Tries

Busca Digital/Árvore Digital

Até o momento fizemos muitos processos de busca: em árvores, em tabelas...

Mas sempre relacionando a busca de chave x, por meio de comparações, em uma estrutura y. Não é mesmo?

> Nos assuntos anteriores as estruturas de busca "possuíam como restrição que as chaves fossem todas de mesmo tamanho. Se esse fato não ocorresse, o tamanho da maior chave seria tomado como padrão, completando-se as demais com caracteres extras até que o tamanho padrão fosse atingido"

Mas nada impede que as chaves possam possuir tamanhos diferentes, que apenas em parte sejam iguais.

Como um texto armazenado em que podemos realizar buscas em parte dele

Então, dessa forma, podemos imaginar uma estrutura que não trate mais as chaves como elementos indivisíveis e sim como um conjunto de caracteres e dígitos

A Busca digital possui essa característica, ao invés de compararmos as chaves como unidades indivisíveis, analisa-se as chaves elemento a elemento ou dígito a dígito.

Então, as Tries buscam chaves em um número de passos igual ao tamanho da chave

$S = \{s1, \ldots, sn\}$ um conjunto de n chaves em que cada si é formada por uma sequência de elementos dj denominados *dígitos*

Em S possuem que tenhamos m dígitos distintos, conforme alfabeto;
Os dígitos são organizados, conforme o alfabeto, $d_1 < \ldots < d_m$;
Como algumas palavras podem ter seu início semelhante, os p primeiros dígitos de uma chave forma o que denominamos de prefixo.

Uma *árvore digital* para S é uma árvore m-ária T, não vazia, tal que:

(i) Se um nó v é o j-ésimo filho de seu pai, então v corresponde ao dígito d_j do alfabeto S, $1 \leq j \leq m$.

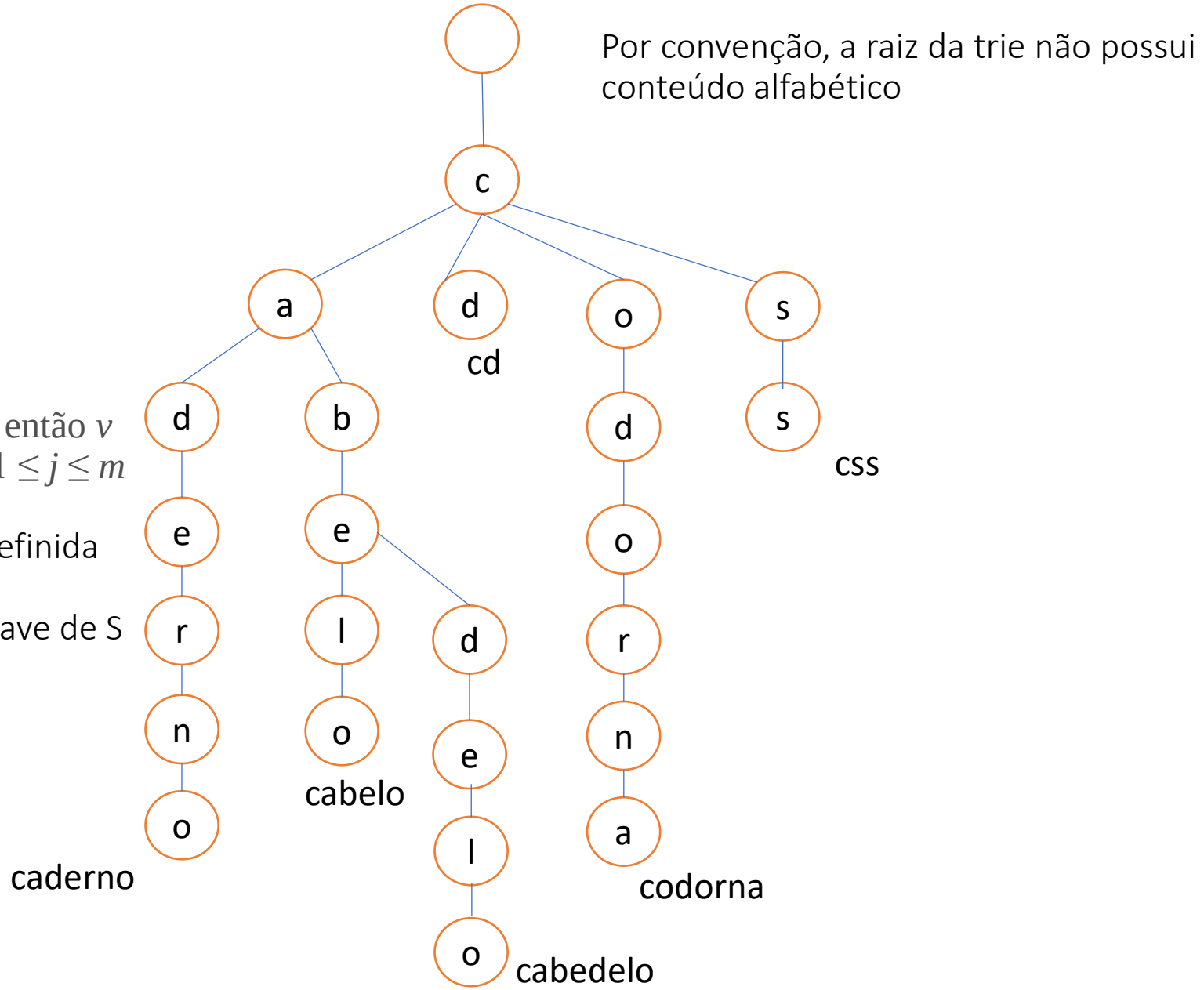
(ii) Para cada nó v, a sequência de dígitos definida pelo caminho desde a raiz de T até v corresponde a um prefixo de alguma chave de S.

