Estruturas de Dados II

Árvore Binária de Busca

Prof^a. Juliana de Santi
Prof. Rodrigo Minetto
Universidade Tecnológica Federal do Paraná
Material compilado de: Cormen, Notas de aula IC-UNICAMP e IME-USP

Sumário

Árvore Binária de Busca

Motivação: suponha um conjunto, potencialmente grande, de operações de inserção, remoção e busca de elementos.

Vetores ou listas são adequados?

Quais os problemas?

Árvore Binária de Busca

Existe alguma estrutura de dados que permite realizar as operações de inserção, remoção e busca de elementos de forma "eficiente" no caso médio?

Árvore **B**inária de **B**usca (**ABB**).

Árvore Binária de Busca

Os nós em uma árvore binária de busca (**ABB**) têm a seguinte estrutura (mínima):

- chave de comparação.
- endereço de sua sub-árvore esquerda.
- endereço de sua sub-árvore direita.

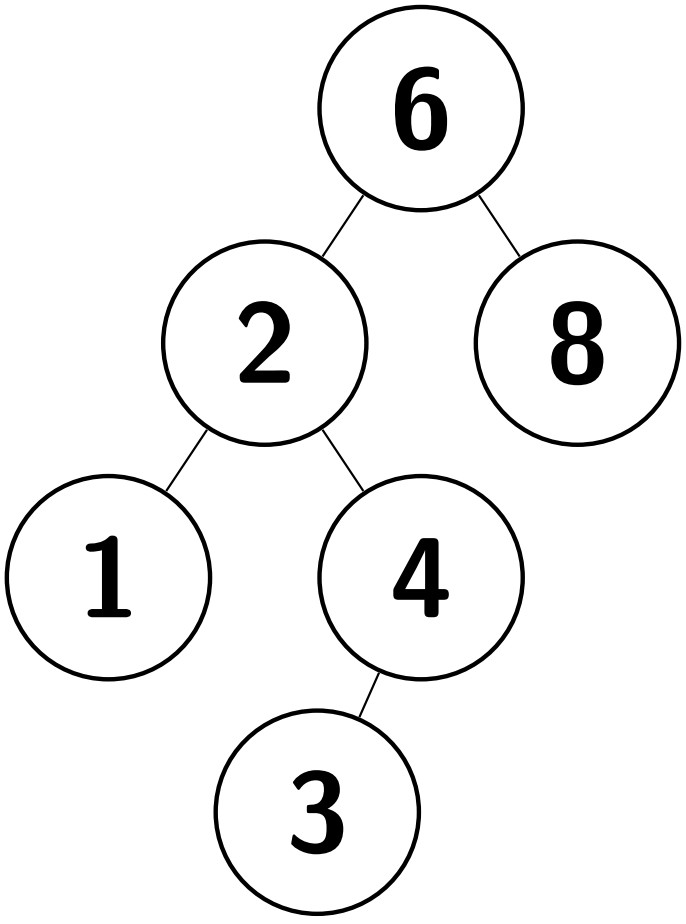
Árvore Binária de Busca

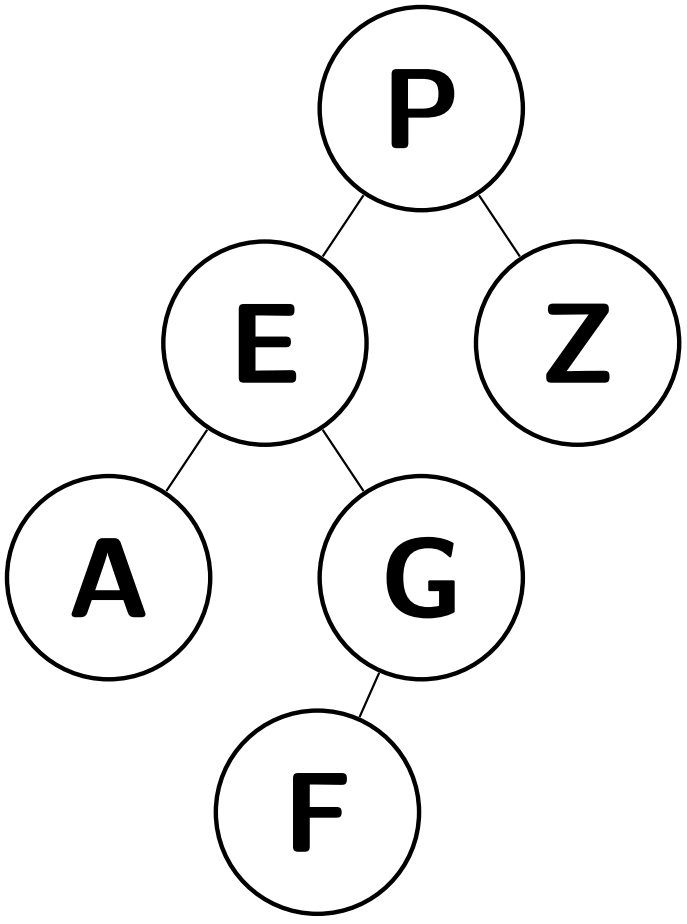
Definição: uma árvore com raiz r é **ABB** se:

- a chave de cada nó da sub-árvore esquerda de r é menor do que a chave do nó r;
- a chave de cada nó da sub-árvore direita de r é maior do que a chave do nó r;
- as chaves são únicas;
- sub-árvores esquerda e direita são **ABB**s.

Árvore Binária de Busca

Exemplos de **ABB**:



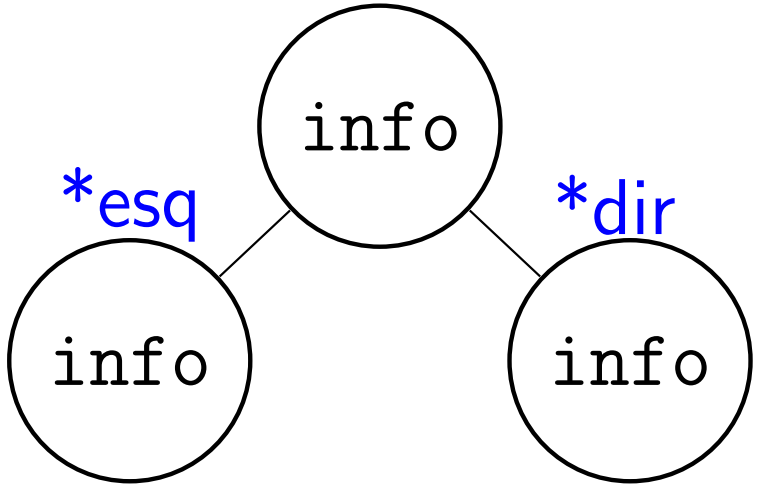


Sumário

Árvore Binária de Busca - Estrutura

Na linguagem C, uma estrutura para representar um **nó da árvore binária de busca** pode ser dada por:

```
typedef struct arvore {
    int info;
    struct arvore *esq;
    struct arvore *dir;
} Arvore;
```



Árvore Binária de Busca - Estrutura

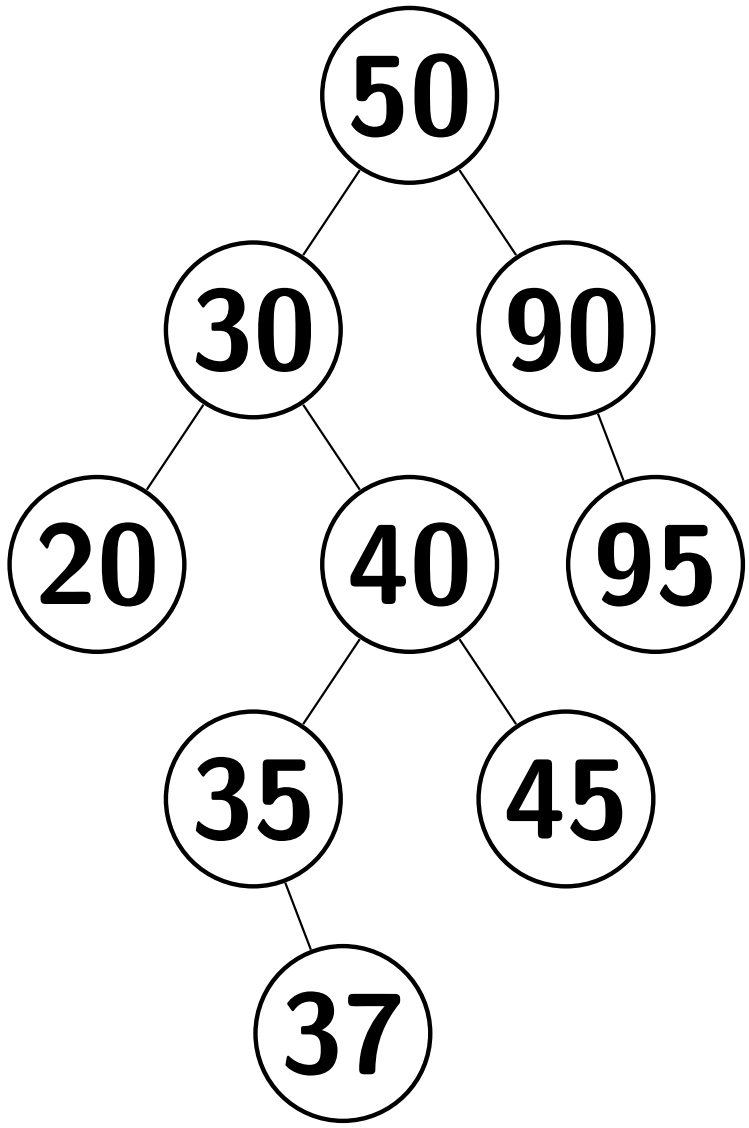
As operações **pesquisar**, **inserir** e **remover** **se baseiam na definição de uma ABB**, que diz que a chave do pai é sempre maior que os filhos à esquerda e menor que os filhos à direita. Assim, só existe **UMA** possibilidade na inserção que deixa a **ABB** correta. A remoção é mais complexa e é dividida em 3 casos: nó folha, nó com 1 filho, nós com 2 filhos.

Sumário

Árvore Binária de Busca - Pesquisar

Procurar o elemento '**37**' na **ABB**:

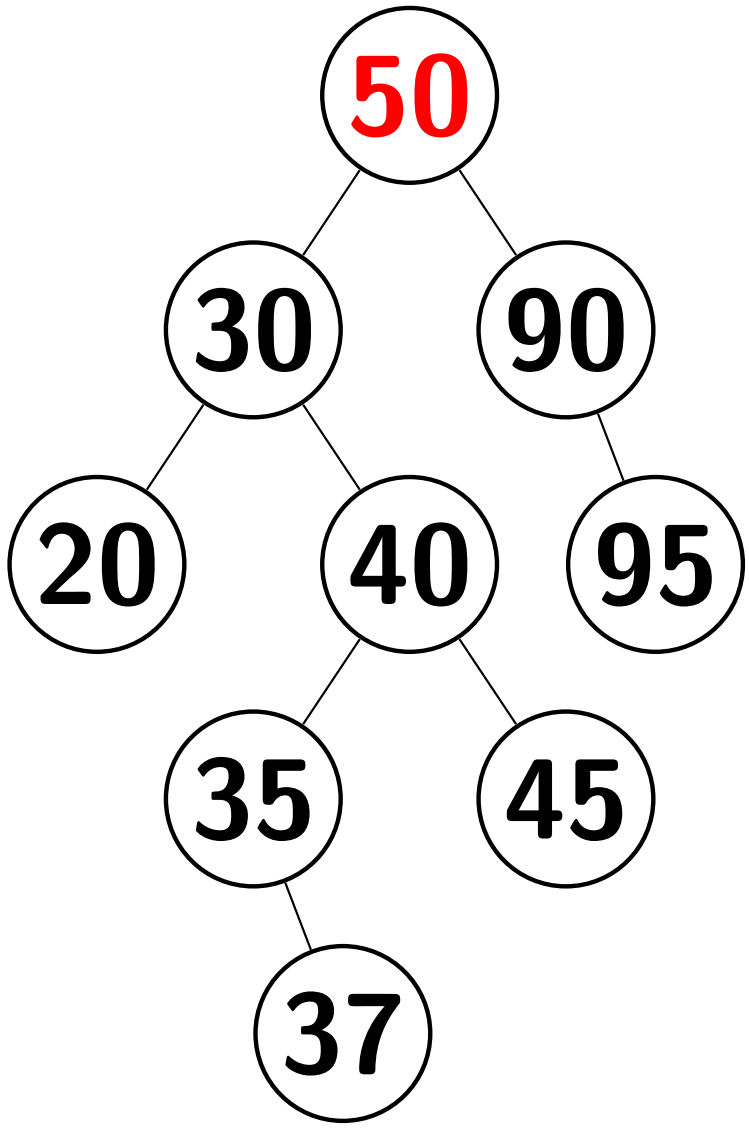
```
int BUSCAR (Arvore *a, int v)
if (a == NULL) {
    return 0;
}
else if (v < a->info) {
    return BUSCAR (a->esq, v);
}
else if (v > a->info) {
    return BUSCAR (a->dir, v);
}
else {
    return 1;
}
```



