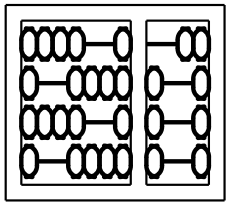


Universidade Estadual de Campinas - UNICAMP
Instituto de Computação - IC

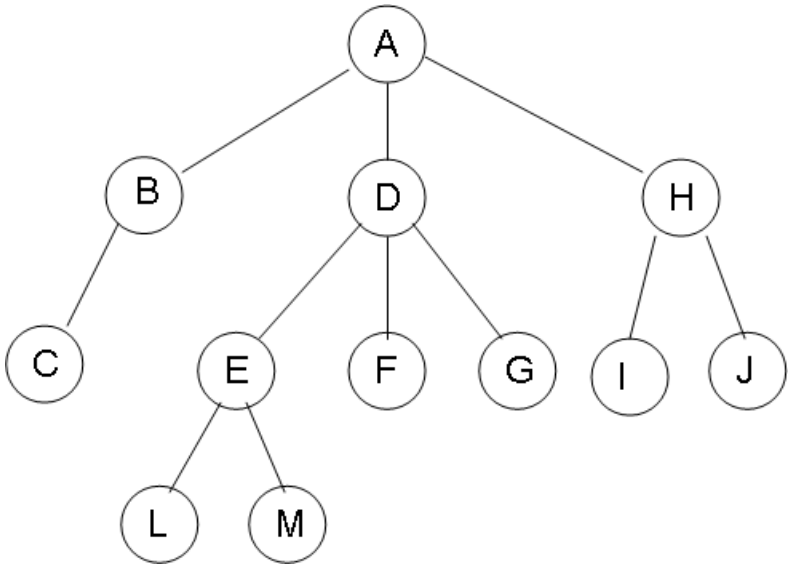
Árvores Gerais

Tópicos

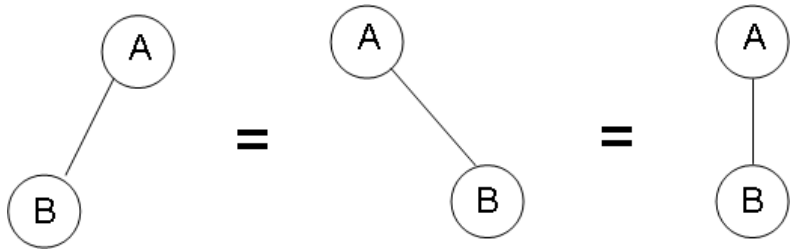
1. Árvores Gerais
2. Florestas
3. Árvores Digitais
4. Árvores Binárias Digitais

Árvores Gerais

• Definição: Uma *árvore geral* T é um conjunto finito e não vazio de objetos (nós ou vértices), tal que T consiste de um nó distinto, denominado raiz de T, e de $m \geq 0$ árvores disjuntas, $T_1, T_2, ..., T_m$, chamadas subárvores de T.



• Observe que não existe o conceito de árvore vazia, como no caso binário e, portanto:

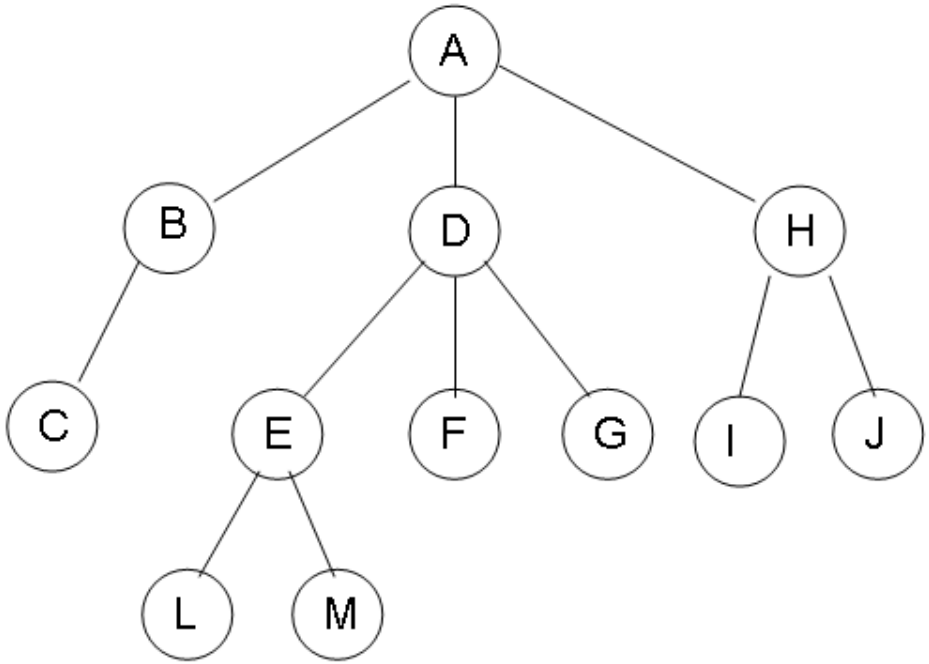


Florestas

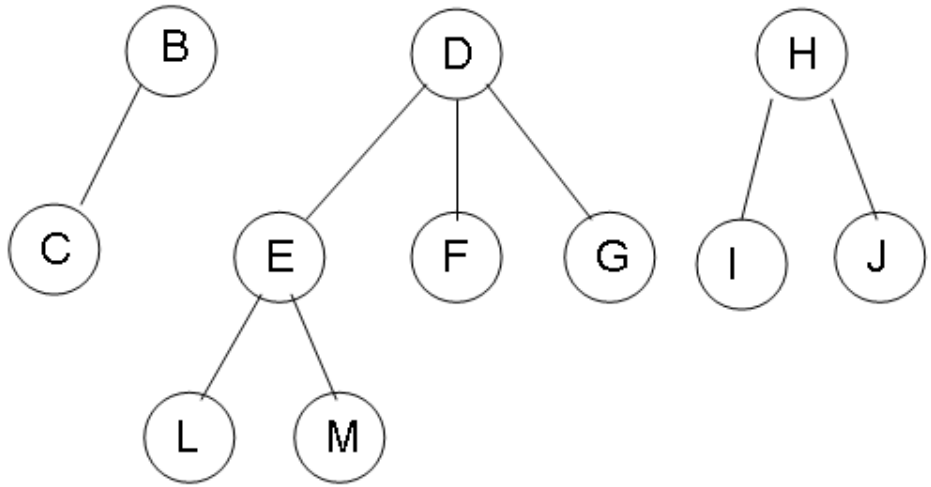
• Definição: Uma floresta F é uma sequência finita de árvores $(F = T_1, T_2, ..., T_m)$, $m \geq 0$ e cada T_i representa uma árvore da floresta.

– Observe que esta definição é recursiva pois uma árvore pode ser vista como a sua raiz juntamente com a floresta das suas subárvores.

• Exemplo: Retirando-se a raiz A da árvore anterior obtém-se uma floresta com raízes B, D, e H.



• Exemplo: Retirando-se a raiz A da árvore anterior obtém-se uma floresta com raízes B, D, e H.



• Representação de árvores gerais:

– Exemplo 1: Considera vetores de apontadores, prevendo-se um número máximo *GrauMax* de filhos em cada nó.

```
#define GRAU_MAX 10
typedef struct NoArvGeral  *ArvGeral ;
typedef struct NoArvGeral {
  int info ;          /* o campo info do nó */
  int grau ;          /* o grau do nó */
  ArvGeral filhos[GRAU_MAX] ;     /* os filhos do nó */
  } NoArvGeral ;
  ...
  p = malloc(sizeof(NoARvGeral)) ;
```

⇒ Eventual desperdício de memória.

