Árvores

- As listas encadeadas usualmente fornecem maior flexibilidade que as matrizes, mas são estruturas lineares e é difícil usá-las para organizar uma representação hierárquica de objetos.
- As pilhas e as filas apresentam hierarquia limitada a uma dimensão.
- Para superar essa limitação as árvores são estruturas próprias para a representação de hierarquia.
- As árvores consistem de nós e arcos.
- As árvores são representadas de cima para baixo com a raiz no topo e as folhas na base.
- A raiz é um nó que não tem ancestrais tem somente filhos.
- As folhas não têm descendentes.
- Uma árvore T é um conjunto finito e não-vazio de nós com as seguintes propriedades:

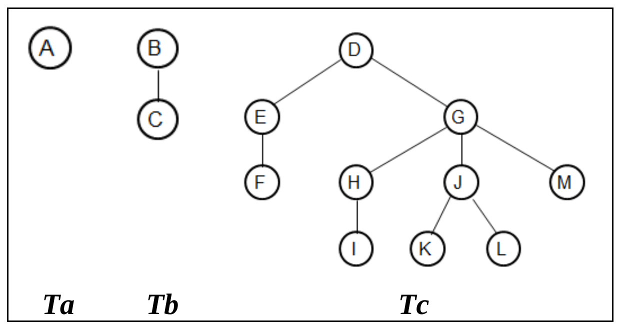
$$T = \{r\} \cup T_1 \cup T_2 \cup \ldots \cup T_n$$

  - um nó especial da árvore, r, é chamado de raiz da árvore; e
  - o restante dos nós é particionado $n \geq 0$ subconjuntos, T_1 , T_2 , …, T_n , cada um dos quais sendo uma árvore.
- A definição de uma árvore é recursiva, uma árvore é definida em termos dela mesma.
- Não há o problema de recursão infinita porque toda árvore tem um número finito de nós e no caso base uma árvore tem n=0 subárvores.
- Uma árvore minimal é composta de um único nó, a raiz.

Ta* = {*A*}, *Ta é uma árvore minimal

***Tb* = {*B*,{*C*}}**

***Tc* = {*D*,{*E*,{*F*}},{*G*,{*H*,{*I*}},{*J*,{*K*},{*L*}},{*M*}}}**



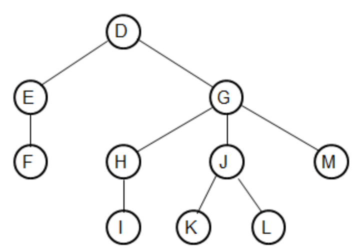
- O **grau** de um nó é o número de subárvores relacionadas com aquele nó.
  - Um nó de grau zero não possui subárvores.
- Cada raiz r_i da subárvore T_i é chamada de **filho** de r.
  - A definição de uma árvore não impõe qualquer condição sobre o número de filhos de um nó.
  - O número de filhos de um nó pode variar de 0 a qualquer inteiro.
- O nó raiz é o **pai** de todas as raízes r_i das subárvores T_i, $1 < i \leq n$.
- Duas raízes r_i e r_j das subárvores distintas T_i e T_j são ditas **irmãs**.
- Dada uma árvore T que contém um conjunto de nós, um **caminho** em T é definido como uma sequência não vazia de nós ou arcos

$$P=\{r_1, r_2, \ldots, r_k\}$$

onde $r_i \in R$, para $1 \leq i \leq k$ tal que o i-ésimo nó da sequência, r_i,é o pai do $(i+1)$-ésimo nó da sequência r_{i+1}.

- O número de arcos em um caminho é chamado de **comprimento do caminho** P e é definido por k-1.

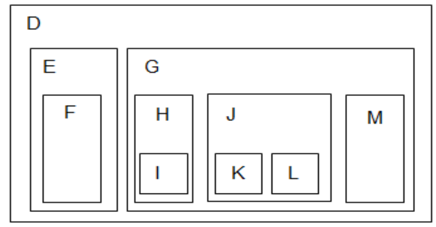
- Na árvore T há 29 caminhos diferentes incluindo os caminhos de comprimento 0; os caminhos de comprimento 1, e os caminhos de comprimento 3.