Árvore Binária de Busca - Pesquisar

Procurar o elemento '**37**' na **ABB**:

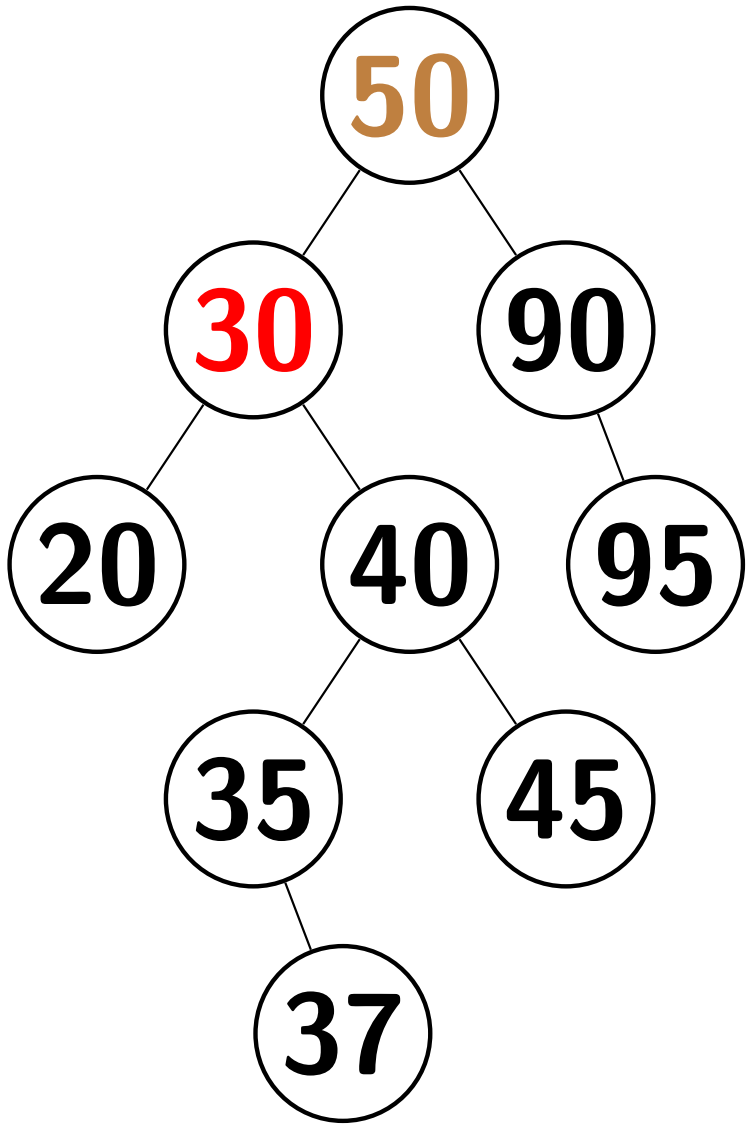
```
int BUSCAR (Arvore *a, int v)
if (a == NULL) {
    return 0;
}
else if (v < a->info) {
    return Buscar (a->esq, v);
}
else if (v > a->info) {
    return BUSCAR (a->dir, v);
}
else {
    return 1;
}
```



Árvore Binária de Busca - Pesquisar

Procurar o elemento '**37**' na **ABB**:

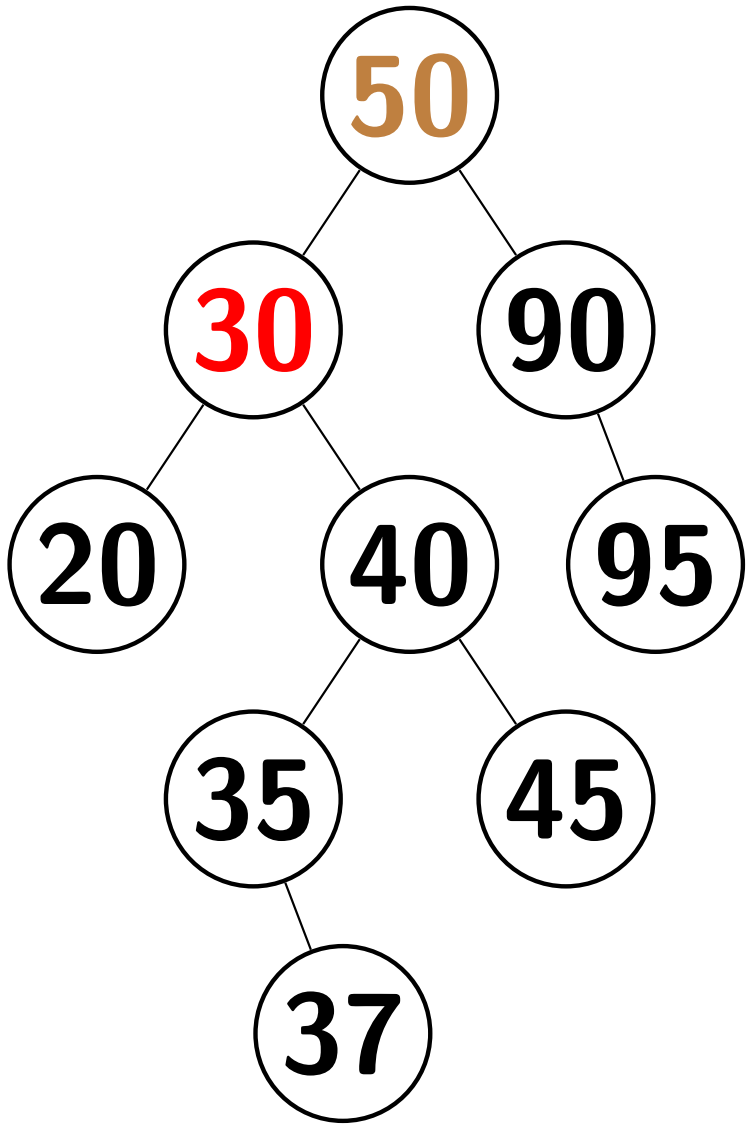
```
int BUSCAR (Arvore *a, int v)
if (a == NULL) {
    return 0;
}
else if (v < a->info) {
    return BUSCAR (a->esq, v);
}
else if (v > a->info) {
    return BUSCAR (a->dir, v);
}
else {
    return 1;
}
```



Árvore Binária de Busca - Pesquisar

Procurar o elemento '**37**' na **ABB**:

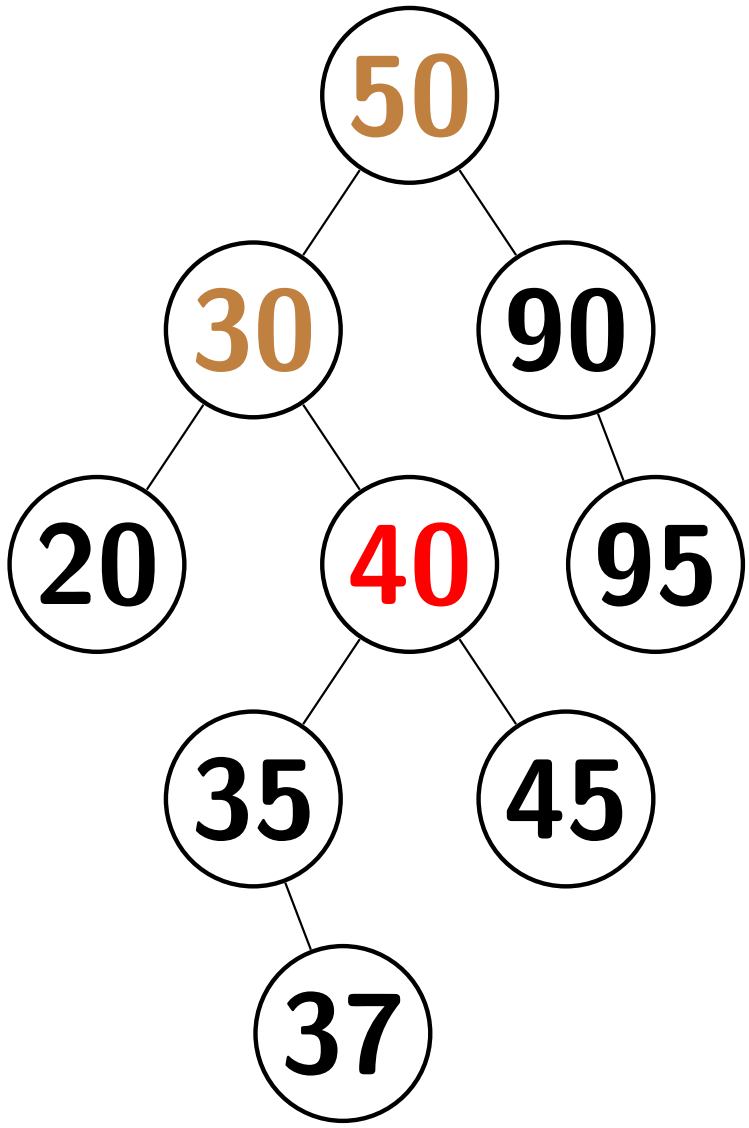
```
int BUSCAR (Arvore *a, int v)
if (a == NULL) {
    return 0;
}
else if (v < a->info) {
    return BUSCAR (a->esq, v);
}
else if (v > a->info) {
    return Buscar (a->dir, v);
}
else {
    return 1;
}
```



Árvore Binária de Busca - Pesquisar

Procurar o elemento '**37**' na **ABB**:

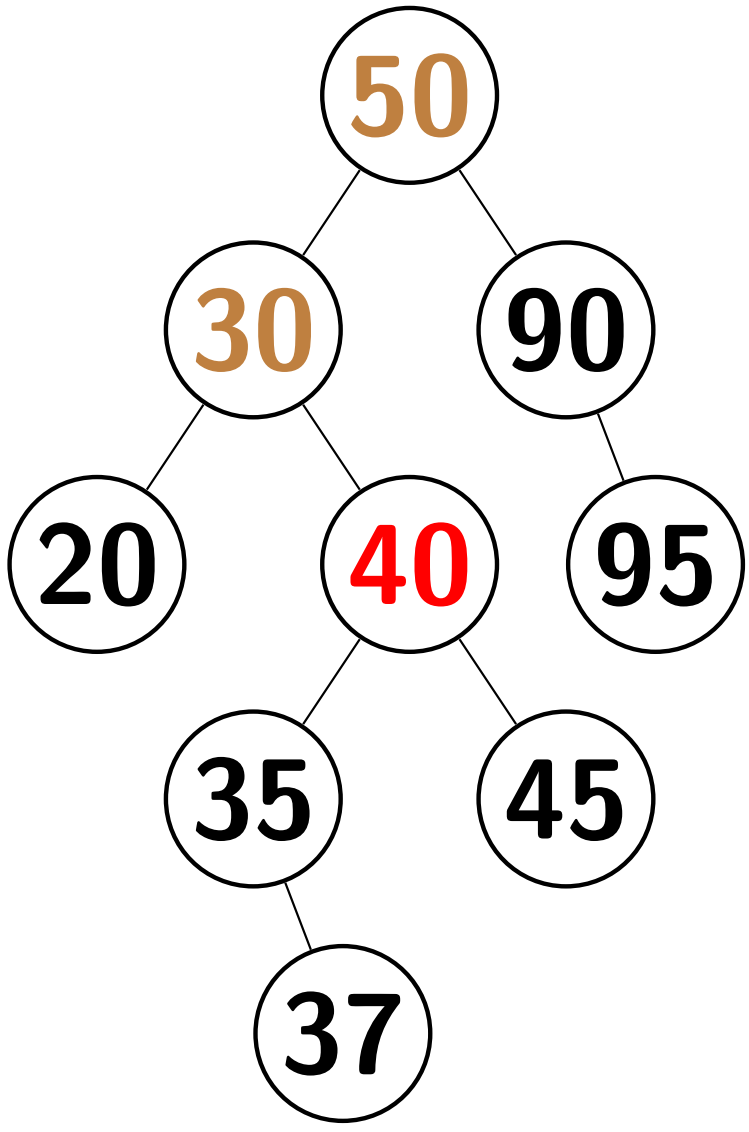
```
int BUSCAR (Arvore *a, int v)
if (a == NULL) {
    return 0;
}
else if (v < a->info) {
    return BUSCAR (a->esq, v);
}
else if (v > a->info) {
    return BUSCAR (a->dir, v);
}
else {
    return 1;
}
```



