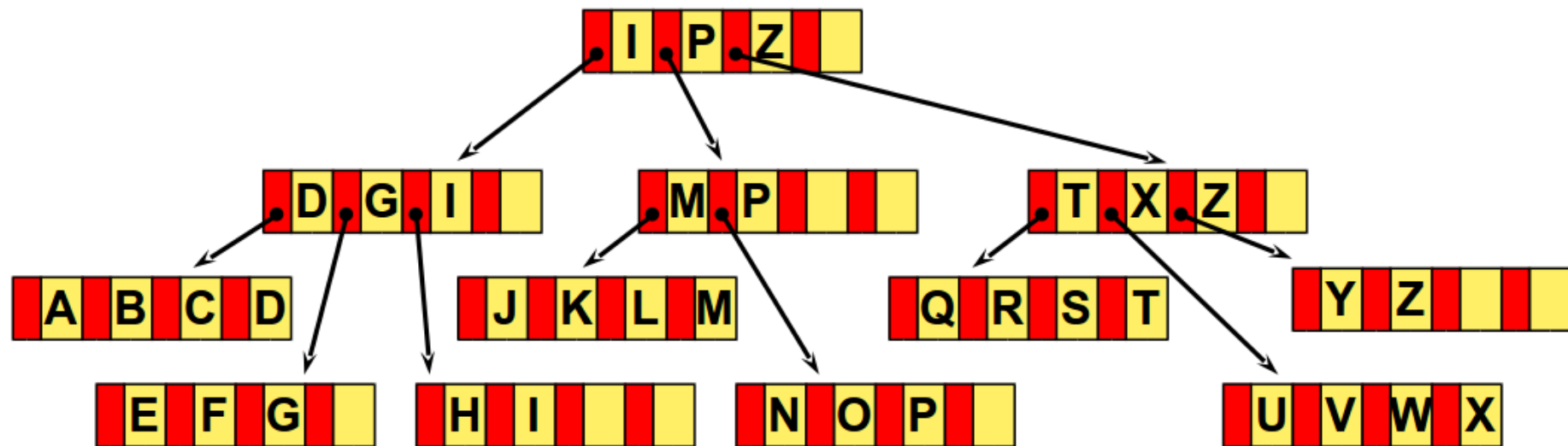
C S D T A M P I B W N G U **R** K E H O L J Y Q Z F X V



Insertion of **R** causes the **rightmost leaf node to split**, insertion into the root causes the **root to split** and the tree **grows to level three**

○ Input Sequence:

C S D T A M P I B W N G U R K E H O L J Y Q Z F X V





Insertions of **F**, **X**, and **V** finish the insertion of the alphabet

Utilização de Árvores B



Sistemas de arquivos:

- Índice para relacionar bloco lógico com setor físico
- FAT original (Windows) era uma lista
 -  Onde no disco estava o byte 542345 do arquivo?
-  Tinha que percorrer a lista
- Apple HFS, NTFS, ext4, usam Árvores B (B-trees)
Btrfs: apelidado de “B-tree file system”



Banco de dados:

- Um índice com operações $O(\log n)$
- Adequado para utilizar memória secundária

Historical Note

- ◆ B-tree : Bayer & McCreight
- ◆ B⁺-tree: Comer
- ◆ B^{*}-tree : Knuth, B-trees with 67% minimum occupancy
- ◆ B[÷]-trees : B⁺-trees with 67% minimum occupancy