T

- O **nível** ou **profundidade** de um nó $r_i \in R$ da árvore T é o comprimento do único caminho em T entre a raiz r e o nó r_i .
    - A raiz T está no nível 0 e as raízes das subárvores estão no nível 1.
- A **altura** de um nó $r_i \in R$ numa árvore T é o comprimento do caminho mais longo do nó r_i a uma folha.
    - Folhas têm altura igual a 0.
    - A altura de uma árvore T é a altura do nó raiz r_i.
    - A árvore vazia é uma árvore legítima de altura 0 (por definição)
- Numa árvore T o nó r_i é um **ancestral** do nó r_j se existe um caminho em T de r_i a r_j .
- Numa árvore T o nó r_j é um **descendente** do nó r_i se existe um caminho em T de r_i a r_j .

- Uma árvore pode ser representada como um conjunto de reuniões aninhadas no plano (Diagrama de Venn).

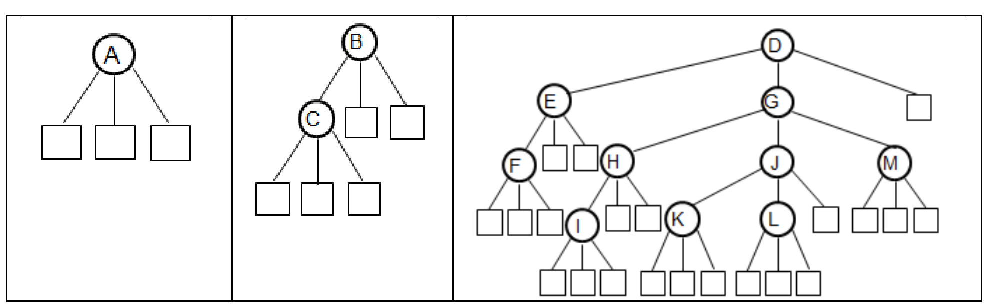


- Definições
    - **Nó pai** – nó ao qual um nó está ligado (diretamente).
    - **Nó filho** – cada um dos nós derivados de um nó pai.
    - **Nó ancestral** – todos os nós acima de um dado nó, em direção à raiz.
    - **Nó descendente** – todos os nós abaixo de um dado nó.
    - **Nós irmãos** – nós com o mesmo pai.
    - **Grau** – número de sub-árvores de um nó.
    - **Nó terminal** (**folha**) – nó sem filho ou com grau zero.
    - **Nível ou profundidade** – número de arcos entre um nó e a raiz.
    - **Altura** da árvore – nível mais alto.
    - **Floresta** – conjunto de árvores disjuntas.

Árvores N-árias

- Uma variação de árvores na qual todos os nós têm exatamente o mesmo grau.
- Não é possível construir uma árvore com um número finito de nós, todos com o mesmo grau N, exceto se $N=0$.
- Uma árvore N-ária T é um conjunto finito de nós com as seguintes propriedades:
- O conjunto é vazio, $T=\varnothing$; ou
- O conjunto consiste numa raiz, R, e exatamente N árvores N-árias distintas. Os nós remanescentes podem ser particionados em $N \geq 0$ subconjuntos, $T_0, T_1, \ldots, T_{N-1}$, cada um deles uma árvore N-ária tal que $T=\{R, T_0, T_1, \ldots, T_{N-1}\}$.
- Assim, os nós de uma árvore N-ária possuem grau 0 ou N.
- A árvore vazia $T=\varnothing$ é uma árvore, um objeto do mesmo tipo que a árvore não vazia.
- As árvores vazias são chamadas de nós externos por não possuírem subárvores e por isso aparecem nas extremidades das árvores.
- As árvores não vazias são chamadas de nós internos.