Árvore Binária de Busca - Pesquisar

Procurar o elemento '**37**' na **ABB**:

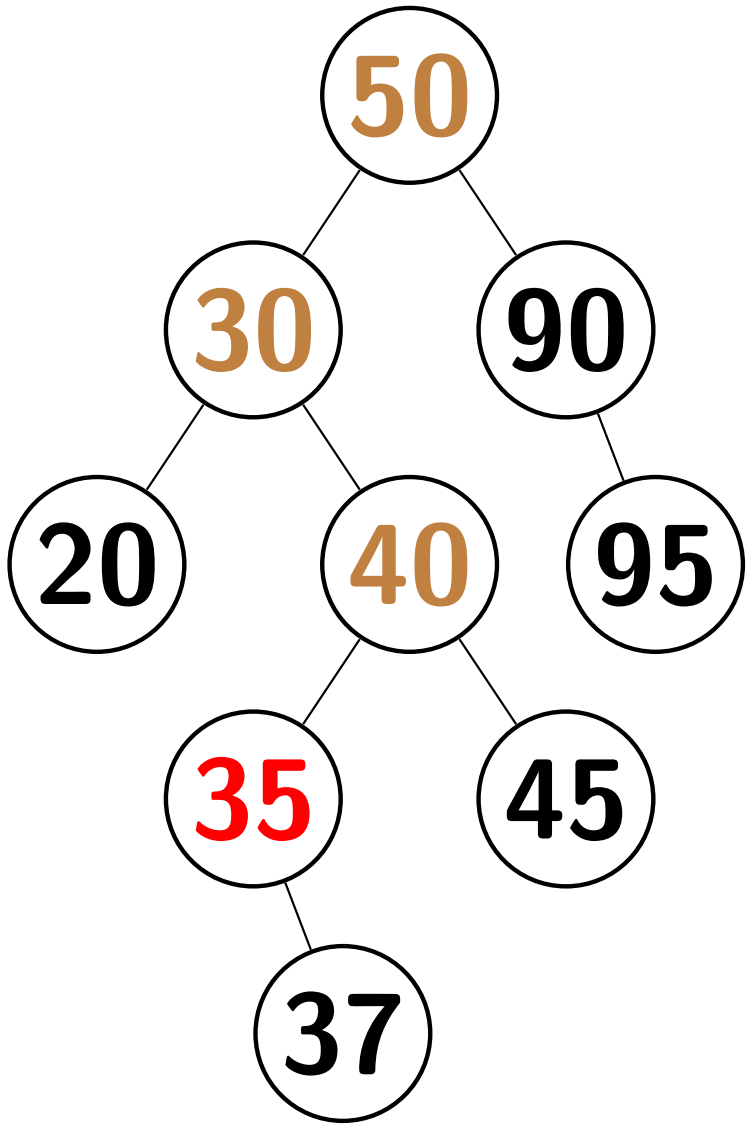
```
int BUSCAR (Arvore *a, int v)
if (a == NULL) {
    return 0;
}
else if (v < a->info) {
    return Buscar (a->esq, v);
}
else if (v > a->info) {
    return BUSCAR (a->dir, v);
}
else {
    return 1;
}
```



Árvore Binária de Busca - Pesquisar

Procurar o elemento '**37**' na **ABB**:

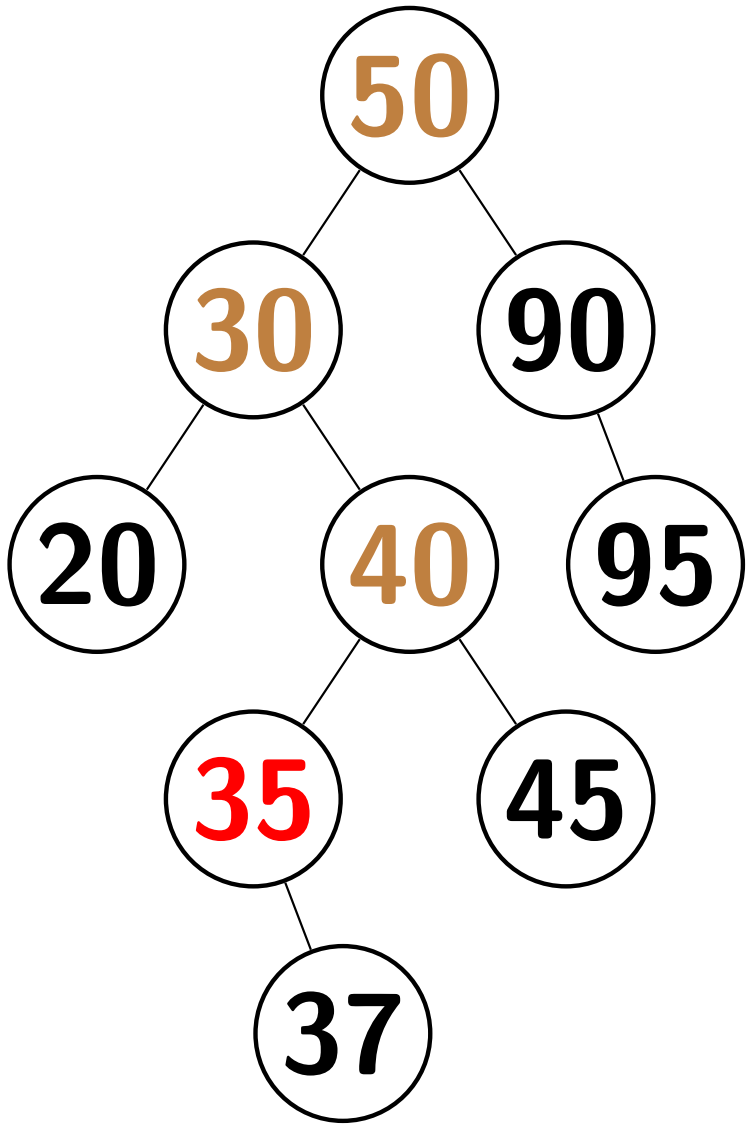
```
int BUSCAR (Arvore *a, int v)
if (a == NULL) {
    return 0;
}
else if (v < a->info) {
    return BUSCAR (a->esq, v);
}
else if (v > a->info) {
    return BUSCAR (a->dir, v);
}
else {
    return 1;
}
```



Árvore Binária de Busca - Pesquisar

Procurar o elemento '**37**' na **ABB**:

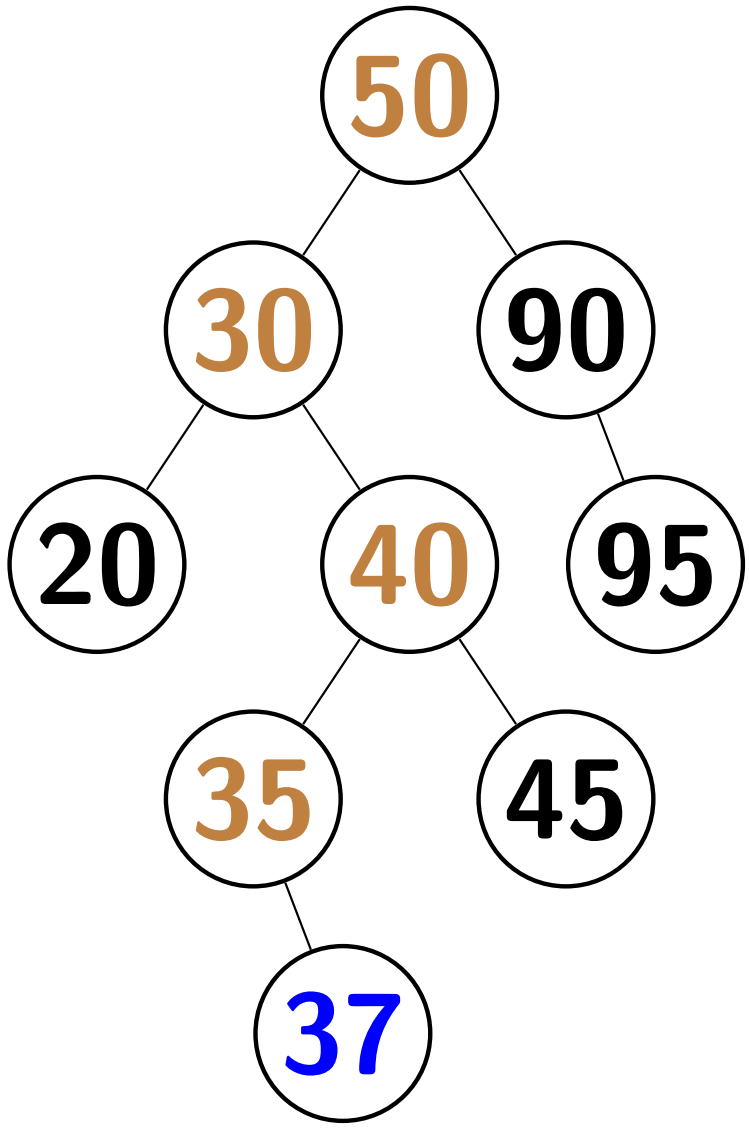
```
int BUSCAR (Arvore *a, int v)
if (a == NULL) {
    return 0;
}
else if (v < a->info) {
    return BUSCAR (a->esq, v);
}
else if (v > a->info) {
    return Buscar (a->dir, v);
}
else {
    return 1;
}
```



Árvore Binária de Busca - Pesquisar

Procurar o elemento '**37**' na **ABB**:

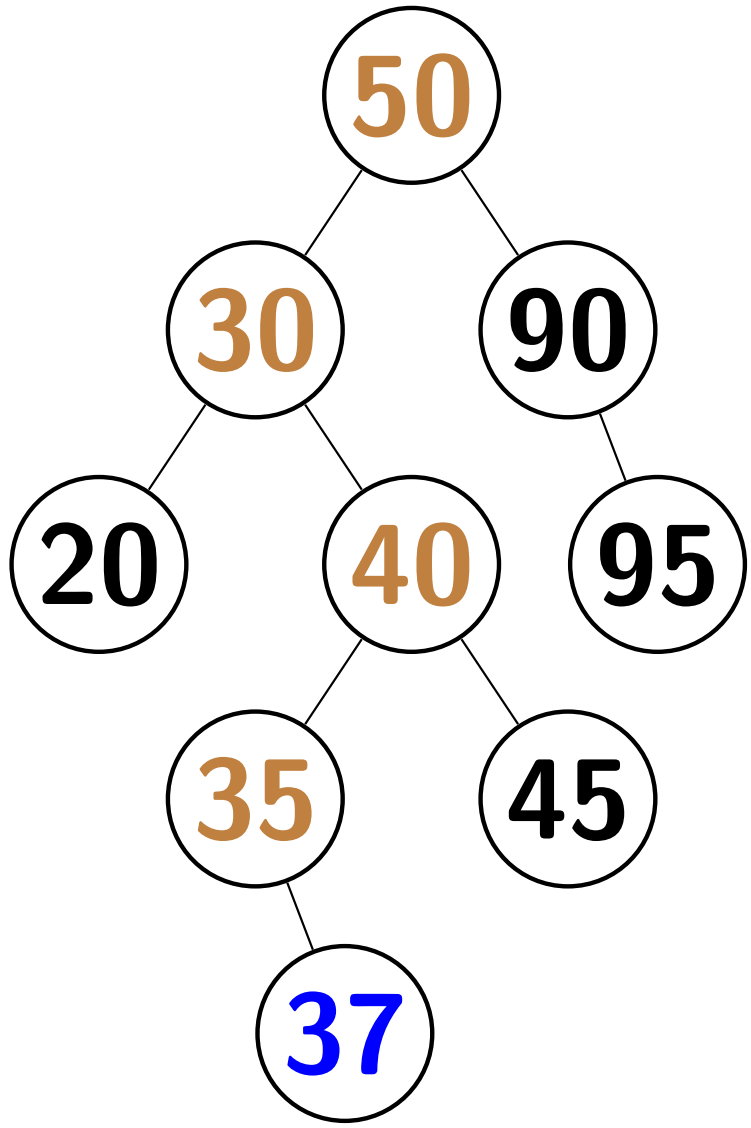
```
int BUSCAR (Arvore *a, int v)
if (a == NULL) {
    return 0;
}
else if (v < a->info) {
    return BUSCAR (a->esq, v);
}
else if (v > a->info) {
    return BUSCAR (a->dir, v);
}
else {
    return 1;
}
```



