INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO PIAUÍ

CAMPUS TERESINA-CENTRAL

DIRETORIA DE ENSINO

Estrutura de Dados II –

Implementação Percurso Pos-*Ordem* em Árvore Alinhada

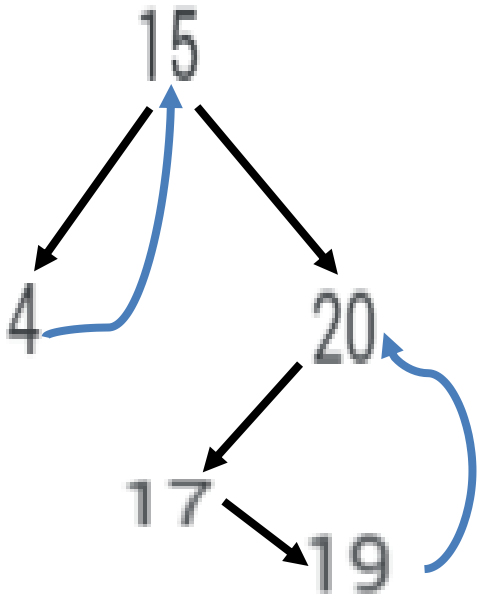
Estratégias de Percurso

- Aula 7

Professora: Elanne Cristina O. dos Santos

elannecristina.santos@gmail.com
elannecristina.santos@ifpi.edu.br

Árvores Alinhadas
Sobre percurso *pos-ordem (LRV)*



Não se dá tão trivial quanto o in –ordem porque é necessário ter uma estrutura que guarde o valor do nó antes de avançar na profundidade da árvore.

Ex.:

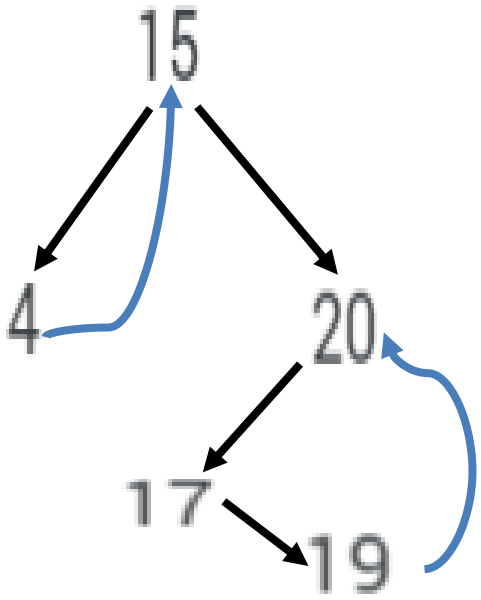
No caso do nó 17.. Ao seguir a direita para 19 é necessário guardar o valor 17 em alguma estrutura para realizar sua posterior impressão.

SUGESTÃO: uma pilha ????

Estaríamos novamente trabalhando com pilhas.

NÓS VISITADOS: 4 19 17 20 15

Árvores Alinhadas
Sobre percurso *pos-ordem (LRV)*



No caso do nó 4.. ao encerrar ele segue o sucessor e volta para o nó 15. É necessário ter uma marcação que indique que 15 já visitou a esquerda e que agora será feita o lado direito.

O controle de marcações no intuito de saber qual dos lados visitar também torna o algoritmo mais complexo.

O mesmo problema vai acontecer com o 20. Quando ele voltar ao 20 pelo sucessor é necessário ter controle que o lado esquerdo de 20 já foi visitado.

NÓS VISITADOS: 4 19 17 20 15

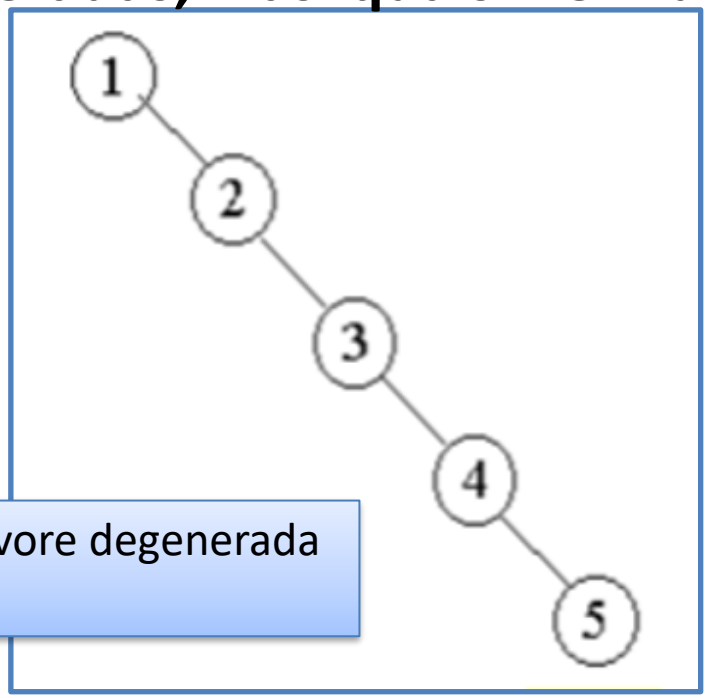
Sobre as implementações

- O autor afirma que:

- “Os percursos recursivos baseiam-se na pilha em tempo de execução, que pode transbordar com árvores muito profundas.”
- “Os percursos iterativos também usam uma pilha em tempo de execução. Podem apresentar um desempenho um pouco melhor no caso de árvores muito profundas.”

Sobre as implementações

- O autor afirma que:
- “O algoritmo de Morris percorre com sucesso a árvore, mas somente uma vez, pois destrói sua estrutura original.”
- **“Morris se baseia no fato de que o percurso in-order é muito simples para árvores degeneradas, nas quais nenhum nó tem filhos à esquerda.” Ex.:**



Exemplo de árvore degenerada ou assimétrica

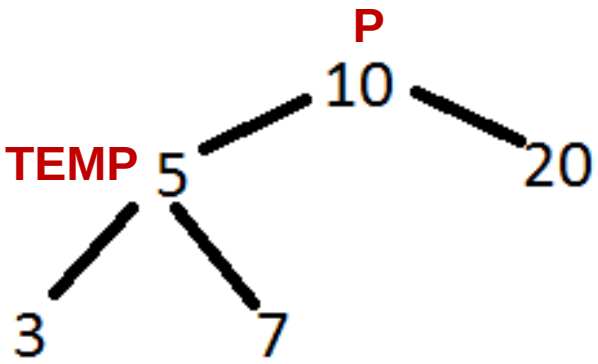
Sobre as implementações

- O autor afirma que:

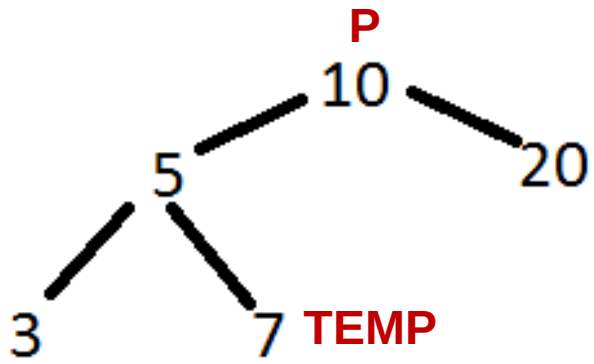
- “Numa árvore degenerada, as 3 etapas usuais LVR para cada nó no percurso ***in-ordem*** se transforma em VR.”
- “Baseado nessa estratégia o algoritmo de MORRIS modifica a árvore para realizar o percurso”.
- “Percorre com sucesso **somente uma vez**”.
- “**Destrói a estrutura original**.”

Morris Percurso inOrder (LVR)

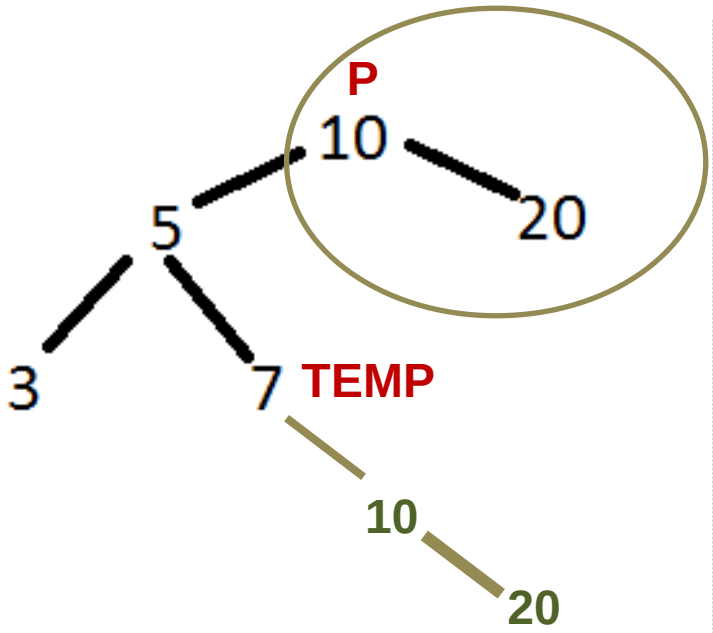
```
void MorrisInorder() {
    ArvoreNo<T> *p=root, *tmp;
    while (p!=0){
        if (p->left==0){
            visit(p);
            p=p->right;
        }
        else {
            tmp = p->left;
            while (tmp->right!=0 && tmp->right!=p)
                tmp=tmp->right;
            if (tmp->right==0){
                tmp->right=p;
                p=p->left;
            }
            else{
                visit(p);
                tmp->right=0;
                p=p->right;
            }
        }
    }//fim while
}
```