Como são?

Validando as propriedades:

Se um nó *v* é o *j*-ésimo filho de seu pai, então *v* corresponde ao dígito *dj* do alfabeto *S*, $1 \leq j \leq m$

Para cada nó v, a sequência de dígitos definida pelo caminho desde a raiz de T até v corresponde a um prefixo de alguma chave de S

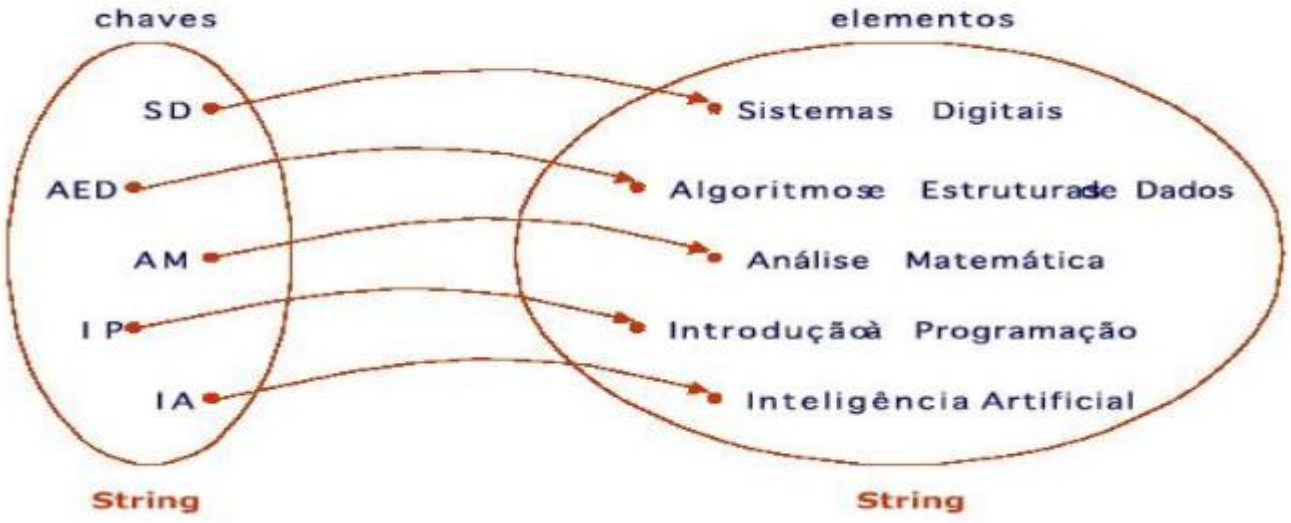


Como visto, cada trie possui:
Um conjunto de palavras com tamanhos variados;