Árvores ternárias



- Os quadrados representam árvores vazias e os círculos denotam os nós não vazios.
- As árvores cujas subárvores estão ordenadas são **árvores ordenadas**.
- As árvores cuja ordenação não é importante são chamadas de **árvores orientadas**.

Árvores Binárias

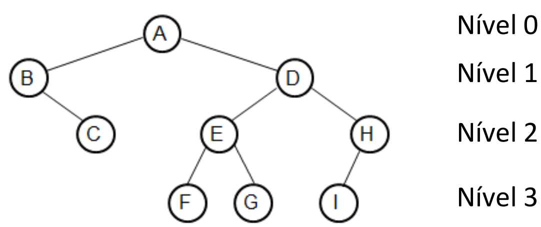
- Uma árvore binária é uma árvore N-ária para a qual $N=2$.
- Uma árvore binária T é um conjunto finito de nós com as seguintes propriedades:
  1. O conjunto é vazio, $T=\varnothing$; ou
  2. O conjunto consiste em uma raiz, r, e em exatamente duas árvores binárias distintas T_L e T_R, $T=\{r, T_L, T_R\}$.
- A árvore T_L é dita a subárvore da esquerda e a árvore T_R é dita a subárvore da direita.

Percurso em Árvores

- Grande parte dos algoritmos para a manipulação de árvores tem a característica comum de visitar sistematicamente todos os nós das árvores.
- O processo de visitação dos nós da árvore é chamado de percurso em árvore.
- Há duas formas para a realização de um percurso:
  - percurso em profundidade e
  - percurso em largura.

Percurso em profundidade: Alguns percursos em profundidade possuem nomes específicos:_

- percurso em pré-ordem,
- percurso em pós-ordem e
- percurso em ordem simétrica.



Percurso em pré-ordem

- A raiz é primeiro nó a ser visitado.
- O percurso é definido recursivamente.
  1. Visite o nó raiz.
  2. Percorra em pré-ordem cada uma das subárvores da raiz na ordem definida.
- Para uma árvore binária:
  1. Visite o nó raiz.
  2. Percorra em pré-ordem a subárvore esquerda.
  3. Percorra em pré-ordem a subárvore direita.
- Ex: A, B, C, D, E, F, G, H, I

Percurso em pós-ordem

- A raiz é último nó a ser visitado.
- O percurso é definido recursivamente.
  1. Percorra em pós-ordem cada uma das subárvores da raiz na ordem definida.
  2. Visite o nó raiz.
- Para uma árvore binária:
  1. Percorra em pós-ordem a subárvore esquerda.
  2. Percorra em pós-ordem a subárvore direita.
  3. Visite o nó raiz.
- Ex: C, B, F, G, E, I, H, D, A

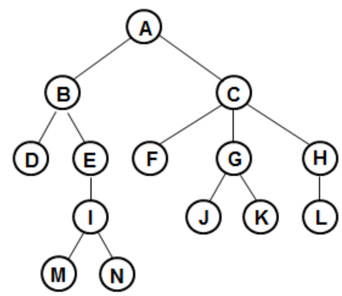
Percurso em Ordem Simétrica (in-ordem)

- O nó raiz é visitado entre as visitas das subárvores esquerda e direita.
- O percurso só faz sentido em árvores binárias.
  1. Percorra a subárvore esquerda.
  2. Visite o nó raiz.
  3. Percorra a subárvore direita.
- Ex: B, C, A, F, E, G, D, I, H

Percurso em Largura ou Extensão

- O percurso em extensão visita os nós na ordem dos níveis da árvore de cima para baixo ou de baixo para cima.
- Pode-se visitar os nós em cada nível da esquerda para a direita ou da direita para esquerda.
- Há 4 possibilidades.
- Ex: A, B, D, C, E, H, F, G, I

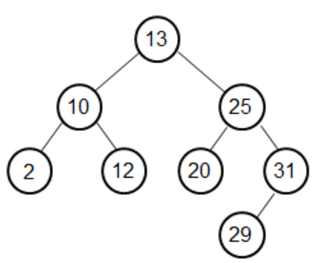
Exercícios



1. Quais nós são folhas. Qual é o nó raiz.
2. Que nó é pai de "C". Quais são os filhos de "C".
3. Quais são os antecessores de "E". Quais os sucessores de "E".
4. Qual a profundidade de "C". Qual a altura de "C".
5. Quantos caminhos diferentes de comprimento 1 existem. Quais são eles.
6. Liste os nós em pré-ordem, ordem simétrica, pós-ordem.