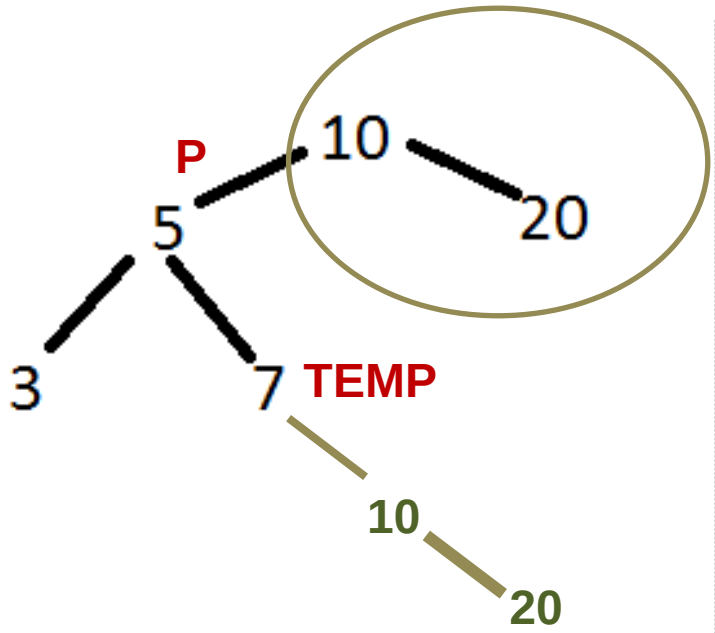

```
void MorrisInorder() {
    ArvoreNo<T> *p=root, *tmp;
    while (p!=0){
        if (p->left==0){
            visit(p);
            p=p->right;
        }
        else {
            tmp = p->left;
            while (tmp->right!=0 && tmp->right!=p)
                tmp=tmp->right;
            if (tmp->right==0){
                tmp->right=p;
                p=p->left;
            }
            else{
                visit(p);
                tmp->right=0;
                p=p->right;
            }
        }
    }//fim while
}
```



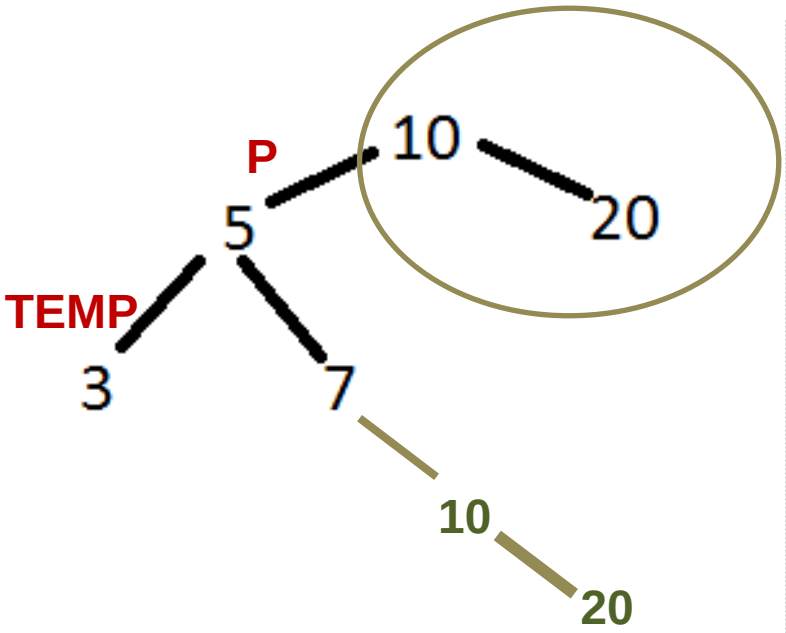

```
void MorrisInorder() {
    ArvoreNo<T> *p=root, *tmp;
    while (p!=0){
        if (p->left==0){
            visit(p);
            p=p->right;
        }
        else {
            tmp = p->left;
            while (tmp->right!=0 && tmp->right!=p)
                tmp=tmp->right;
            if (tmp->right==0){
                tmp->right=p;
                p=p->left;
            }
            else{
                visit(p);
                tmp->right=0;
                p=p->right;
            }
        }
    }//fim while
}
```