Árvore Binária de Busca - Pesquisar

Procurar o elemento '**37**' na **ABB**:

```
int BUSCAR (Arvore *a, int v)
if (a == NULL) {
    return 0;
}
else if (v < a->info) {
    return BUSCAR (a->esq, v);
}
else if (v > a->info) {
    return BUSCAR (a->dir, v);
}
else {
    return 1;
}
```



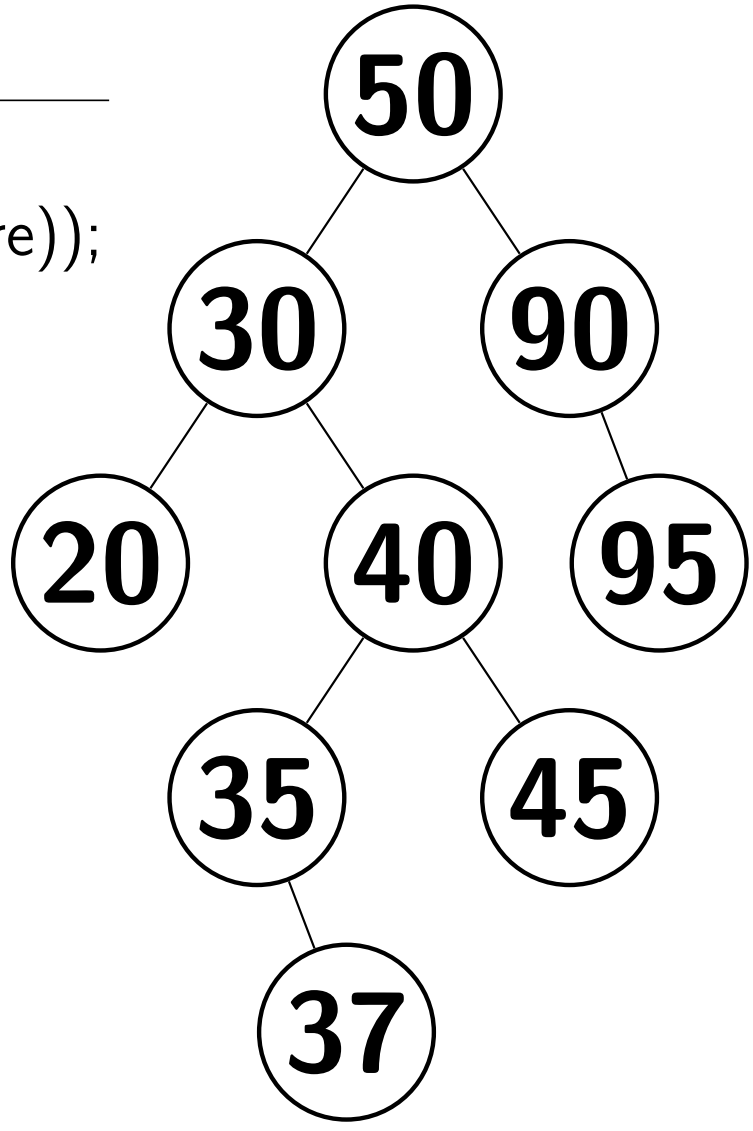
Sumário

Árvore Binária de Busca - Inserir

Inserir o elemento '47' na ABB:

Arvore* INSERIR (Arvore *a, int v)

```
if (a == NULL) {
    a = (Arvore*)malloc(sizeof(Arvore));
    a->info = v;
    a->esq = NULL;
    a->dir = NULL;
}
else if (v < a->info) {
    a->esq = INSERIR (a->esq, v);
}
else {
    a->dir = INSERIR (a->dir, v);
}
return a;
```

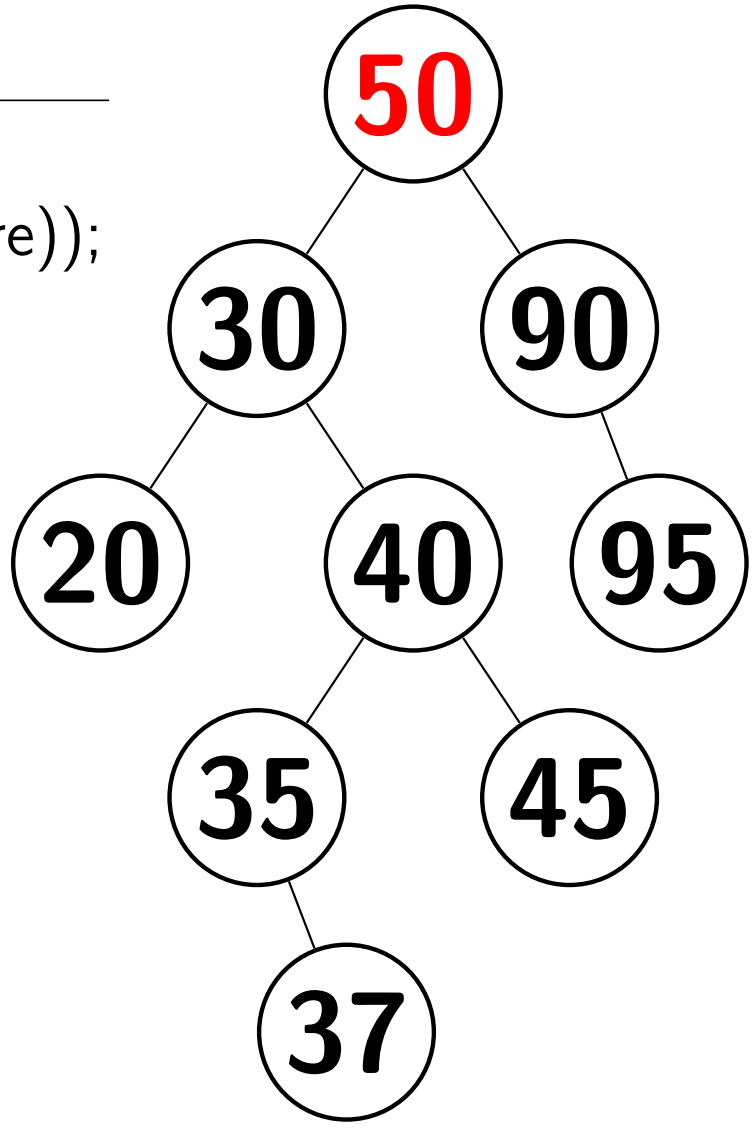


Árvore Binária de Busca - Inserir

Inserir o elemento '47' na ABB:

Arvore* INSERIR (Arvore *a, int v)

```
if (a == NULL) {
    a = (Arvore*)malloc(sizeof(Arvore));
    a->info = v;
    a->esq = NULL;
    a->dir = NULL;
}
else if (v < a->info) {
    a->esq = Inserir (a->esq, v);
}
else {
    a->dir = INSERIR (a->dir, v);
}
return a;
```

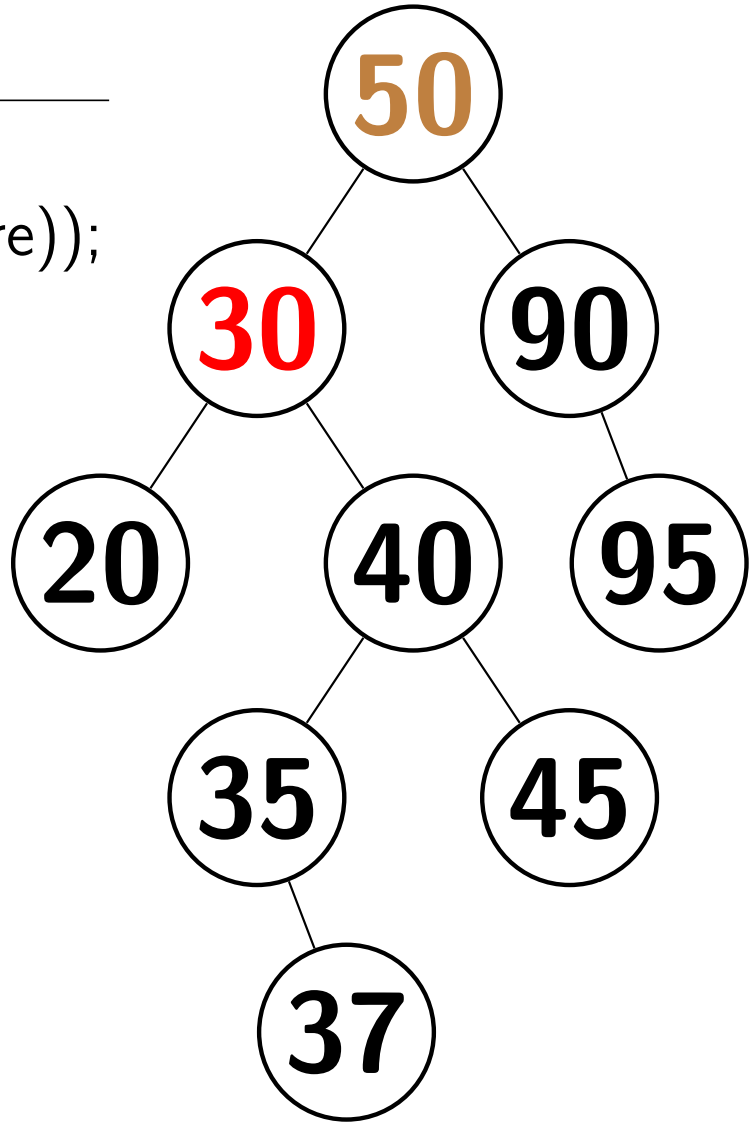


Árvore Binária de Busca - Inserir

Inserir o elemento '47' na ABB:

Arvore* INSERIR (Arvore *a, int v)