Implementação de árvores binárias

- As árvores binárias podem ser implementadas como matrizes e como estruturas encadeadas.
- Para implementar uma árvore como uma matriz um nó é declarado como uma estrutura com um campo de informação e dois campos de ponteiros.
- Os campos de ponteiros contém os índices das células da matriz em que os filhos à esquerda e à direita são armazenados se houver algum.



Índice	Info	Esq.	Dir.
0	13	4	2
1	31	6	-1
2	25	7	1
3	12	-1	-1
4	10	5	3
5	2	-1	-1
6	29	-1	-1
7	20	-1	-1

- A implementação de árvores baseadas em matrizes pode ser inconveniente em função da alocação estática.
- Essa característica é importante pois pode ser difícil prever quantos nós devem ser criados.
- Normalmente uma estrutura de dados dinâmica é o modo mais eficiente de representar uma árvore.
- Numa estrutura de dados dinâmica um nó é instância de uma classe composta por um membro de informação e dois ponteiros que apontem para as subárvores direita e esquerda.

Árvores de Busca

- São árvores que suportam operações eficientes de busca, inserção e remoção.
- A árvore armazena um número finito de chaves a partir de um conjunto de chaves K totalmente ordenado.
- Cada nó na árvore contém pelo menos uma chave e todas as chaves são diferentes.
- Numa árvore de busca há um critério de ordenação de dados.

Buscas em Árvores de Busca

- A vantagem principal de uma árvore de busca é que o critério de ordenação dos dados assegura que não é necessário fazer um percurso completo numa árvore para encontrar um valor.

Árvores M-Múltiplas de Busca

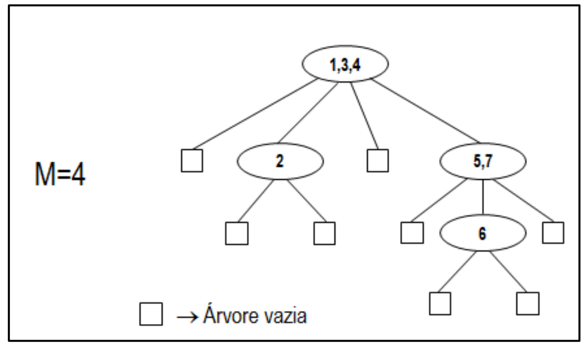
- Uma árvore M-múltipla de busca T é um conjunto finito de chaves onde ou T é vazio ou T consiste de n árvores M-múltiplas de busca T_0, T_1,... ,T_{n-1}, e n-1 chaves k_1, k_2,... ,k_{n-1},

$$T=\{T_0, k_1, T_1, k_2, T_2, \dots, k_{n-1}, T_{n-1}\}$$

- onde $2 \le n \le M$, tal que as chaves e os nós satisfazem as seguintes propriedades de ordenação:

1. As chaves em cada nó são distintas e ordenadas: $k_i < k_{i+1}$ para $1 \le i \le n$-1.
2. Todas as chaves das subárvores T_{i-1} são menores do que k_i. A árvore T_{i-1} é dita a subárvore da esquerda em relação a k_i.
3. Todas as chaves das subárvores T_{i+1} são maiores do que k_i. A árvore T_{i+1} é dita a subárvore da direita em relação a k_i. .

- Cada nó não vazio tem entre 1 e 3 chaves e no máximo 4 subárvores.
- As chaves de cada nó estão ordenadas.