• Representação de árvores gerais:

– Exemplo 1: Considera vetores de apontadores, prevendo-se um número máximo *GrauMax* de filhos em cada nó.

```
#define GRAU_MAX 10
typedef struct NoArvGeral  *ArvGeral ;
typedef struct NoArvGeral {
  int info ;          /* o campo info do nó */
  int grau ;          /* o grau do nó */
  ArvGeral filhos[GRAU_MAX] ;     /* os filhos do nó */
  } NoArvGeral ;
  ...
  p = malloc(sizeof(NoARvGeral)) ;
```

⇒ Eventual desperdício de memória.

– Exemplo 2: O número de componentes do vetor filho é indicado na alocação dinâmica.

```
typedef struct NoArvGeral  *ArvGeral ;
typedef struct NoArvGeral {
   int info ;          /* o campo info do nó */
   int grau ;         /* o grau do nó */
   ArvGeral filhos[1] ;
} NoArvGeral ;
...
p = malloc(sizeof(NoArvGeral)+(grau-1)*sizeof(ArvGeral)) ;
```

⇒ Maior dificuldade de manipulação em caso de operações que modifiquem o grau de um nó, tais como inserção e remoção.

– Exemplo 2: O número de componentes do vetor filho é indicado na alocação dinâmica.

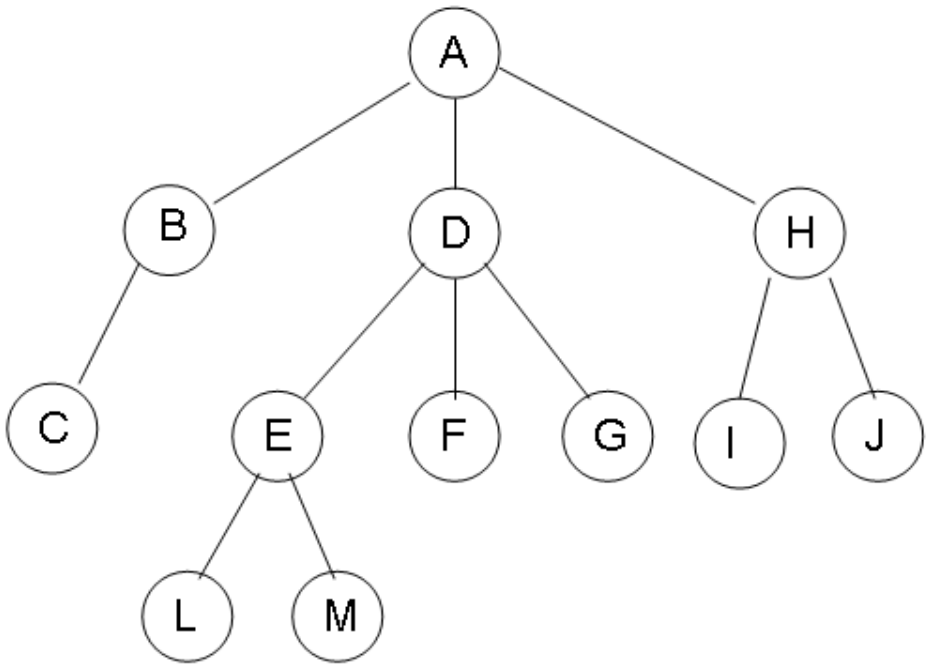
```
typedef struct NoArvGeral  *ArvGeral ;
typedef struct NoArvGeral {
   int info ;          /* o campo info do nó */
   int grau ;         /* o grau do nó */
   ArvGeral filhos[1] ;
} NoArvGeral ;
...
p = malloc(sizeof(NoArvGeral)+(grau-1)*sizeof(ArvGeral)) ;
```

⇒ Maior dificuldade de manipulação em caso de operações que modifiquem o grau de um nó, tais como inserção e remoção.

- Alternativa: Representação binária da árvore geral:

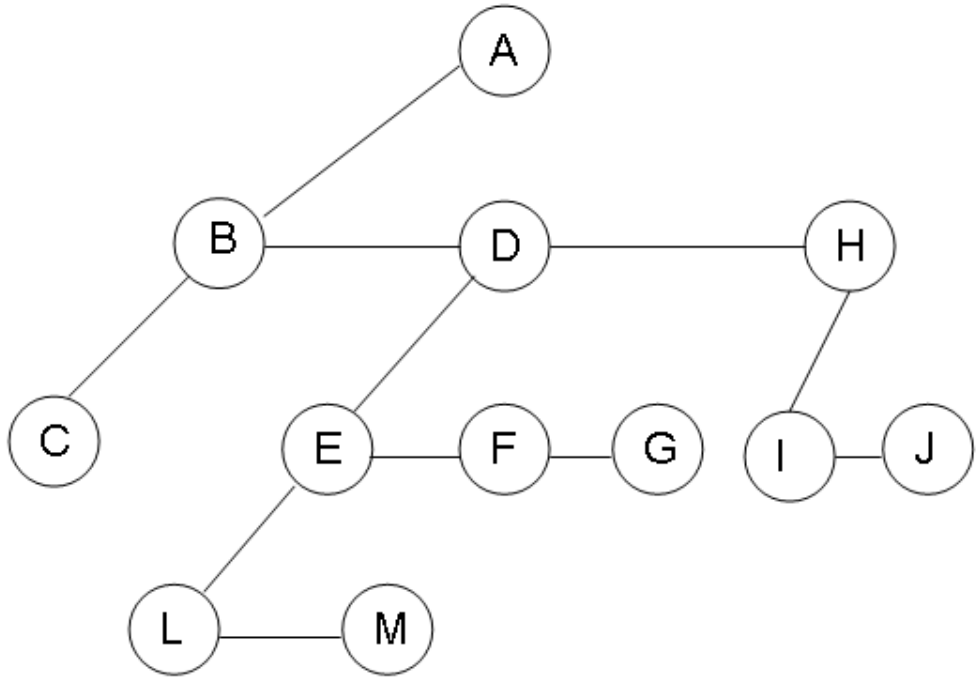
- Alternativa: Representação binária da árvore geral:

Passo 1: O filho mais à esquerda permanece ligado à esquerda do seu respectivo pai. Seus irmãos são colocados no mesmo nível.