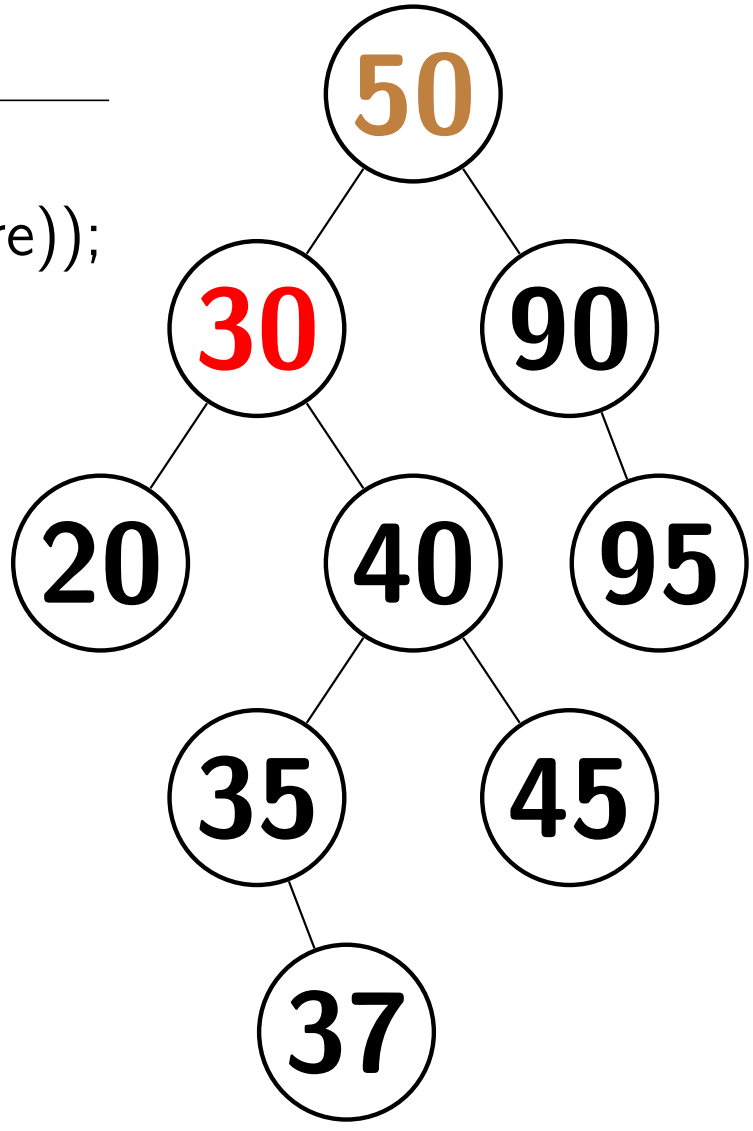
```
if (a == NULL) {
    a = (Arvore*)malloc(sizeof(Arvore));
    a->info = v;
    a->esq = NULL;
    a->dir = NULL;
}
else if (v < a->info) {
    a->esq = INSERIR (a->esq, v);
}
else {
    a->dir = INSERIR (a->dir, v);
}
return a;
```



Árvore Binária de Busca - Inserir

Inserir o elemento '47' na ABB:

```
Arvore* INSERIR (Arvore *a, int v)
if (a == NULL) {
    a = (Arvore*)malloc(sizeof(Arvore));
    a->info = v;
    a->esq = NULL;
    a->dir = NULL;
}
else if (v < a->info) {
    a->esq = INSERIR (a->esq, v);
}
else {
    a->dir = Inserir (a->dir, v);
}
return a;
```

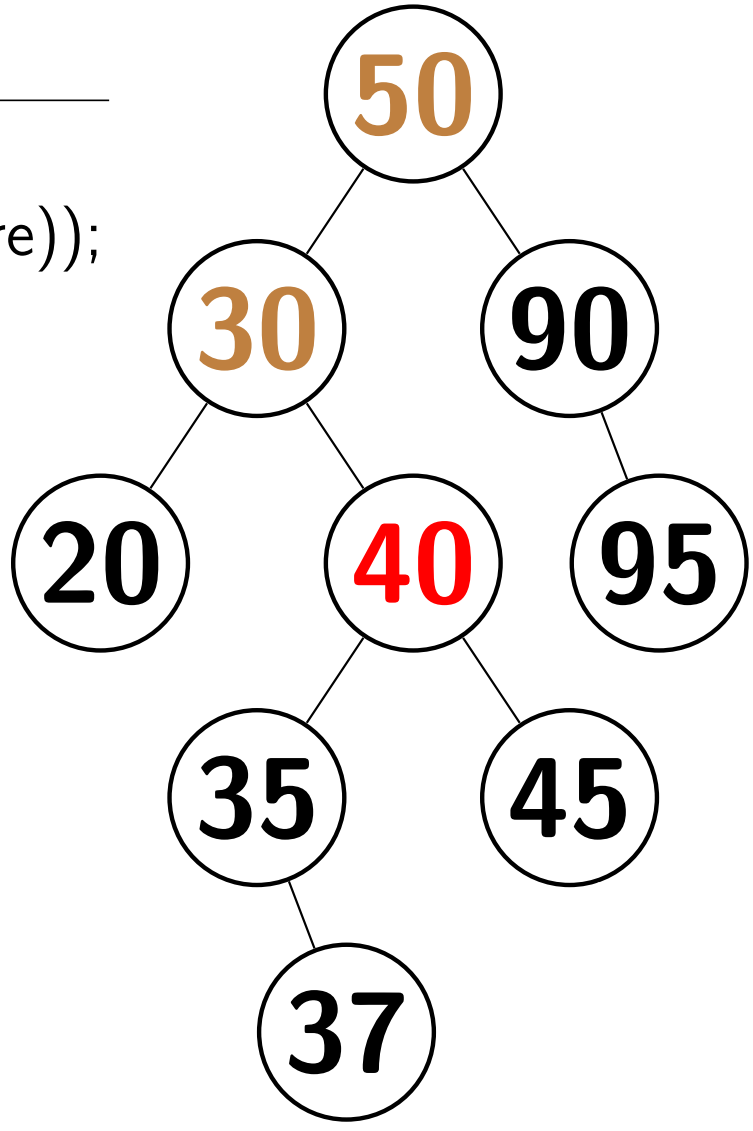


Árvore Binária de Busca - Inserir

Inserir o elemento '47' na ABB:

Arvore* INSERIR (Arvore *a, int v)

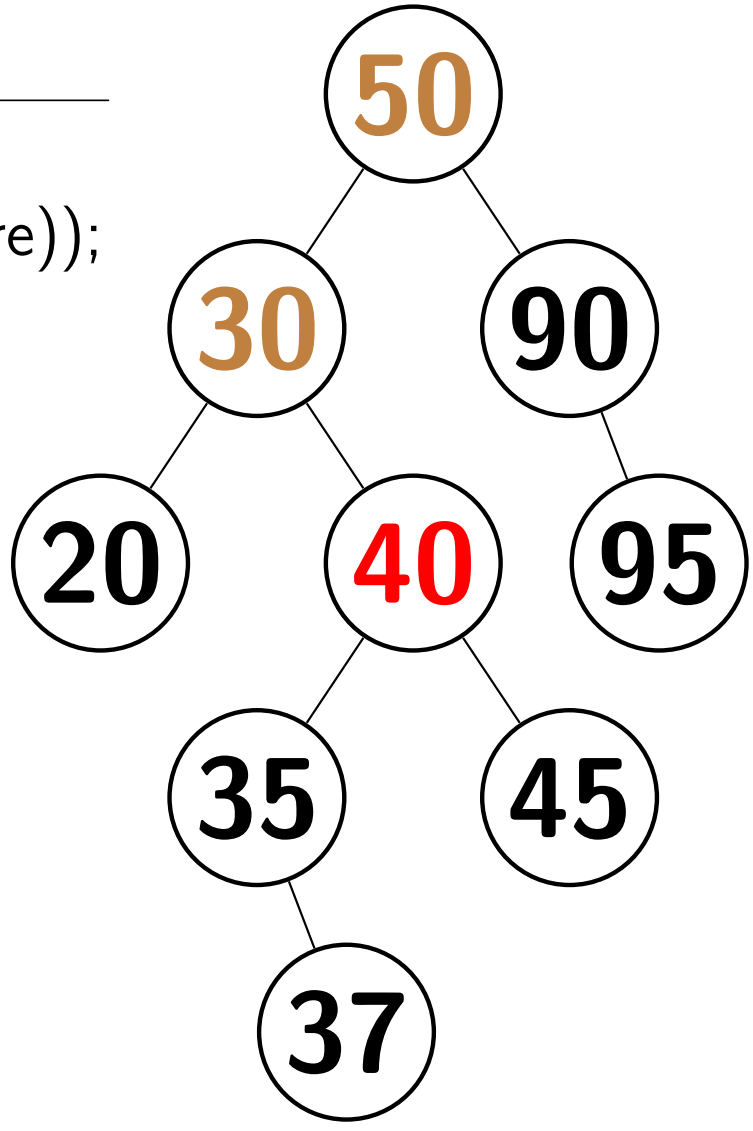
```
if (a == NULL) {
    a = (Arvore*)malloc(sizeof(Arvore));
    a->info = v;
    a->esq = NULL;
    a->dir = NULL;
}
else if (v < a->info) {
    a->esq = INSERIR (a->esq, v);
}
else {
    a->dir = INSERIR (a->dir, v);
}
return a;
```



Árvore Binária de Busca - Inserir

Inserir o elemento '47' na ABB:

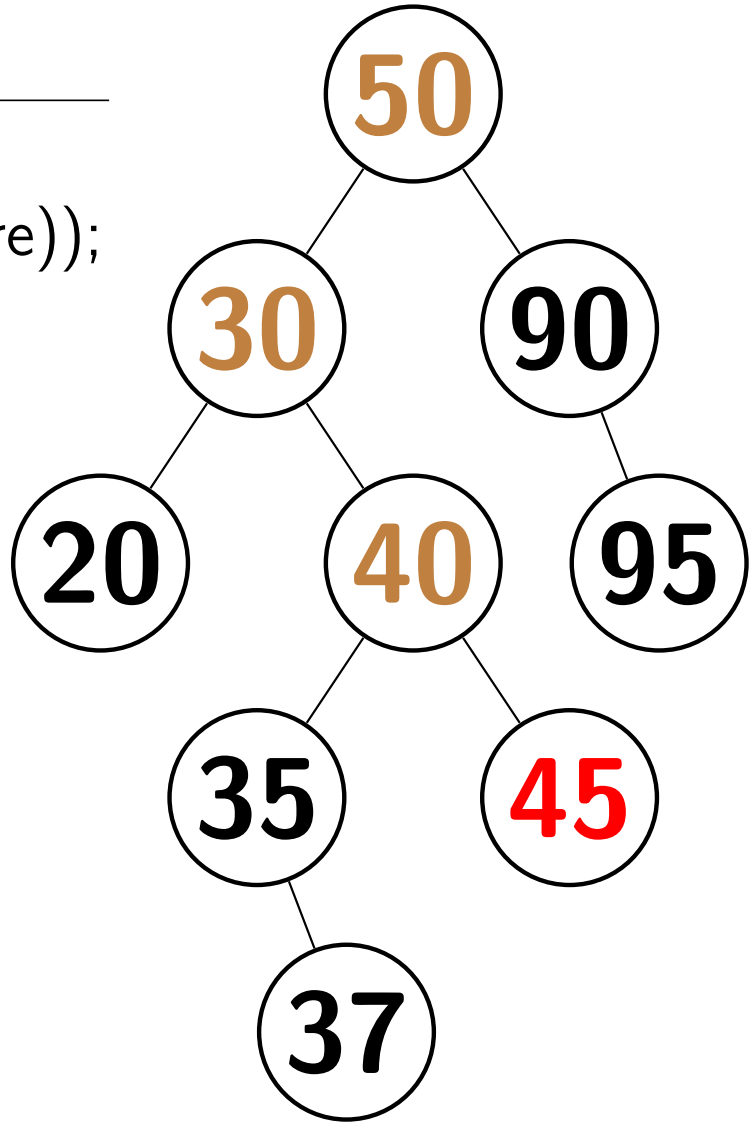
```
Arvore* INSERIR (Arvore *a, int v)
if (a == NULL) {
    a = (Arvore*)malloc(sizeof(Arvore));
    a->info = v;
    a->esq = NULL;
    a->dir = NULL;
}
else if (v < a->info) {
    a->esq = INSERIR (a->esq, v);
}
else {
    a->dir = Inserir (a->dir, v);
}
return a;
```



Árvore Binária de Busca - Inserir

Inserir o elemento '47' na ABB:

```
Arvore* INSERIR (Arvore *a, int v)
if (a == NULL) {
    a = (Arvore*)malloc(sizeof(Arvore));
    a->info = v;
    a->esq = NULL;
    a->dir = NULL;
}
else if (v < a->info) {
    a->esq = INSERIR (a->esq, v);
}
else {
    a->dir = INSERIR (a->dir, v);
}
return a;
```