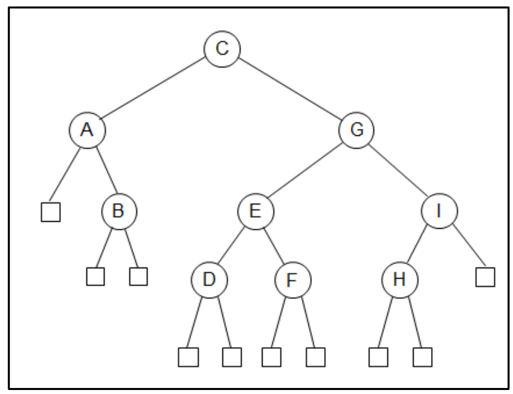


Árvores Binárias de Busca

- Uma árvore binária de busca T é um conjunto finito de chaves onde ou T é vazio ou T consiste em uma raiz e em exatamente 2 árvores binárias de busca T_L e T_R , tal que as seguintes propriedades são satisfeitas:

$$T=\{r, T_L, T_R\}$$

1. Todas as chaves da subárvore T_L são menores do que r. A árvore T_L é dita a subárvore da esquerda em relação a r.
2. Todas as chaves da subárvore T_R são maiores do que r. A árvore T_R é dita a subárvore da direita em relação a r.



Busca numa Árvore de Busca

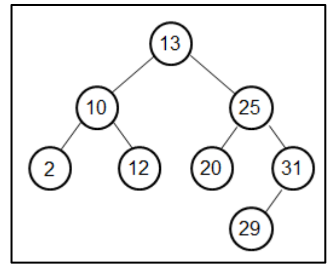
- A vantagem principal de uma árvore de busca é o critério de ordenação dos dados.
- Não é necessário fazer um percurso completo numa árvore para encontrar um valor.
- Busca em uma árvore M-múltipla de busca
    - Na busca de um valor x primeiramente examina-se as chaves do nó raiz.
    - Se x não estiver no nó raiz há 3 possibilidades:
        1. x é menor que k_1 , neste caso a busca deve prosseguir na subárvore T_0 ;
        2. x é maior que k_{n-1} , neste caso a busca deve prosseguir na subárvore T_{n-1} ;
        3. Ou existe um i tal que $1 \le i < n-1$ para o qual $k_i < x < k_{i+1}$, caso em que a busca deve prosseguir na árvore T_i .
- Para uma árvore M-múltipla de busca quando x não é encontrado em um certo nó, a busca continua apenas em uma das subárvores daquele nó.
- Uma busca com sucesso começa na raiz e percorre o caminho até o nó no qual a chave se encontra.
- Quando o objeto de busca não está na árvore, o método de busca descrito traça um caminho da raiz até uma subárvore vazia.

Busca em uma árvore binária de busca

- Algoritmo para localizar um elemento em uma árvore binária.
    - Para cada nó compare a chave a ser localizada com o valor armazenado no nó correspondente.
    - Se a chave for menor vá para a subárvore esquerda.

- Se a chave for maior vá para a subárvore direita.
- A busca para quando a chave é igual ao nó.
- O algoritmo também deverá ser parar se o valor não for encontrado.

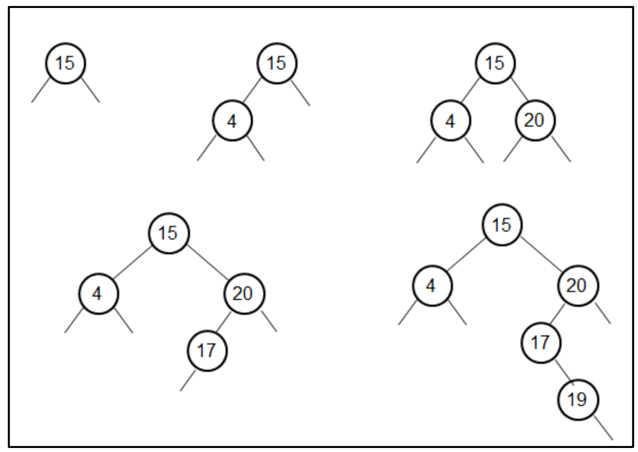
Para localizar o valor 31 apenas 3 testes são realizados.
O pior caso é quando se busca o valor 29.