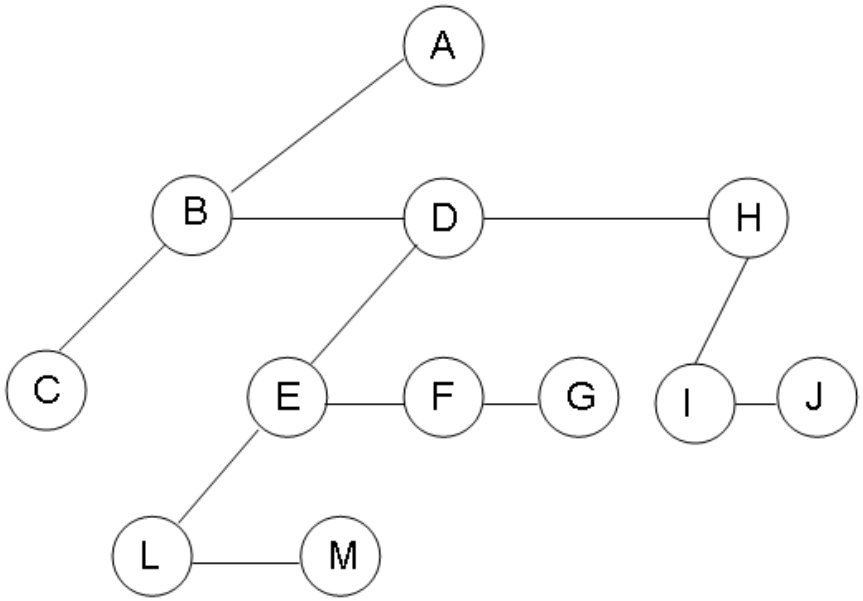


- Alternativa: Representação binária da árvore geral:

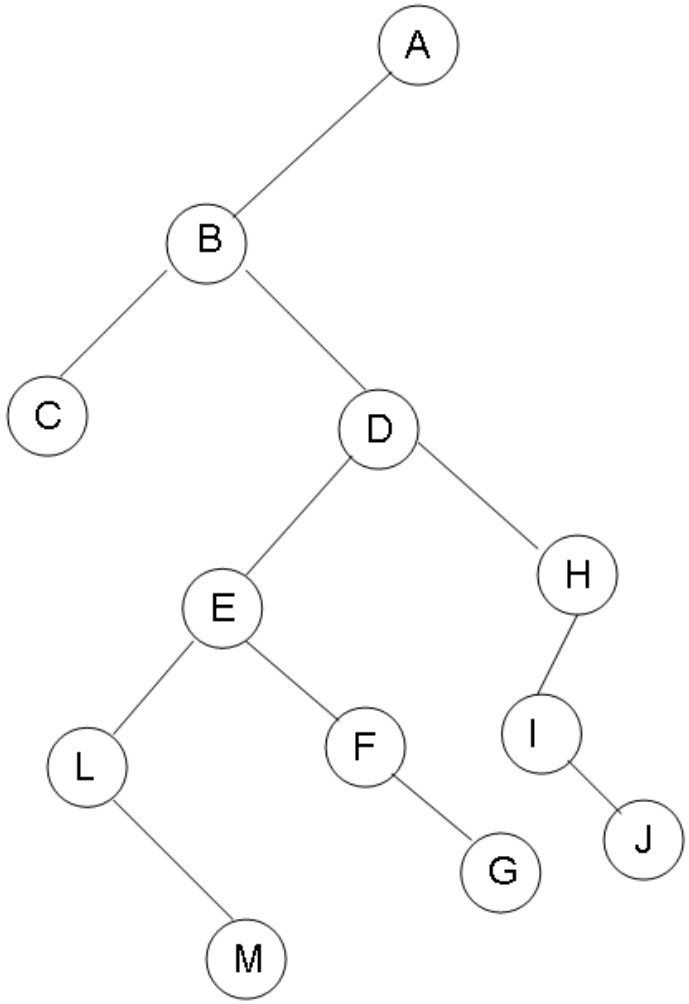
Passo 1: O filho mais à esquerda permanece ligado à esquerda do seu respectivo pai. Seus irmãos são colocados no mesmo nível.



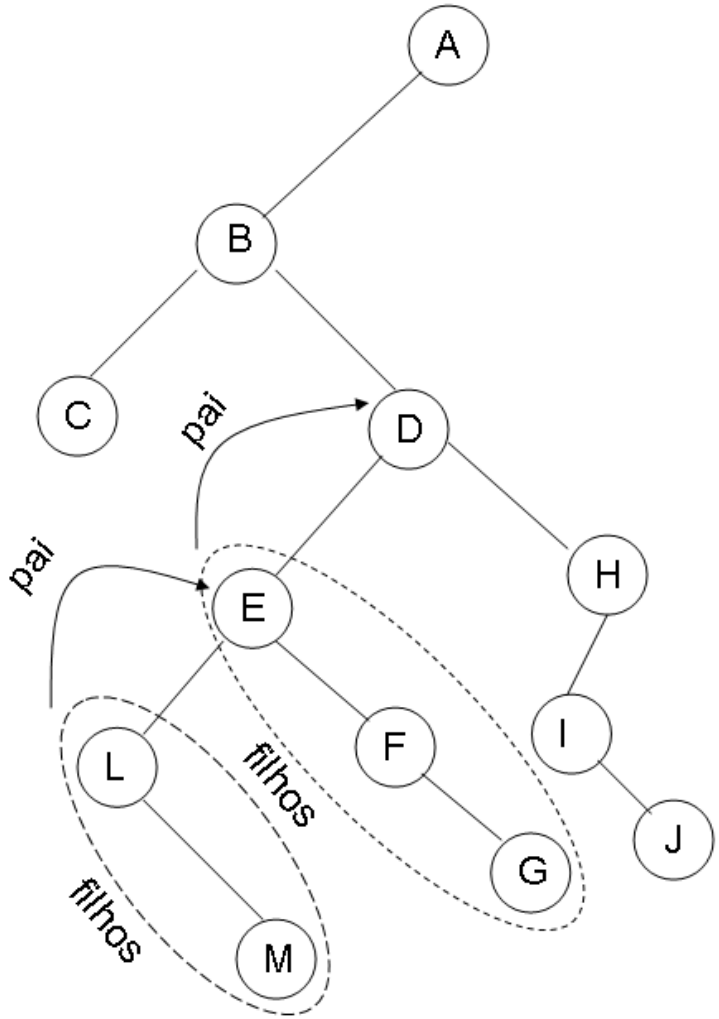
Passo 2: Obtém-se a árvore binária girando-se as linhas horizontais da representação anterior de -45^0.



Passo 2: Obtém-se a árvore binária girando-se as linhas horizontais da representação anterior de -45^0.



Passo 2: Obtém-se a árvore binária girando-se as linhas horizontais da representação anterior de -45^0.



chave	
esq	dir

esq = filho
dir = irmão

- Representação recursiva de florestas:

– Seja $F = (T_1, T_2, ..., T_m)$ uma floresta com $m \geq 0$. A árvore binária $B(F)$, representando F, é dada por:

1. Árvore vazia se F é uma árvore vazia ($m = 0$).
2. Árvore binária cuja raiz contém a mesma informação da raiz de T_1;
   1. cuja subárvore esquerda é dada por $B((T_{11}, T_{12}, ..., T_{1m1}))$, em que $(T_{11}, T_{12}, ..., T_{1m1})$ é a floresta das subárvores de T_1;
   2. cuja subárvore direita é dada por $B((T_2, ..., T_m))$.

Florestas

- Representação recursiva de florestas:

– Seja $F = (T_1, T_2, ..., T_m)$ uma floresta com $m \geq 0$. A árvore binária $B(F)$, representando F, é dada por:

1. Árvore vazia se F é uma árvore vazia ($m = 0$).
2. Árvore binária cuja raiz contém a mesma informação da raiz de T_1;
   1. cuja subárvore esquerda é dada por $B((T_{11}, T_{12}, ..., T_{1m1}))$, em que $(T_{11}, T_{12}, ..., T_{1m1})$ é a floresta das subárvores de T_1;
   2. cuja subárvore direita é dada por $B((T_2, ..., T_m))$.