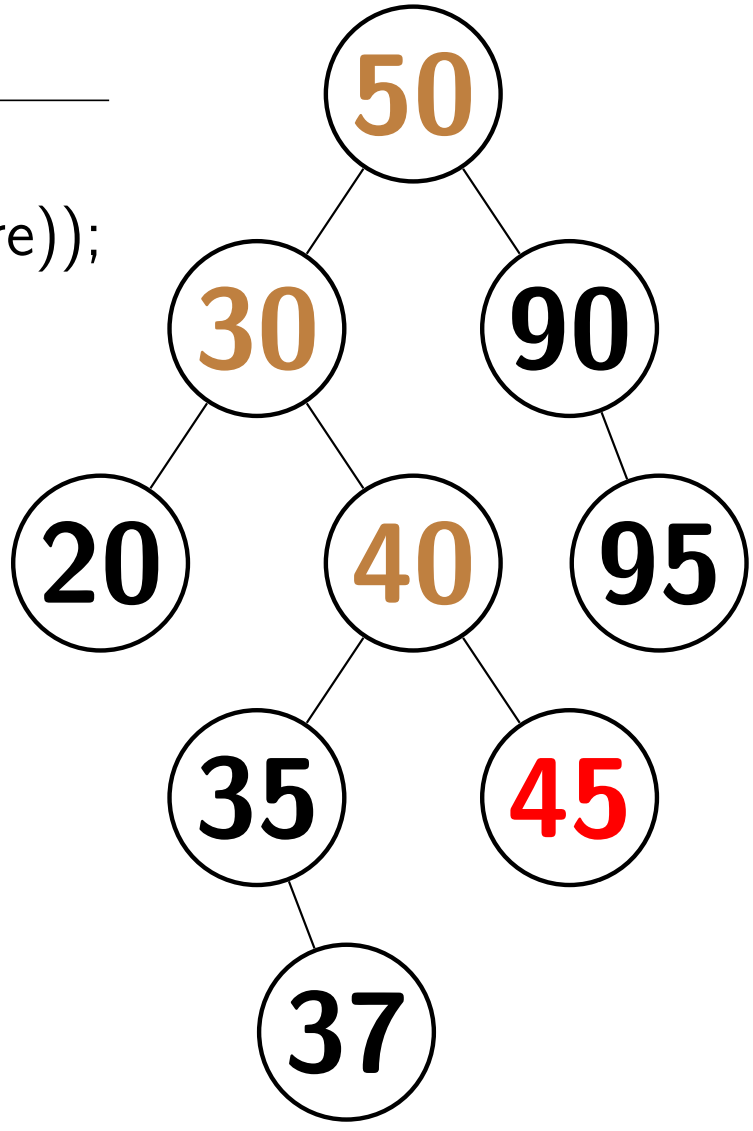


Árvore Binária de Busca - Inserir

Inserir o elemento '47' na ABB:

Arvore* INSERIR (Arvore *a, int v)

```
if (a == NULL) {
    a = (Arvore*)malloc(sizeof(Arvore));
    a->info = v;
    a->esq = NULL;
    a->dir = NULL;
}
else if (v < a->info) {
    a->esq = INSERIR (a->esq, v);
}
else {
    a->dir = Inserir (a->dir, v);
}
return a;
```

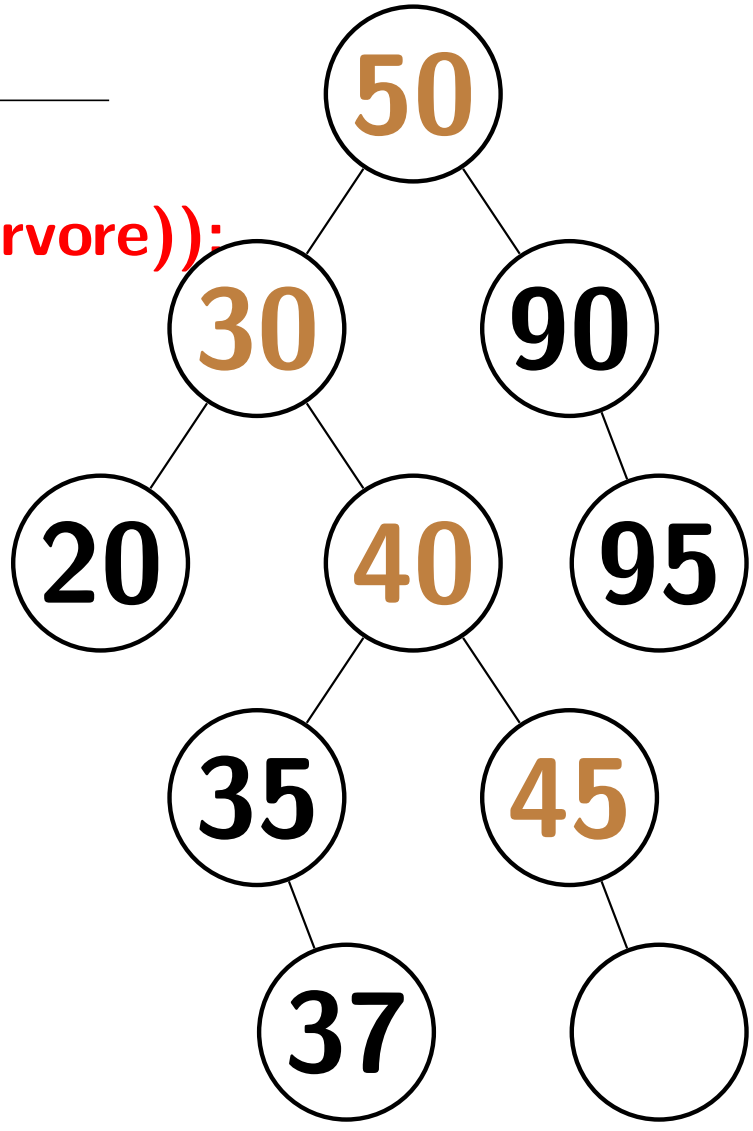


Árvore Binária de Busca - Inserir

Inserir o elemento '**47**' na **ABB**:

Arvore* INSERIR (Arvore *a, int v)

```
if (a == NULL) {
    a = (Arvore*)malloc(sizeof(Arvore));
    a->info = v;
    a->esq = NULL;
    a->dir = NULL;
}
else if (v < a->info) {
    a->esq = INSERIR (a->esq, v);
}
else {
    a->dir = INSERIR (a->dir, v);
}
return a;
```

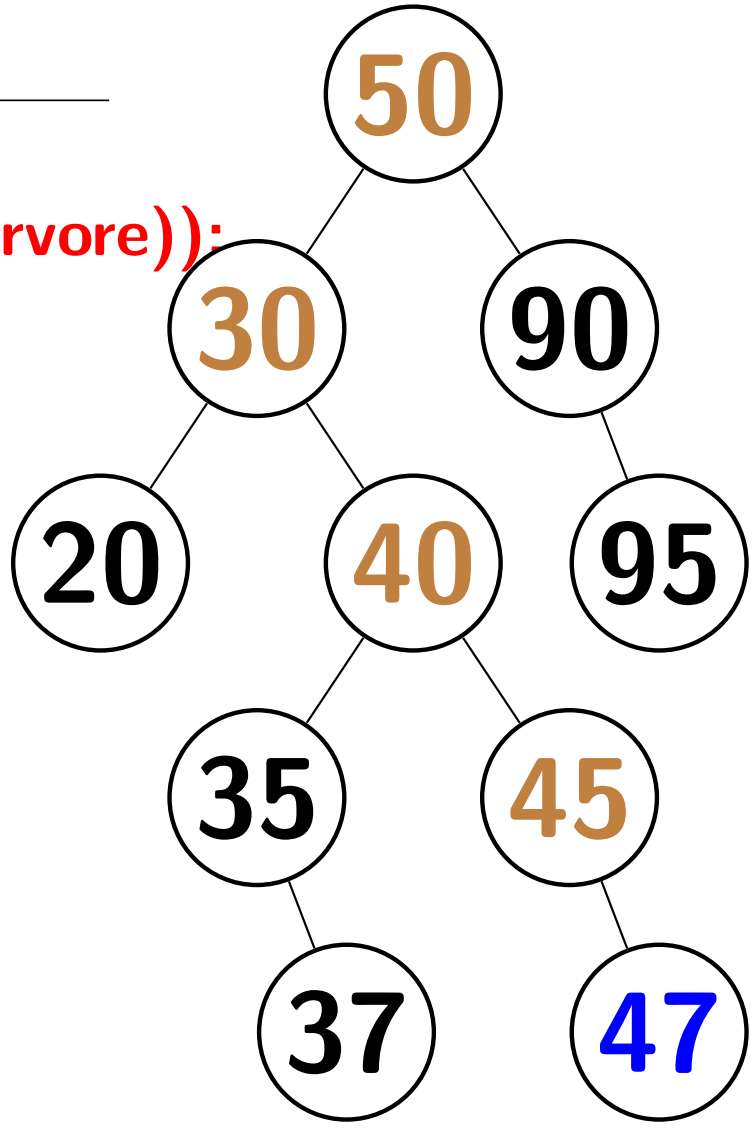


Árvore Binária de Busca - Inserir

Inserir o elemento '**47**' na **ABB**:

Arvore* INSERIR (Arvore *a, int v)

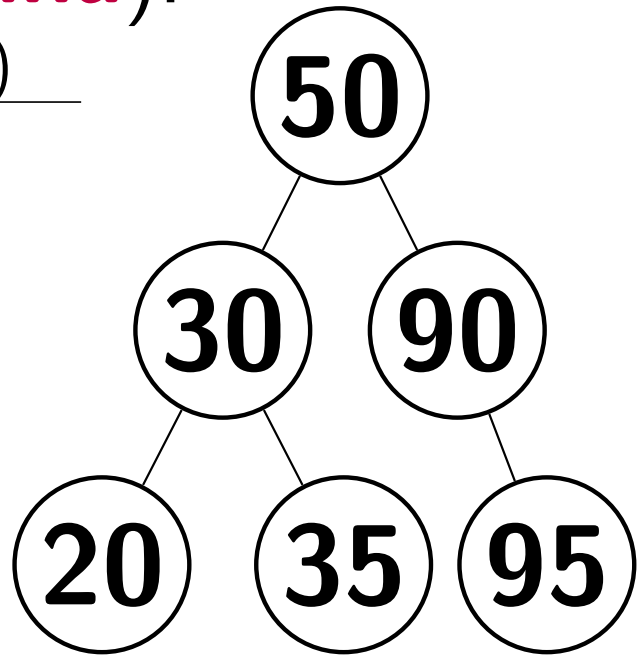
```
if (a == NULL) {
    a = (Arvore*)malloc(sizeof(Arvore));
    a->info = v;
    a->esq = NULL;
    a->dir = NULL;
}
else if (v < a->info) {
    a->esq = INSERIR (a->esq, v);
}
else {
    a->dir = INSERIR (a->dir, v);
}
return a;
```



Sumário

Árvore Binária de Busca - Remover

Remover o elemento '20' (folha):

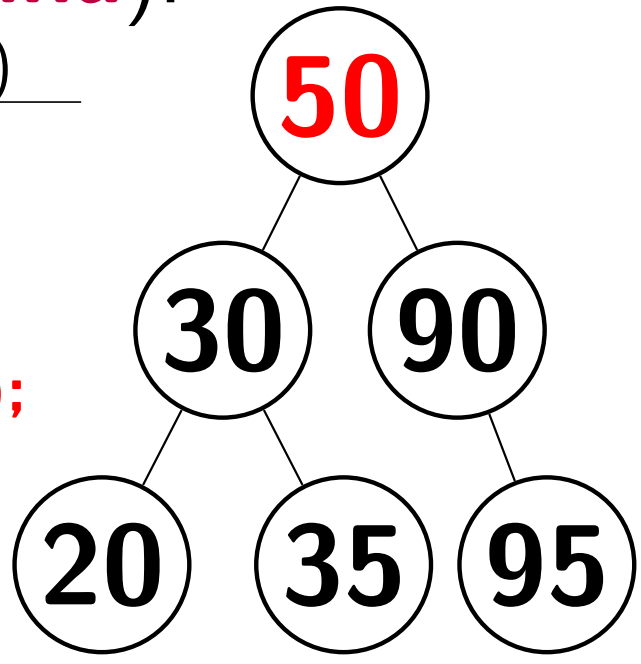


Arvore* REMOVER (Arvore *a, int v)

```
if (a == NULL) { return NULL; }
else {
  if (a->info > v)
    a->esq = REMOVER (a->esq, v);
  else if (a->info < v)
    a->dir = REMOVER (a->dir, v);
  else
    if ((a->esq == NULL) && (a->dir == NULL)) { ... }
    else if (a->dir == NULL) { ... }
    else if (a->esq == NULL) { ... }
    else { ... }
return a;
```