Inserção de um nó em uma árvore binária

Inserção de um novo nó (*n_node*)

Faz-se *c_node* = raiz
Enquanto *c_node* <> nulo
{
Se *n_node* < *c_node*
c_node = filho esquerdo de *c_node*
Se *n_node* > *c_node*
c_node = filho direito de *c_node*
}
Anexar *n_node*
filho esquerdo de *n_node* = nulo
filho direito de *n_node*= nulo

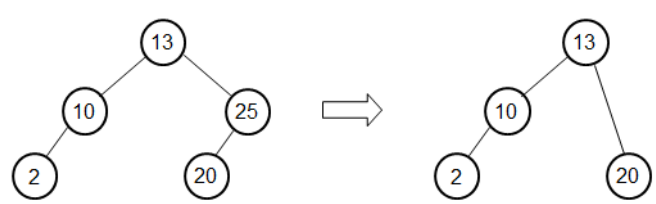


Remoção de um nó em uma árvore binária

- A remoção de um nó é outra operação necessária para se manter uma árvore binária de busca.
- O nível de complexidade depende da posição do nó a ser removido da árvore.
- Existem 3 casos de remoção de um nó da árvore binária de busca:
  - O nó é uma folha e não tem filhos.
    - O ponteiro de seu ascendente é ajustado para nulo e o nó é removido.
  - O nó tem um filho.
    - O ponteiro de seu ascendente é ajustado para apontar para o filho do nó e o nó é removido.
  - O nó tem dois filhos.
    - Nenhuma operação de uma etapa pode ser realizada, pois os ponteiros esquerdo e direito do ascendente não podem apontar para ambos os filhos do nó a ser removido.

O nó é uma folha e não tem filhos.

O nó tem um filho.



- Se o nó tem 2 filhos a remoção pode ser feita de duas formas: por fusão ou por cópia.

Remoção por fusão

- Uma árvore é extraída de duas subárvores do nó e esta árvore é anexada ao ascendente do nó.
- Os valores da subárvore direita são maiores do que os valores da subárvore esquerda.
- Na subárvore da esquerda o nó com o maior valor deve tornar-se ascendente da subárvore direita.
- O nó com o maior valor na subárvore da esquerda é o nó mais à direita desta subárvore.
- Este nó pode ser encontrado movendo-se ao longo da subárvore e tomando-se sempre os ponteiros direitos até encontrar-se o valor nulo.
- Simetricamente na subárvore da direita o nó com o menor valor deve tornar-se um ascendente da subárvore esquerda.
- Algoritmo