- Representação recursiva de florestas:

– Seja $F = (T_1, T_2, ..., T_m)$ uma floresta com $m \geq 0$. A árvore binária $B(F)$, representando F, é dada por:

1. Árvore vazia se F é uma árvore vazia ($m = 0$).
2. Árvore binária cuja raiz contém a mesma informação da raiz de T_1;
   1. cuja subárvore esquerda é dada por $B((T_{11}, T_{12}, ..., T_{1m1}))$, em que $(T_{11}, T_{12}, ..., T_{1m1})$ é a floresta das subárvores de T_1;
   2. cuja subárvore direita é dada por $B((T_2, ..., T_m))$.

- Representação recursiva de florestas:

– Seja $F = (T_1, T_2, ..., T_m)$ uma floresta com $m \geq 0$. A árvore binária $B(F)$, representando F, é dada por:

1. Árvore vazia se F é uma árvore vazia ($m = 0$).
2. Árvore binária cuja raiz contém a mesma informação da raiz de T_1;
   1. cuja subárvore esquerda é dada por $B((T_{11}, T_{12}, ..., T_{1m1}))$, em que $(T_{11}, T_{12}, ..., T_{1m1})$ é a floresta das subárvores de T_1;
   2. cuja subárvore direita é dada por $B((T_2, ..., T_m))$.

- Representação recursiva de florestas:

– Seja $F = (T_1, T_2, ..., T_m)$ uma floresta com $m \geq 0$. A árvore binária $B(F)$, representando F, é dada por:

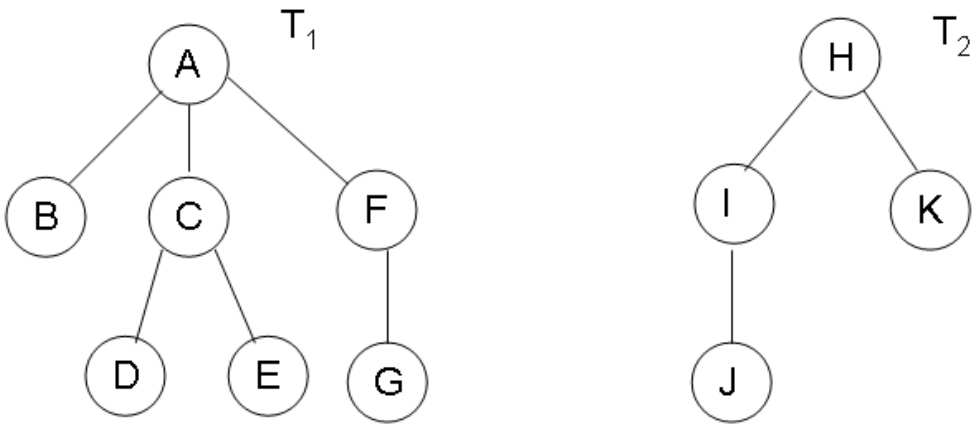
1. Árvore vazia se F é uma árvore vazia ($m = 0$).
2. Árvore binária cuja raiz contém a mesma informação da raiz de T_1;
   1. cuja subárvore esquerda é dada por $B((T_{11}, T_{12}, ..., T_{1m1}))$, em que $(T_{11}, T_{12}, ..., T_{1m1})$ é a floresta das subárvores de T_1;
   2. cuja subárvore direita é dada por $B((T_2, ..., T_m))$.

- Representação recursiva de florestas:

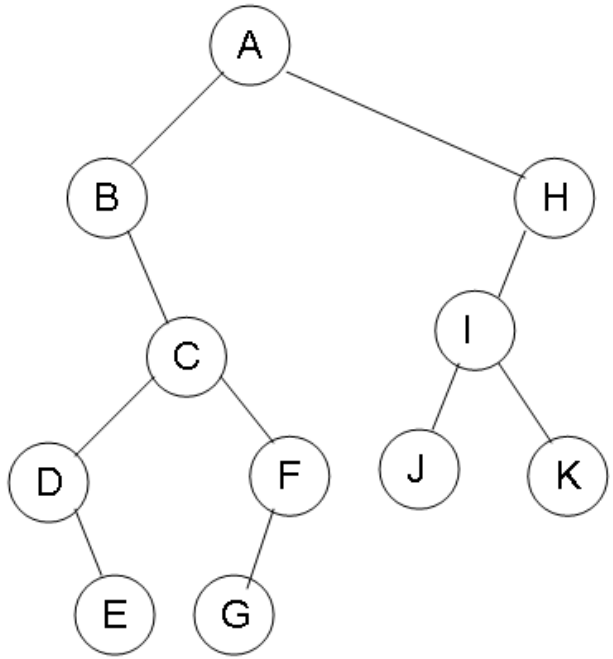
– Seja $F = (T_1, T_2, ..., T_m)$ uma floresta com $m \geq 0$. A árvore binária $B(F)$, representando F, é dada por:

1. Árvore vazia se F é uma árvore vazia ($m = 0$).
2. Árvore binária cuja raiz contém a mesma informação da raiz de T_1;
   1. cuja subárvore esquerda é dada por $B((T_{11}, T_{12}, ..., T_{1m1}))$, em que $(T_{11}, T_{12}, ..., T_{1m1})$ é a floresta das subárvores de T_1;
   2. cuja subárvore direita é dada por $B((T_2, ..., T_m))$.

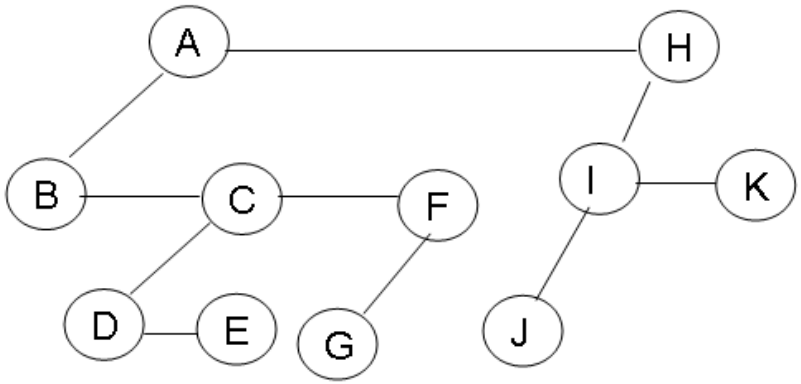
- Exemplo:



- Exemplo:



- Exemplo:



- Percursos de floresta:

- Percursos de floresta:

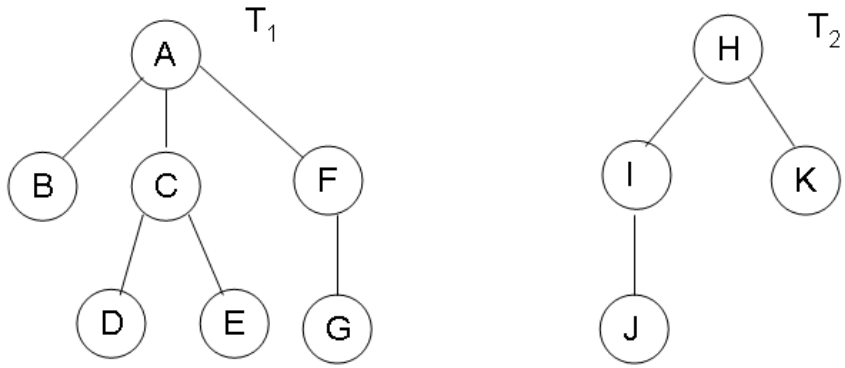
– Pré-ordem de floresta:

1. visitar a raiz da 1a árvore T_1 da floresta.
2. percorrer em pré-ordem a floresta das subárvores de T_1.
3. percorrer em pré-ordem as florestas $T_2, ..., T_n$.