Árvore Binária de Busca - Remover

Remover o elemento '**20**' (**folha**):

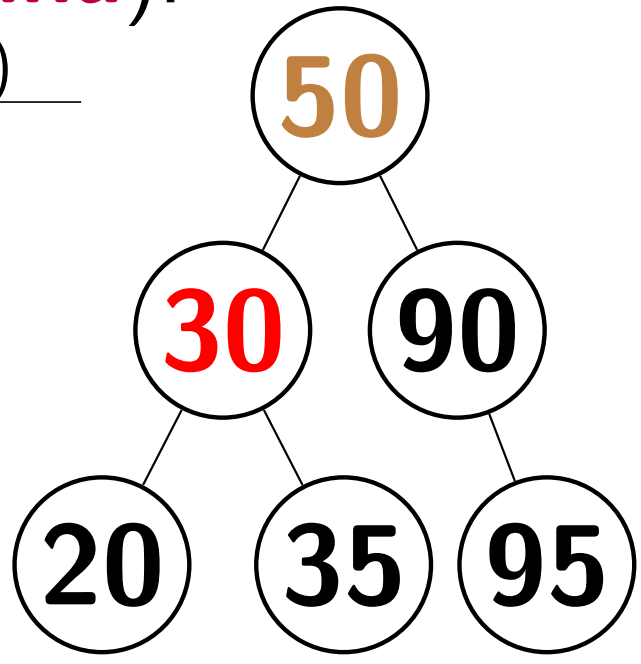


Arvore* REMOVER (Arvore *a, int v)

```
if (a == NULL) { return NULL; }
else {
  if (a->info > v)
    a->esq = Remover (a->esq, v);
  else if (a->info < v)
    a->dir = REMOVER (a->dir, v);
  else
    if ((a->esq == NULL) && (a->dir == NULL)) { ... }
    else if (a->dir == NULL) { ... }
    else if (a->esq == NULL) { ... }
    else { ... }
return a;
```

Árvore Binária de Busca - Remover

Remover o elemento '**20**' (**folha**):

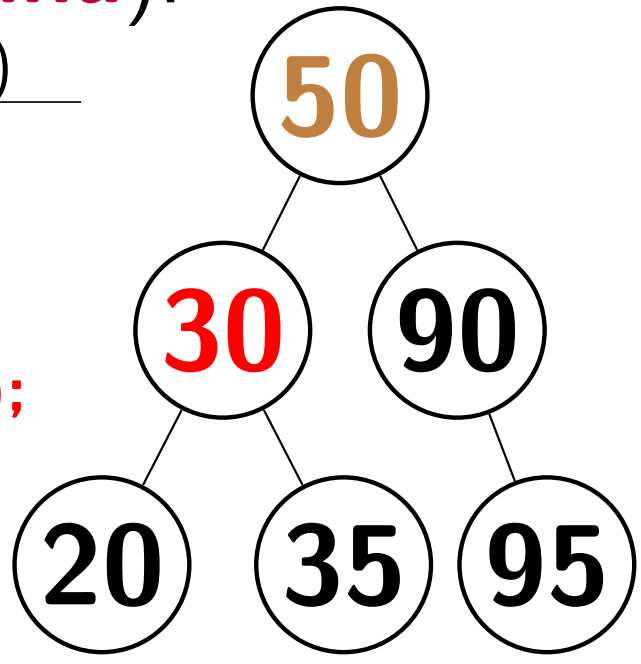


Arvore* REMOVER (Arvore *a, int v)

```
if (a == NULL) { return NULL; }
else {
  if (a->info > v)
    a->esq = REMOVER (a->esq, v);
  else if (a->info < v)
    a->dir = REMOVER (a->dir, v);
  else
    if ((a->esq == NULL) && (a->dir == NULL)) { ... }
    else if (a->dir == NULL) { ... }
    else if (a->esq == NULL) { ... }
    else { ... }
return a;
```

Árvore Binária de Busca - Remover

Remover o elemento '**20**' (**folha**):

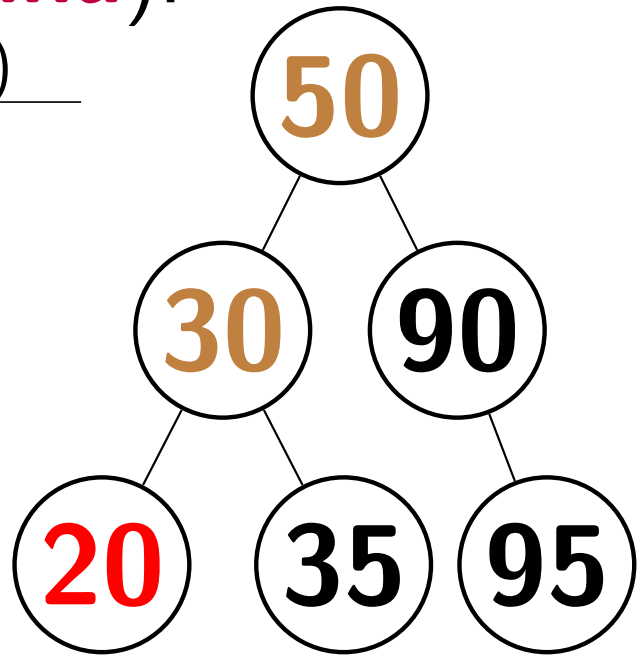


Arvore* REMOVER (Arvore *a, int v)

```
if (a == NULL) { return NULL; }
else {
  if (a->info > v)
    a->esq = Remover (a->esq, v);
  else if (a->info < v)
    a->dir = REMOVER (a->dir, v);
  else
    if ((a->esq == NULL) && (a->dir == NULL)) { ... }
    else if (a->dir == NULL) { ... }
    else if (a->esq == NULL) { ... }
    else { ... }
return a;
```

Árvore Binária de Busca - Remover

Remover o elemento '20' (folha):

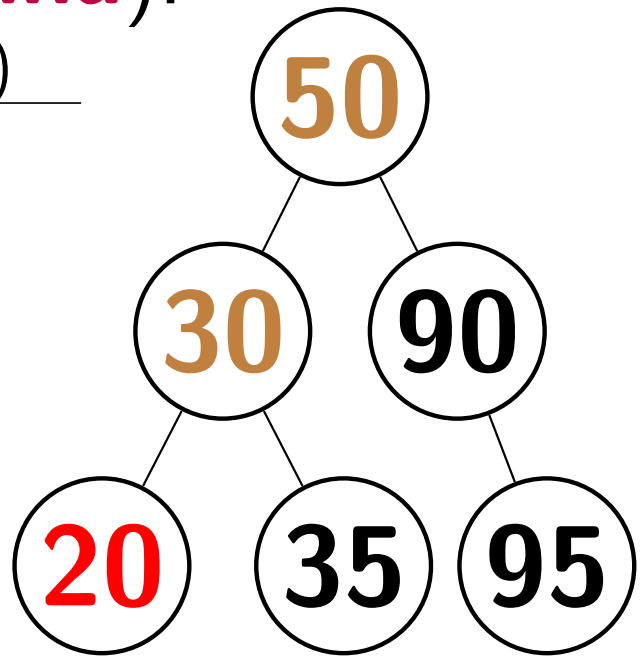


Arvore* REMOVER (Arvore *a, int v)

```
if (a == NULL) { return NULL; }
else {
  if (a->info > v)
    a->esq = REMOVER (a->esq, v);
  else if (a->info < v)
    a->dir = REMOVER (a->dir, v);
  else
    if ((a->esq == NULL) && (a->dir == NULL)) { ... }
    else if (a->dir == NULL) { ... }
    else if (a->esq == NULL) { ... }
    else { ... }
return a;
```

Árvore Binária de Busca - Remover

Remover o elemento '20' (folha):

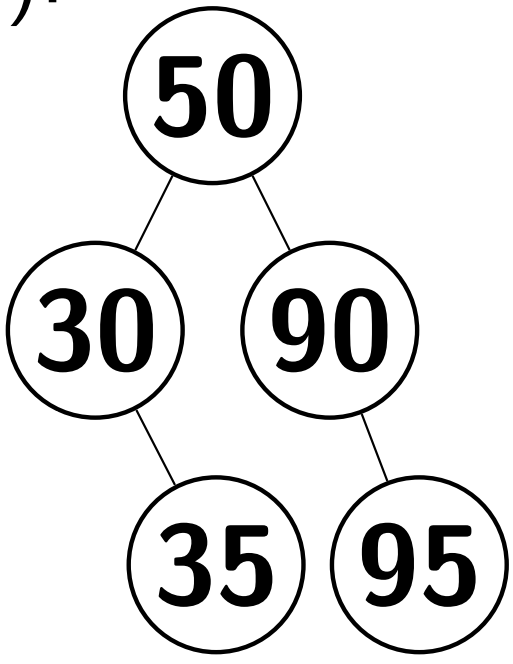


Arvore* REMOVER (Arvore *a, int v)

```
if (a == NULL) { return NULL; }
else {
  if (a->info > v)
    a->esq = REMOVER (a->esq, v);
  else if (a->info < v)
    a->dir = REMOVER (a->dir, v);
  else
    if ((a->esq == NULL) && (a->dir == NULL))
      free (a);
      a = NULL;
    else if (a->dir == NULL) { ... }
    else if (a->esq == NULL) { ... }
    else { ... }
return a;
```

Árvore Binária de Busca - Remover

Remover o elemento '20' (folha):

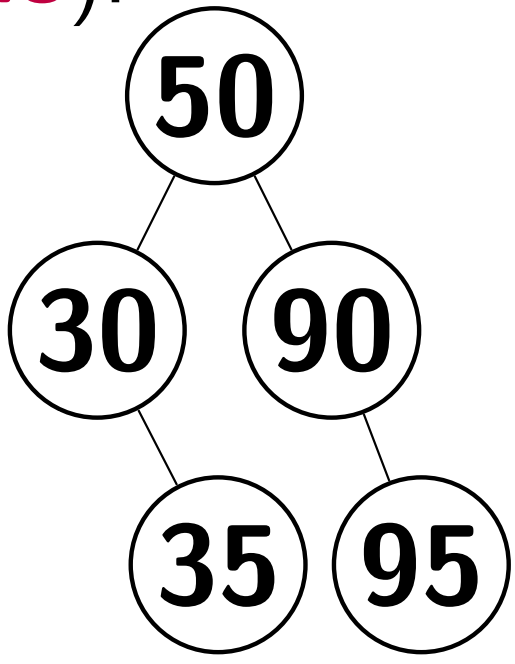


Arvore* REMOVER (Arvore *a, int v)

```
if (a == NULL) { return NULL; }
else {
  if (a->info > v)
    a->esq = REMOVER (a->esq, v);
  else if (a->info < v)
    a->dir = REMOVER (a->dir, v);
  else
    if ((a->esq == NULL) && (a->dir == NULL)) { ... }
    else if (a->dir == NULL) { ... }
    else if (a->esq == NULL) { ... }
    else { ... }
return a;
```

Árvore Binária de Busca - Remover

Remover o elemento '**90**' (**1 filho**):

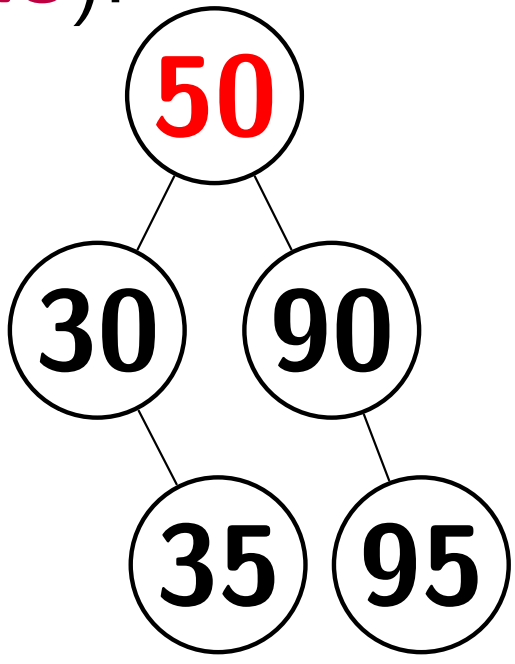


Arvore* REMOVER (Arvore *a, int v)

```
if (a == NULL) { return NULL; }
else {
  if (a->info > v)
    a->esq = REMOVER (a->esq, v);
  else if (a->info < v)
    a->dir = REMOVER (a->dir, v);
  else
    if ((a->esq == NULL) && (a->dir == NULL)) { ... }
    else if (a->dir == NULL) { ... }
    else if (a->esq == NULL) { ... }
    else { ... }
return a;
```