Se o nó não tem filhos à direita

  Anexe seu filho da esquerda à seu ascendente

Senão

  Se o nó não tem filhos à esquerda

    Anexe seu filho da direita à seu ascendente

  Senão {

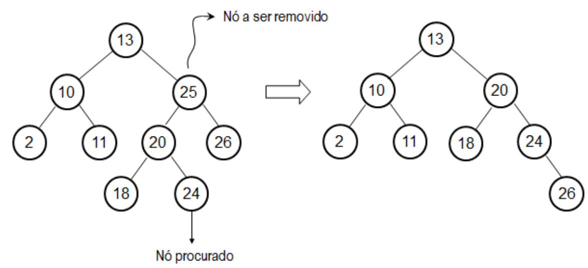
    Mova-se para a esquerda

    Mova-se para a direita até encontrar o valor nulo

    O nó encontrado será ascendente da subárvore direita

  }

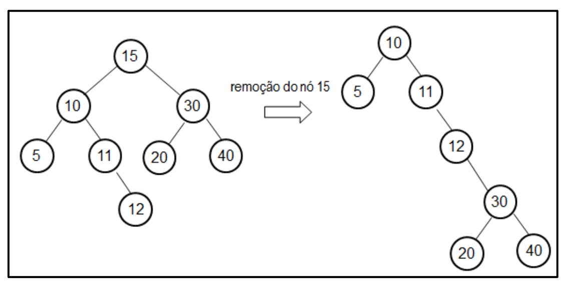
  Remova o nó



Implementação

```
if (node !=0) {
       if (node->right==0) node=node->left;
       else
          if (node->left==0) node=node->right;
          else {
             tmp=node->left;
             while (tmp->right != 0) tmp=tmp->right;
             tmp->right = node->right;
             tmp=node;
             node = node->left;
          }
        delete tmp;}
```

O algoritmo de remoção por fusão pode resultar no aumento da altura da árvore causando “desbalanceamento” da árvore.



Remoção de um nó em uma árvore binária por cópia

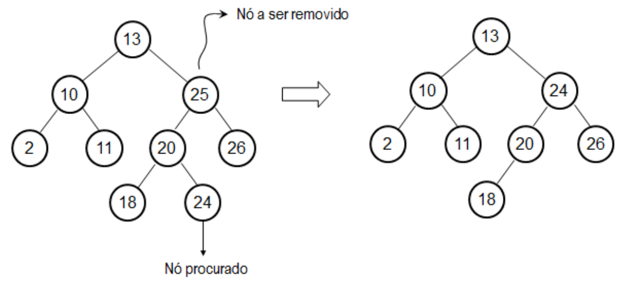
- A remoção por cópia remove uma chave *k1* sobreescrevendo-a por uma outra chave *k2* e então removendo o nó que contém *k2.* e então removendo o nó que contém
- Se o nó tem 2 filhos ele pode ser reduzido a um dos dois casos simples: ou o nó é uma folha ou o nó tem somente um filho não-vazio.
  - Pode-se substituir a chave que está sendo removida pelo seu predecessor (ou sucessor) imediato.
  - O predecessor de chave é a chave do nó mais à direita na subárvore da esquerda.
  - Analogamente, seu sucessor imediato é a chave do nó mais à esquerda na subárvore da direita.
  - Localiza-se o predecessor a partir da raiz da subárvore esquerda e movendo-se para direita o tanto quanto

possível até encontrar o valor nulo.

- Substitui-se a chave do predecessor pelo nó a ser removido.
- Dessa forma o nó se transforma em uma folha ou antecessor de apenas um filho não vazio.

Implementação

```
if (node->right==0) node=node->left;
 else
   if (node->left==0) node=node->right;
   else {
      tmp=node->left;
        prev = node;
        while (tmp->right != 0){
           prev=tmp;
           tmp=tmp->right;
      }
      node->key = tmp->key;
      if (prev->node==0) prev->left = tmp->left;
      else prev->right = tmp->left;
   delete tmp;}
```



Exercícios

Sobre árvores binárias de pesquisa:

1. Qual a principal propriedade deste tipo de árvore?
2. Desenhe a árvore binária de pesquisa que resulta da inserção sucessiva das chaves QUESTAOFCIL em uma árvore inicialmente vazia.
3. Desenhe as árvores resultantes das retiradas dos elementos E e depois U da árvore do item anterior por fusão.
4. Desenhe as árvores resultantes das retiradas dos elementos E e depois U da árvore do item anterior por cópia.

Exercícios para apresentar em sala

Elaborar um programa que apresente o seguinte menu:

1) Inserção em árvore binária
2) Remoção em árvore binária
3) Apresentação da árvore

A árvore deve ser apresentada da seguinte forma:

Raiz: 25 FE: 20 FD: 30
Nó 20: FE: 10 FD: 23
Nó 30: FE: 28 FD: 40
Nó 10: FE: 5 FD: 15
Nó 23: FE: -1 FD: -1
Nó 28: FE: -1 FD: -1
Nó 40: FE: -1 FD: -1
Nó 5: FE: -1 FD:-1
Nó 15: FE: -1 FD: -1