- Percursos de floresta:

– Pré-ordem de floresta:

1. visitar a raiz da 1a árvore T_1 da floresta.
2. percorrer em pré-ordem a floresta das subárvores de T_1.
3. percorrer em pré-ordem as florestas $T_2, ..., T_n$.



pré-ordem: A B C D E F G H I J K

- Percursos de floresta:

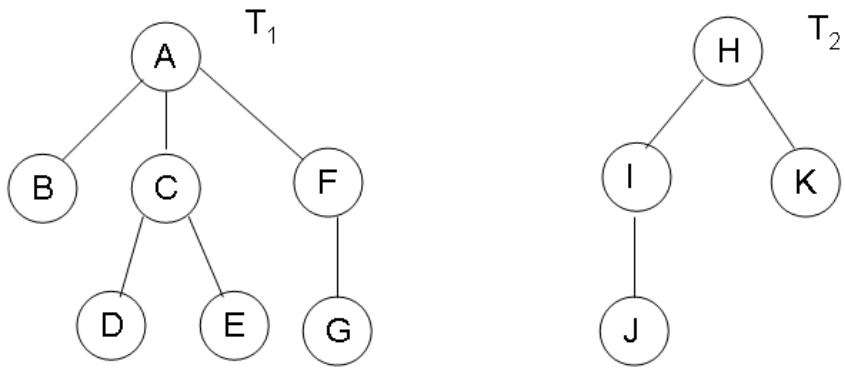
– Pós-ordem de floresta:

1. percorrer em pós-ordem a floresta das subárvores da 1a árvore T_1.
2. percorrer em pós-ordem as florestas $T_2, ..., T_n$.
3. visitar a raiz de T_1.

- Percursos de floresta:

– Pós-ordem de floresta:

1. percorrer em pós-ordem a floresta das subárvores da 1a árvore T_1.
2. percorrer em pós-ordem as florestas $T_2, ..., T_n$.
3. visitar a raiz de T_1.



pós-ordem: E D G F C B J K I H A

- Percursos de floresta:

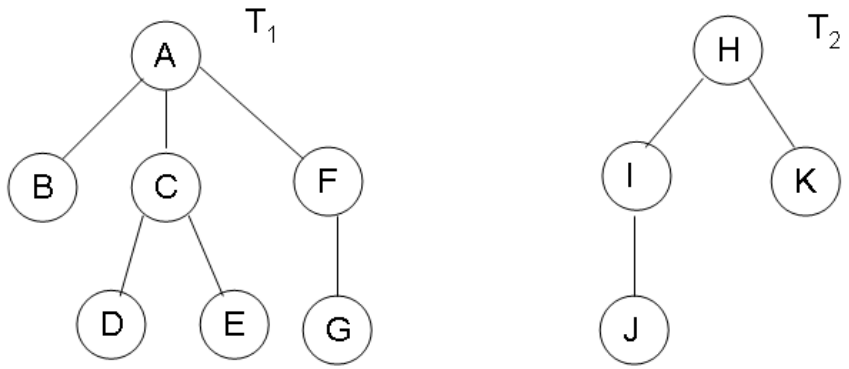
– Inordem de floresta:

1. percorrer em inordem a floresta das subárvores da 1a árvore T_1.
2. visitar a raiz de T_1.
3. percorrer em inordem as florestas $T_2, ..., T_n$.

- Percursos de floresta:

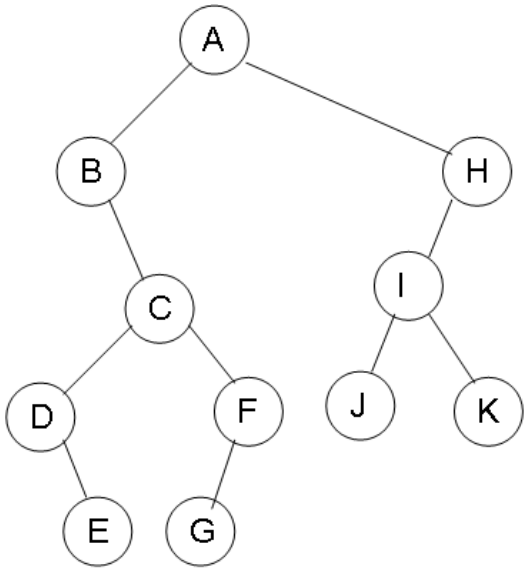
– Inordem de floresta:

1. percorrer em inordem a floresta das subárvores da 1a árvore T_1.
2. visitar a raiz de T_1.
3. percorrer em inordem as florestas $T_2, ..., T_n$.



inordem: B D E C G F A J I K H

• Todos os percursos anteriores correspondem àqueles da representação binária.



pré-ordem: A B C D E F G H I J K
pós-ordem: E D G F C B J K I H A
inordem: B D E C G F A J I K H

Pesquisa Digital

• Baseia-se na representação de chaves como uma sequência possivelmente variável de caracteres ou dígitos.

– Exemplo: Procurar palavras x associadas a um *alfabeto* de m letras.

$\Rightarrow$ Problema geral: Considera-se a existência de um conjunto de chaves $S = (s_1, ..., s_n)$ e um valor x correspondente a uma chave que se deseja localizar em S.

Pesquisa Digital

• Baseia-se na representação de chaves como uma sequência possivelmente variável de caracteres ou dígitos.

– Exemplo: Procurar palavras x associadas a um *alfabeto* de m letras.

$\Rightarrow$ Problema geral: Considera-se a existência de um conjunto de chaves $S = (s_1, ..., s_n)$ e um valor x correspondente a uma chave que se deseja localizar em S.

- Sejam:
  1. $S = (s_1, ..., s_n)$ um conjunto de n chaves em que cada s_i é formada por uma sequência de elementos d_j denominados *dígitos*.
  2. O *alfabeto* de S é constituído por um conjunto de m dígitos distintos.
  3. Os dígitos do alfabeto admitem ordenação: $d_1 < d_2 < ... < d_m$.

Definição: Uma árvore digital (ou *Trie*) de S é uma árvore *m-ária* T, não vazia, tal que:

1. Se um nó p é o j-ésimo filho de seu pai, então p corresponde ao dígito d_j do alfabeto de S, $1 \leq j \leq m$.
2. Para cada nó p, a sequência de dígitos definida pelo caminho desde a raiz de T até p corresponde a um prefixo de alguma chave de S.

- Sejam:
  1. $S = (s_1, ..., s_n)$ um conjunto de n chaves em que cada s_i é formada por uma sequência de elementos d_j denominados *dígitos*.
  2. O *alfabeto* de S é constituído por um conjunto de m dígitos distintos.
  3. Os dígitos do alfabeto admitem ordenação: $d_1 < d_2 < ... < d_m$.

Definição: Uma árvore digital (ou *Trie*) de S é uma árvore *m-ária* T, não vazia, tal que:

1. Se um nó p é o j-ésimo filho de seu pai, então p corresponde ao dígito d_j do alfabeto de S, $1 \leq j \leq m$.
2. Para cada nó p, a sequência de dígitos definida pelo caminho desde a raiz de T até p corresponde a um prefixo de alguma chave de S.

- Sejam:
  1. $S = (s_1, ..., s_n)$ um conjunto de n chaves em que cada s_i é formada por uma sequência de elementos d_j denominados *dígitos*.
  2. O *alfabeto* de S é constituído por um conjunto de m dígitos distintos.
  3. Os dígitos do alfabeto admitem ordenação: $d_1 < d_2 < ... < d_m$.