Árvore Binária de Busca - Remover

Remover o elemento '**90**' (**1 filho**):

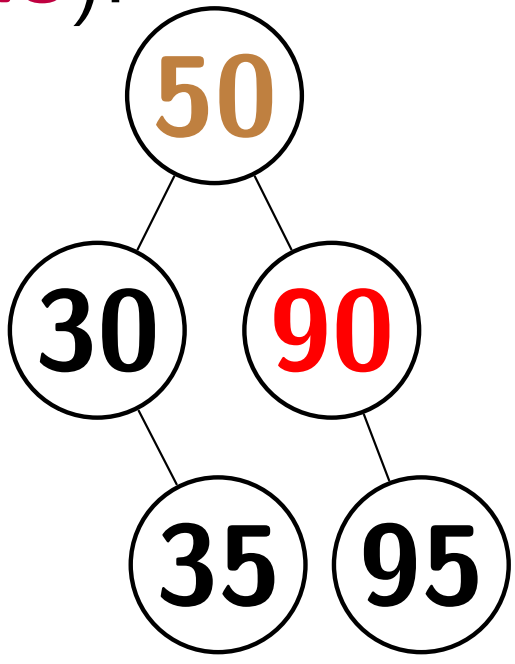


Arvore* REMOVER (Arvore *a, int v)

```
if (a == NULL) { return NULL; }
else {
  if (a->info > v)
    a->esq = REMOVER (a->esq, v);
  else if (a->info < v)
    a->dir = Remover (a->dir, v);
  else
    if ((a->esq == NULL) && (a->dir == NULL)) { ... }
    else if (a->dir == NULL) { ... }
    else if (a->esq == NULL) { ... }
    else { ... }
return a;
```

Árvore Binária de Busca - Remover

Remover o elemento '**90**' (**1 filho**):

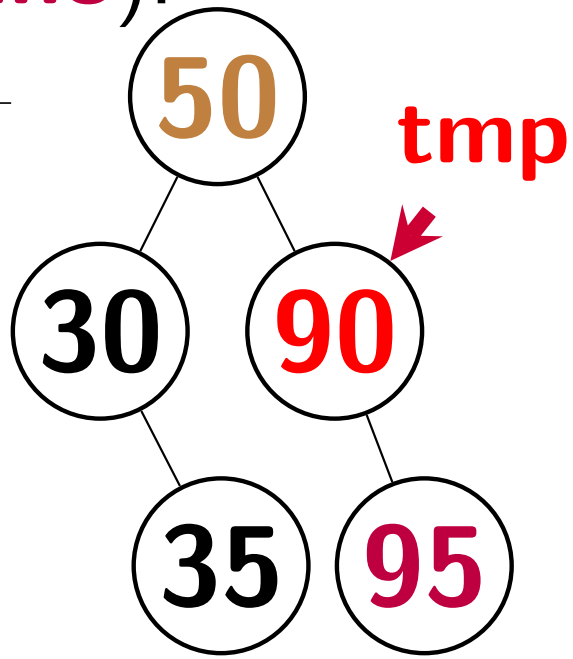


Arvore* REMOVER (Arvore *a, int v)

```
if (a == NULL) { return NULL; }
else {
  if (a->info > v)
    a->esq = REMOVER (a->esq, v);
  else if (a->info < v)
    a->dir = REMOVER (a->dir, v);
  else
    if ((a->esq == NULL) && (a->dir == NULL)) { ... }
    else if (a->dir == NULL) { ... }
    else if (a->esq == NULL) { ... }
    else { ... }
return a;
```

Árvore Binária de Busca - Remover

Remover o elemento '90' (1 filho):

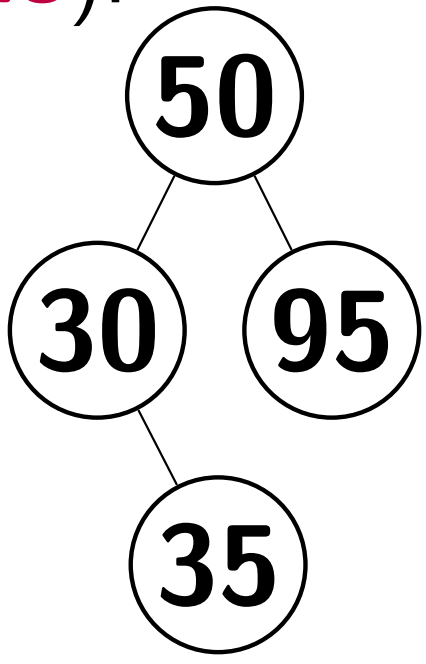


Arvore* REMOVER (Arvore *a, int v)

```
if (a == NULL) { return NULL; }
else {
  if (a->info > v)
    a->esq = REMOVER (a->esq, v);
  else if (a->info < v)
    a->dir = REMOVER (a->dir, v);
  else
    if ((a->esq == NULL) && (a->dir == NULL)) { ... }
    else if (a->dir == NULL) { ... }
    else if (a->esq == NULL)
      Arvore *tmp = a;
      a = a->dir;
      free (tmp);
    else { ... }
```

Árvore Binária de Busca - Remover

Remover o elemento '**90**' (**1 filho**):

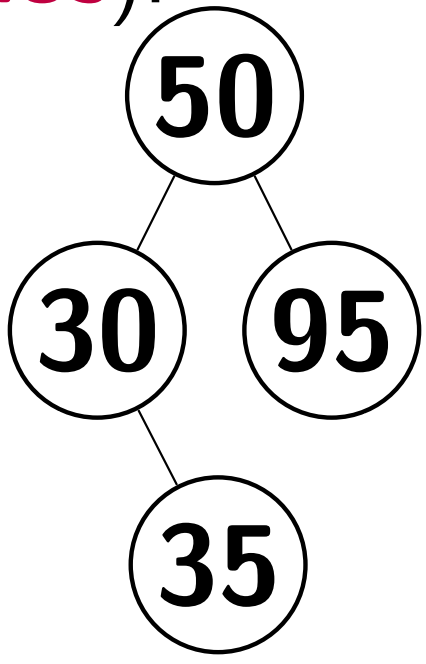


Arvore* REMOVER (Arvore *a, int v)

```
if (a == NULL) { return NULL; }
else {
  if (a->info > v)
    a->esq = REMOVER (a->esq, v);
  else if (a->info < v)
    a->dir = REMOVER (a->dir, v);
  else
    if ((a->esq == NULL) && (a->dir == NULL)) { ... }
    else if (a->dir == NULL) { ... }
    else if (a->esq == NULL) { ... }
    else { ... }
return a;
```

Árvore Binária de Busca - Remover

Remover o elemento '**50**' (**2 filhos**):

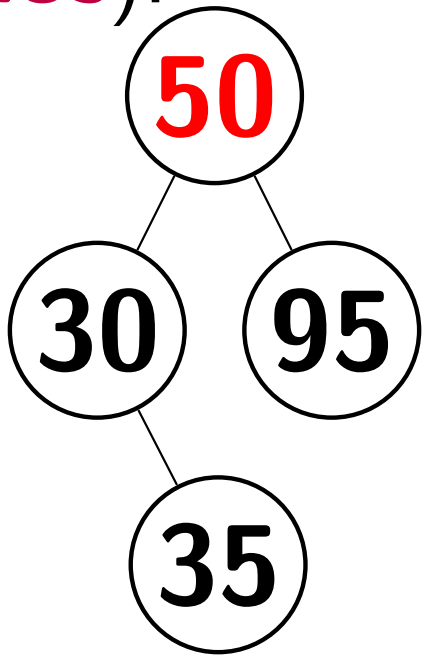


Arvore* REMOVER (Arvore *a, int v)

```
if (a == NULL) { return NULL; }
else {
  if (a->info > v)
    a->esq = REMOVER (a->esq, v);
  else if (a->info < v)
    a->dir = REMOVER (a->dir, v);
  else
    if ((a->esq == NULL) && (a->dir == NULL)) { ... }
    else if (a->dir == NULL) { ... }
    else if (a->esq == NULL) { ... }
    else { ... }
return a;
```

Árvore Binária de Busca - Remover

Remover o elemento '**50**' (**2 filhos**):



Arvore* REMOVER (Arvore *a, int v)

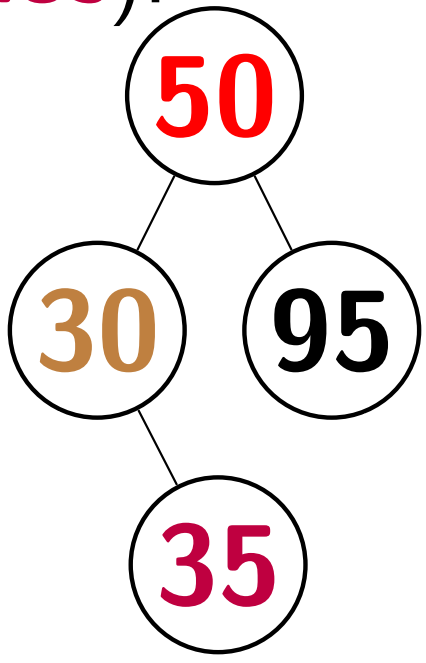
```
if (a == NULL) { return NULL; }
else {
  if (a->info > v)
    a->esq = REMOVER (a->esq, v);
  else if (a->info < v)
    a->dir = REMOVER (a->dir, v);
  else
    if ((a->esq == NULL) && (a->dir == NULL)) { ... }
    else if (a->dir == NULL) { ... }
    else if (a->esq == NULL) { ... }
    else { ... }
return a;
```

Árvore Binária de Busca - Remover

Remover o elemento '50' (2 filhos):

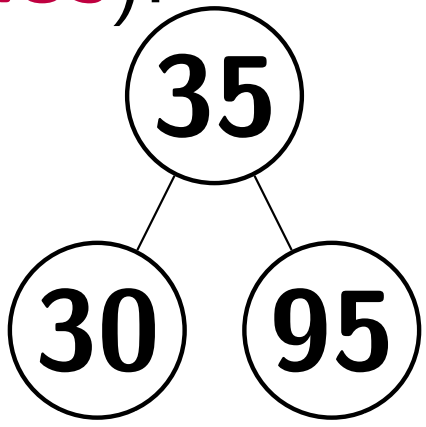
Arvore* REMOVER (Arvore *a, int v)

```
if (a == NULL) { return NULL; }
else {
  if (a->info > v)
    a->esq = REMOVER (a->esq, v);
  else if (a->info < v)
    a->dir = REMOVER (a->dir, v);
  else
      Arvore *tmp = a->esq;
      while (tmp->dir != NULL)
        tmp = tmp->dir;
      a->info = tmp->info;
      tmp->info = v;
      a->esq = remover (a->esq, v);
return a;
```



Árvore Binária de Busca - Remover

Remover o elemento '**50**' (**2 filhos**):



Arvore* REMOVER (Arvore *a, int v)

```
if (a == NULL) { return NULL; }
else {
  if (a->info > v)
    a->esq = REMOVER (a->esq, v);
  else if (a->info < v)
    a->dir = REMOVER (a->dir, v);
  else
    if ((a->esq == NULL) && (a->dir == NULL)) { ... }
    else if (a->dir == NULL) { ... }
    else if (a->esq == NULL) { ... }
    else { ... }
return a;
```

Sumário

Complexidade

Operação	Média	Pior Caso
Pesquisar	$\mathcal{O}(\log n)$	$\mathcal{O}(n)$
Inserir	$\mathcal{O}(\log n)$	$\mathcal{O}(n)$
Remover	$\mathcal{O}(\log n)$	$\mathcal{O}(n)$