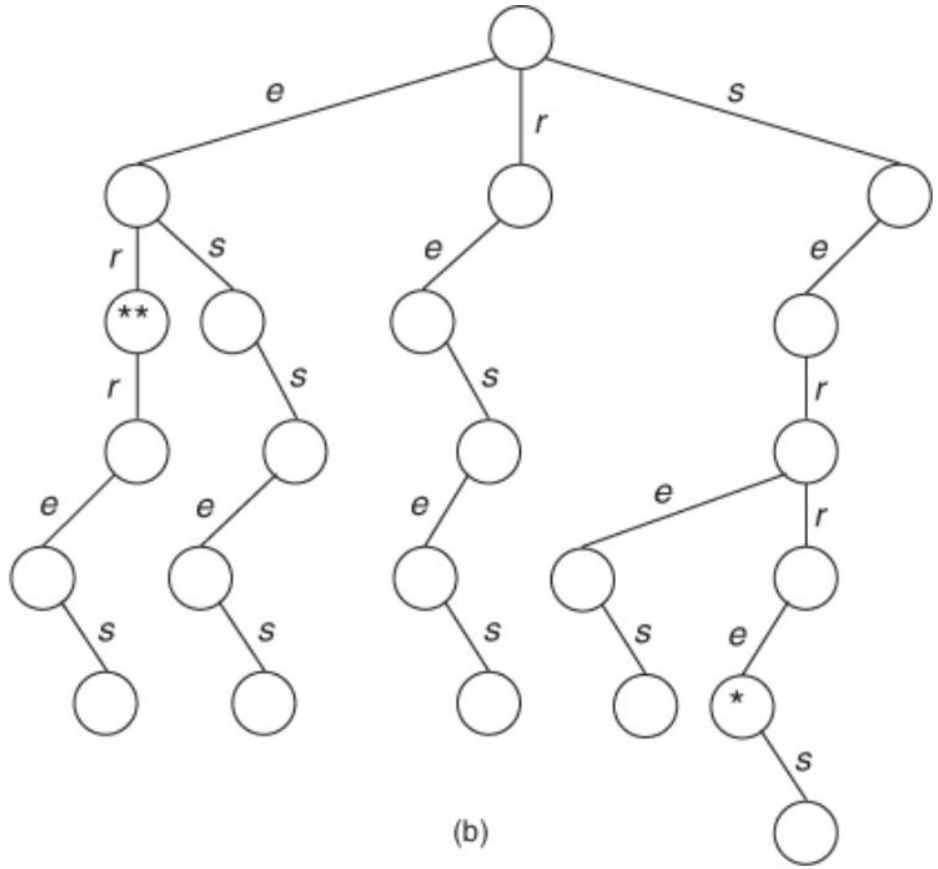
As palavras são formadas por símbolos, pertencentes a um conjunto finito = alfabetos, por exemplo: {a,b,c}, {0,1}, {a-z};

O processo de reconhecimento de uma chave na estrutura é por meio do percurso da raiz até uma folha terminal.