Definição: Uma árvore digital (ou *Trie*) de S é uma árvore *m-ária* T, não vazia, tal que:

1. Se um nó p é o j-ésimo filho de seu pai, então p corresponde ao dígito d_j do alfabeto de S, $1 \leq j \leq m$.
2. Para cada nó p, a sequência de dígitos definida pelo caminho desde a raiz de T até p corresponde a um prefixo de alguma chave de S.

- Sejam:
  1. $S = (s_1, ..., s_n)$ um conjunto de n chaves em que cada s_i é formada por uma sequência de elementos d_j denominados *dígitos*.
  2. O *alfabeto* de S é constituído por um conjunto de m dígitos distintos.
  3. Os dígitos do alfabeto admitem ordenação: $d_1 < d_2 < ... < d_m$.

Definição: Uma árvore digital (ou *Trie*) de S é uma árvore *m-ária* T, não vazia, tal que:

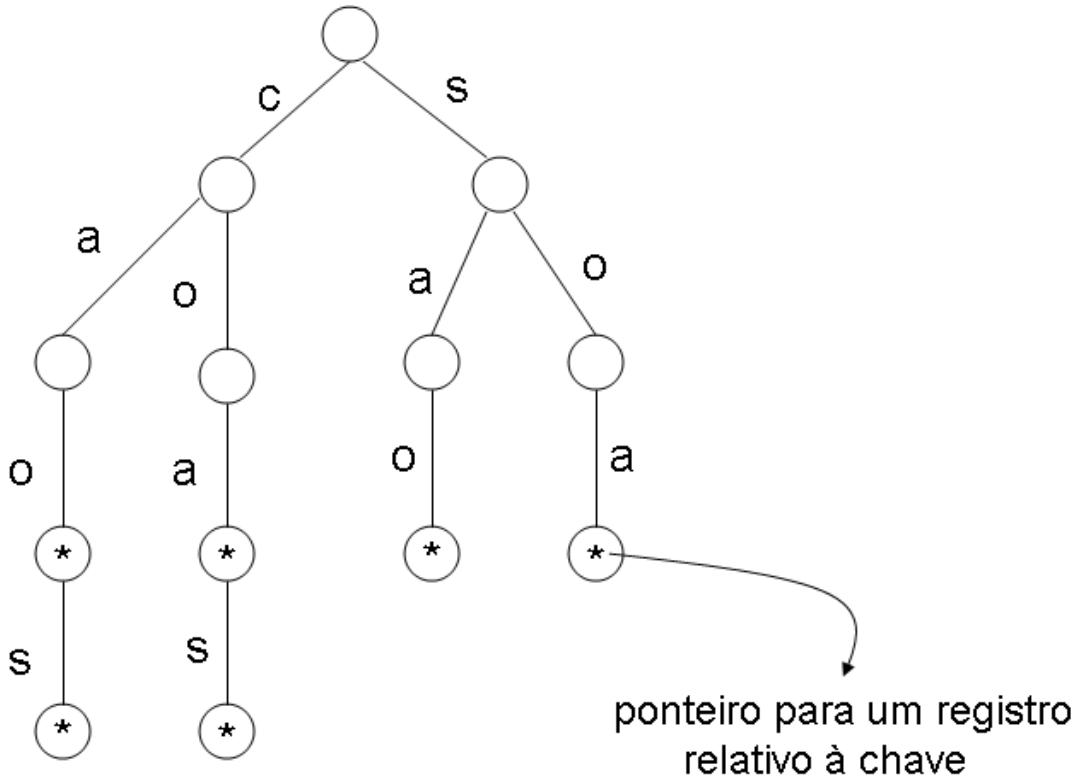
1. Se um nó p é o j-ésimo filho de seu pai, então p corresponde ao dígito d_j do alfabeto de S, $1 \leq j \leq m$.
2. Para cada nó p, a sequência de dígitos definida pelo caminho desde a raiz de T até p corresponde a um prefixo de alguma chave de S.

- Exemplo 1:

– $m = 4$; alfabeto $= \{a, c, o, s\}$;
– conjunto de chaves $S = \{cao, coa, coas, caos, soa, sao\}$:

- Exemplo 1:

– $m = 4$; alfabeto $= \{a, c, o, s\}$;
– conjunto de chaves $S = \{cao, coa, coas, caos, soa, sao\}$:



– Da condição 1 da definição anterior de uma árvore digital, tem-se que o j-ésimo filho de um nó qualquer da árvore, se existir, corresponde ao dígito d_j do alfabeto de S.

$\Rightarrow$ Pode-se eliminar as informações dos nós já implícitas na própria estrutura da árvore digital.

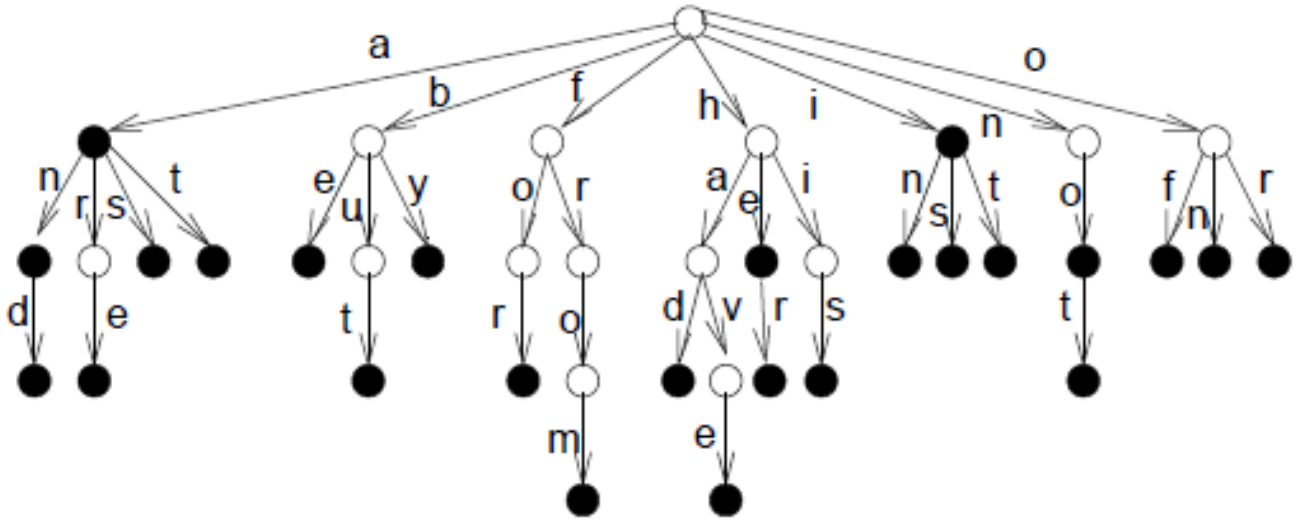
– Da condição 1 da definição anterior de uma árvore digital, tem-se que o j-ésimo filho de um nó qualquer da árvore, se existir, corresponde ao dígito d_j do alfabeto de S.

$\Rightarrow$ Pode-se eliminar as informações dos nós já implícitas na própria estrutura da árvore digital.