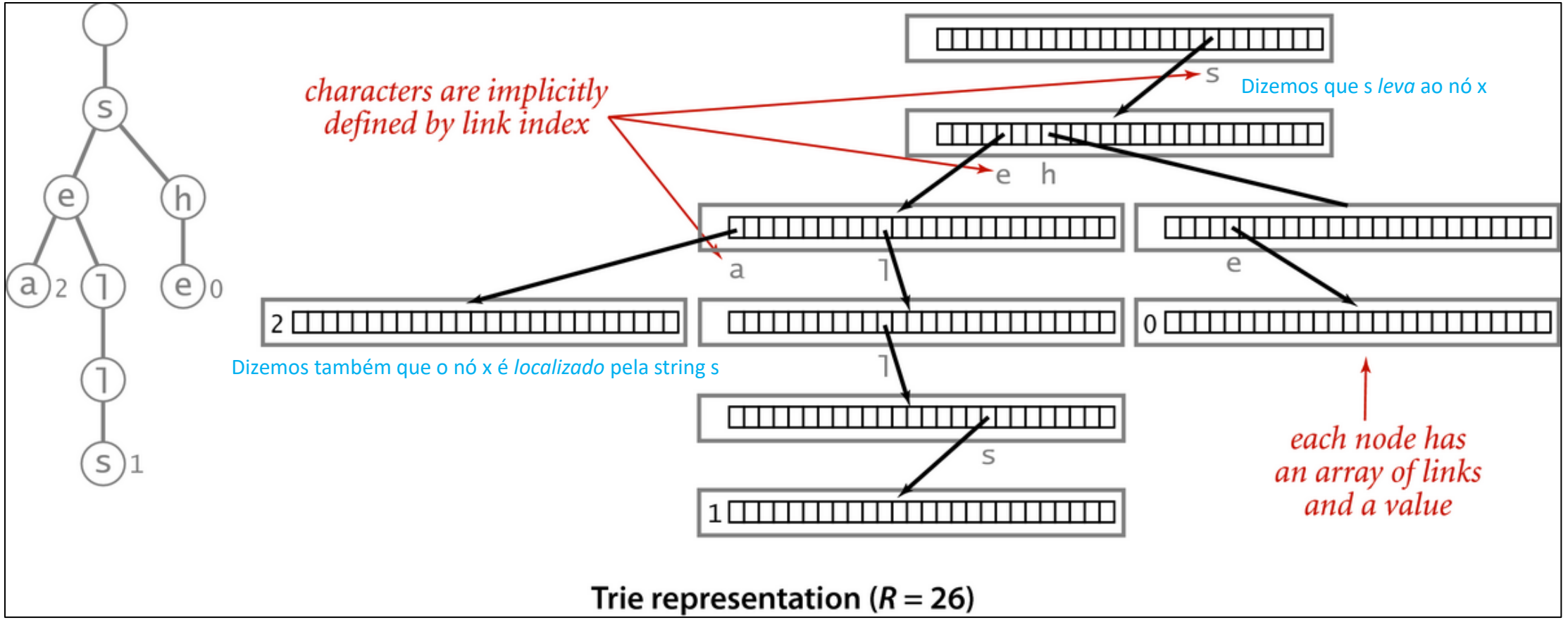
Tais estruturas podem ser associadas a registros:



Fonte: https://www.ufjf.br/jairo_souza/files/2009/12/6-Strings-Pesquisa-Digital.pdf



E sua representação pode corresponder a um conjunto de m caminhos, por vetores de m posições



Fonte: https://www.ime.usp.br/~pf/estruturas-de-dados/aulas/tries.html

Dessa forma, cada unidade de espaço não armazena conteúdos (partes) de chaves e sim, referências para posições do alfabeto

Quando uma string leva a um nó x que é chave, haverá algum valor associado a ela, um marcador para término, por exemplo: true, 1, nome da string, algo diferente de nulo

Analisando o processo de busca

Corresponde ao caminhar na estrutura comparando elemento a elemento que compõe a chave

Caso o último elemento procurado for correspondente ao elemento comparado, então verifica-se seu valor

Pensando em analisar a complexidade dessas operações, torna-se fácil. O número de iterações realizadas corresponde, ao máximo, o número de elementos que compõe a chave, e que podemos denominar de k

Outras Características

Outro ponto importante a ser observado é o tamanho do alfabeto, pois estruturas com o alfabetos grandes promovem um gasto de alocação de espaços. Há outras versões mas aperfeiçoadas, como TST que ajuste este ponto.

Mas em linhas gerais esta é uma característica negativa na estrutura

Por ter essa característica, na maioria dos casos m tem um valor pequeno, e o processo de busca, para estes casos, pode ser O(k)

Por ter essa característica, na maioria dos casos m tem um valor pequeno, e o processo de busca, para estes casos, pode ser O(k)

A aparência, diferentemente de árvores binárias independe da ordem de inserção/remoção de chaves, assim como o consumo de tempo

Vamos então conhece-la?

Próxima semana
Trabalharemos na Disciplina:

União e Busca = Union Finds

Fontes e Referências:

SZWARCFITER, Jayme Luiz; MARKENZON, Lilian. **Estruturas de Dados e seus Algoritmos**. Livros Tecnicos e Cientificos, 1994.