- Exemplo 2: Representação de cadeias de caracteres

```
a     at    from  his   no
an    be    had   i     not
and   but   have  in    of
are   by    he    is    on
as    for   her   it    or
```

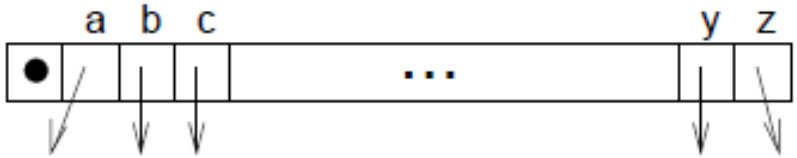
- Árvore digital:



Nós da forma ● indicam o fim de uma cadeia.

39 nós, 20 folhas, 19 nós não folhas, 25 nós com ●

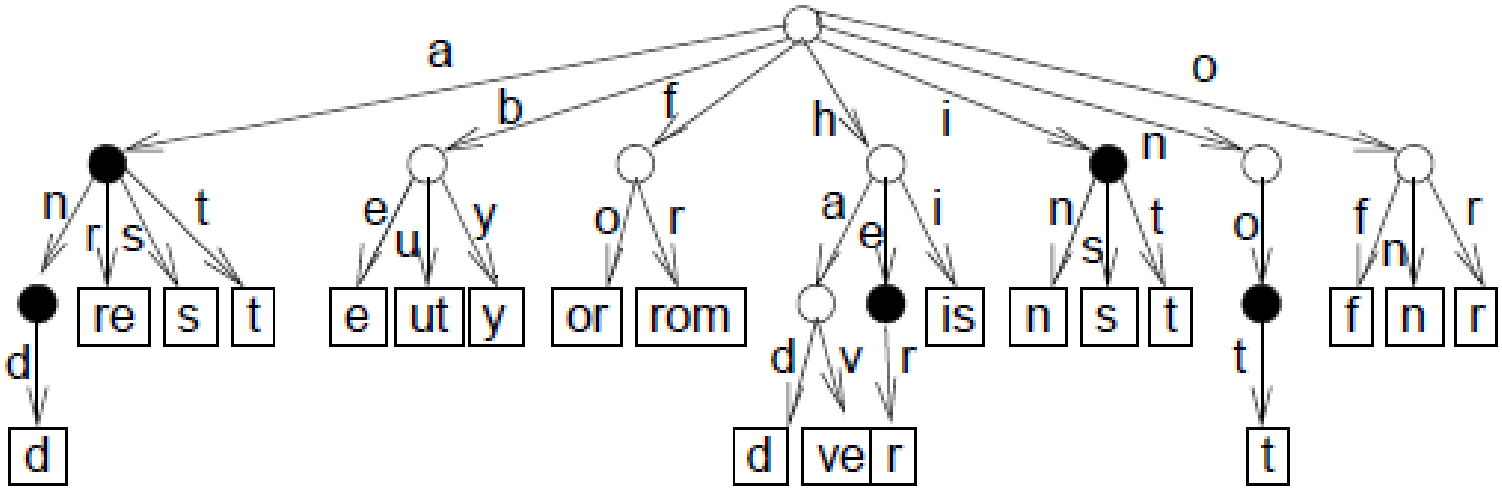
- Implementação da árvore digital:



  - desperdício de memória: dos $39 \times 26 = 1014$ campos, 38 são não nulos
  - simplificando as folhas: dos $19 \times 26 = 494$ campos, 38 são não nulos

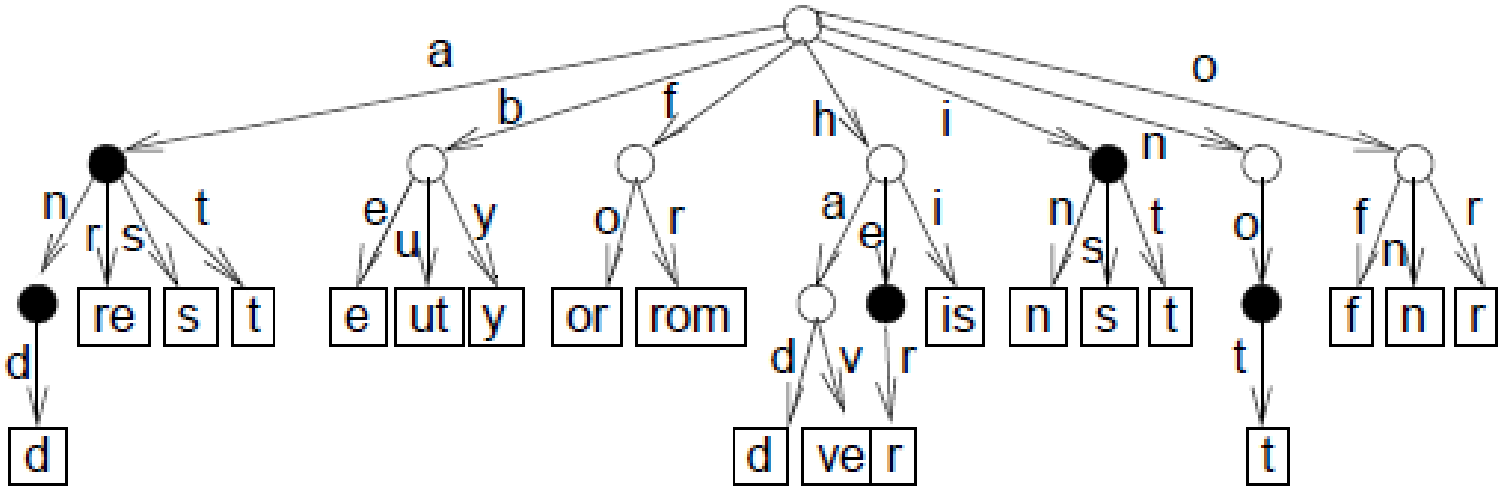
• Outra alternativa para se diminuir a memória necessária: juntando cadeias quando há uma única possibilidade de se continuar na árvore.

– Árvore digital modificada:



• Outra alternativa para se diminuir a memória necessária: juntando cadeias quando há uma única possibilidade de se continuar na árvore.

– Árvore digital modificada:

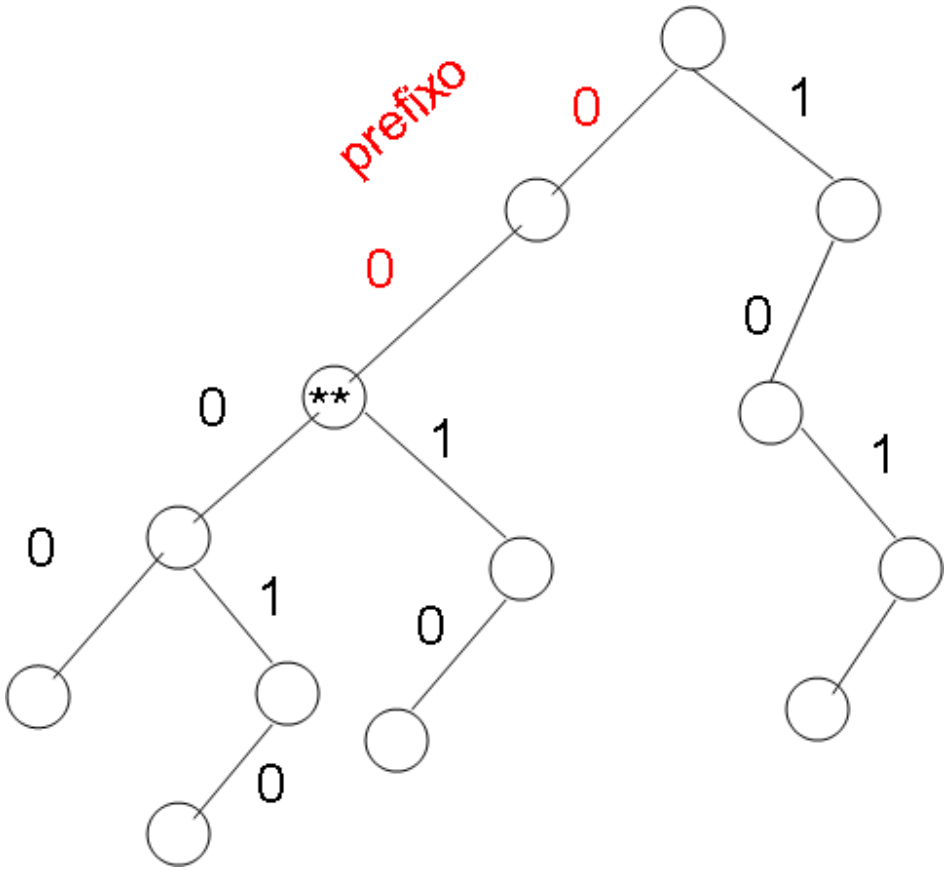


$\Rightarrow 12 \times 26 = 312$ campos, dos quais 31 não nulos.

Árvore Digital Binária

• Árvore digital com $m = 2$, em que cada chave x é uma sequência binária. Neste caso, o *alfabeto* é dado por $\{0, 1\}$.

– Exemplo: $S = \{00, 0000, 0010, 1010, 00010\}$



Árvore Binária de Prefixo

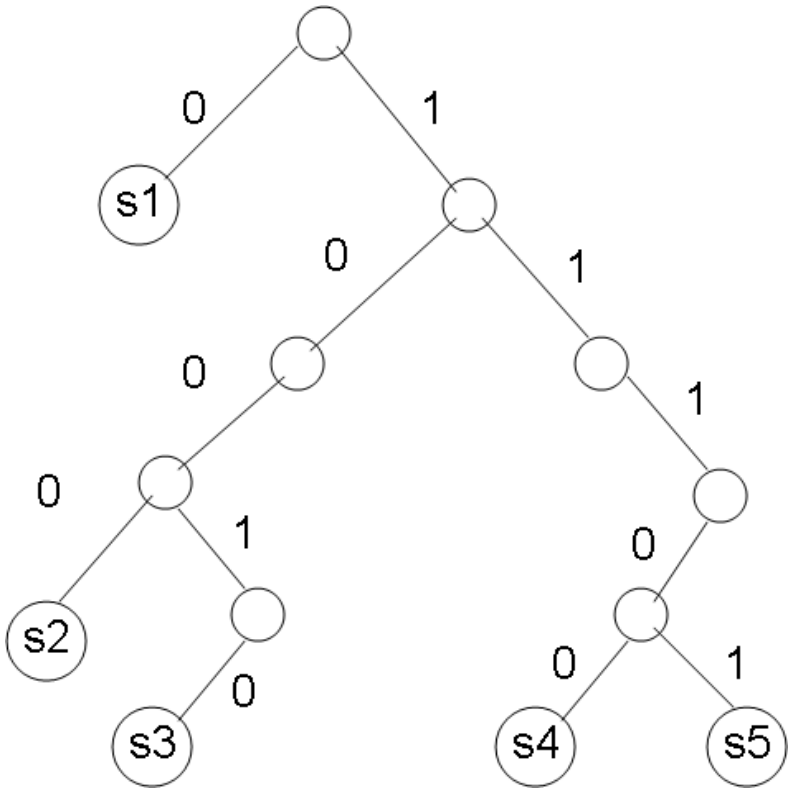
- Árvore digital binária tal que nenhum código é prefixo de outro.

–Exemplo: $S = \{s_1, s_2, s_3, s_4, s_5\} = \{0, 1000, 10010, 11100, 11101\}$

Árvore Binária de Prefixo

- Árvore digital binária tal que nenhum código é prefixo de outro.

–Exemplo: $S = \{s_1, s_2, s_3, s_4, s_5\} = \{0, 1000, 10010, 11100, 11101\}$



Criação da árvore digital

- Faz-se uma pesquisa na árvore com a chave a ser inserida:

  1. Se o nó externo em que a pesquisa terminar for vazio, cria-se um nó externo nesse ponto contendo a nova chave.

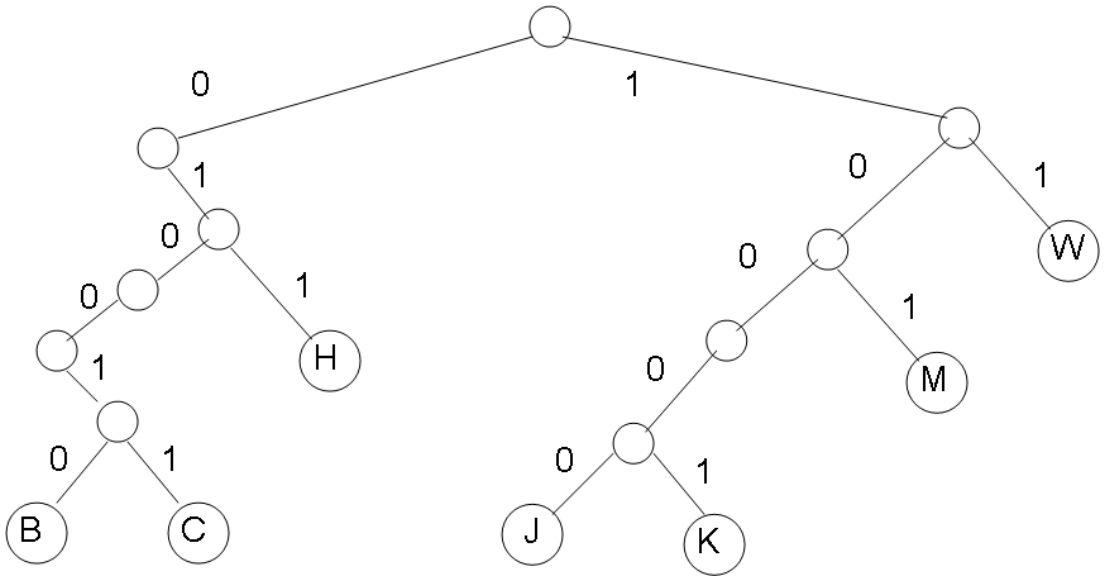
  2. Se o nó externo contiver uma chave, cria-se um ou mais nós internos cujos descendentes conterão a chave já existente e a nova chave

- Exemplo:

Criação da árvore digital (*trie*) binária contendo as seguintes chaves $S = \{B, C, H, J, M, W, K\}$
$= \{010010, 010011, 011000, 100001, 101000, 110110, 100010\}$

- Exemplo:

Criação da árvore digital (*trie*) binária contendo as seguintes chaves $S = \{B, C, H, J, M, W, K\}$
$= \{010010, 010011, 011000, 100001, 101000, 110110, 100010\}$



Referências:

– Jayme L. Szwarcfiter e Lilian Markenzon. Estruturas de Dados e seus Algoritmos.Editora LTC, 1994.

– Nívio Ziviani. Projeto de Algoritmos com Implementações em Pascal e C, Thomson Learning, 2004.

– Apostila dos professores Thomasz e Lucchesi.