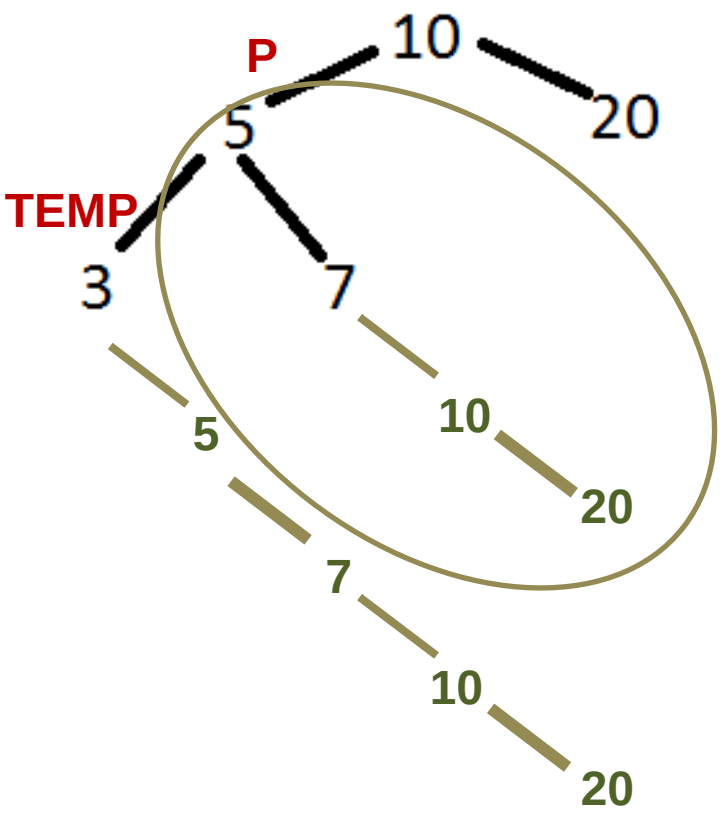

```
void MorrisInorder() {
    ArvoreNo<T> *p=root, *tmp;
    while (p!=0){
        if (p->left==0){
            visit(p);
            p=p->right;
        }
        else {
            tmp = p->left;
            while (tmp->right!=0 && tmp->right!=p)
                tmp=tmp->right;
            if (tmp->right==0){
                tmp->right=p;
                p=p->left;
            }
            else{
                visit(p);
                tmp->right=0;
                p=p->right;
            }
        }
    }//fim while
}
```



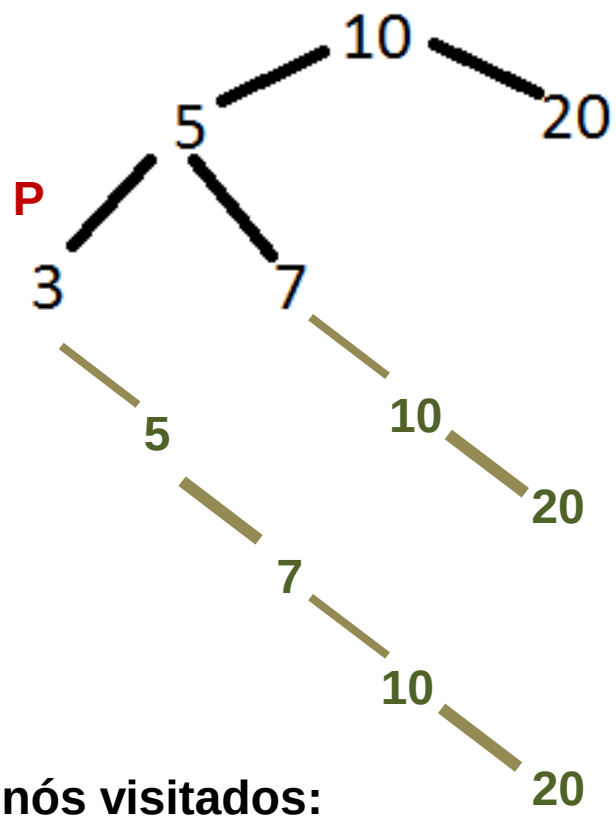

```
void MorrisInorder() {
    ArvoreNo<T> *p=root, *tmp;
    while (p!=0){
        if (p->left==0){
            visit(p);
            p=p->right;
        }
        else {
            tmp = p->left;
            while (tmp->right!=0 && tmp->right!=p)
                tmp=tmp->right;
            if (tmp->right==0){
                tmp->right=p;
                p=p->left;
            }
            else{
                visit(p);
                tmp->right=0;
                p=p->right;
            }
        }
    }//fim while
}
```




```
void MorrisInorder() {
    ArvoreNo<T> *p=root, *tmp;
    while (p!=0){
        if (p->left==0){
            visit(p);
            p=p->right;
        }
        else {
            tmp = p->left;
            while (tmp->right!=0 && tmp->right!=p)
                tmp=tmp->right;
            if (tmp->right==0){
                tmp->right=p;
                p=p->left;
            }
            else{
                visit(p);
                tmp->right=0;
                p=p->right;
            }
        }
    }//fim while
}
```

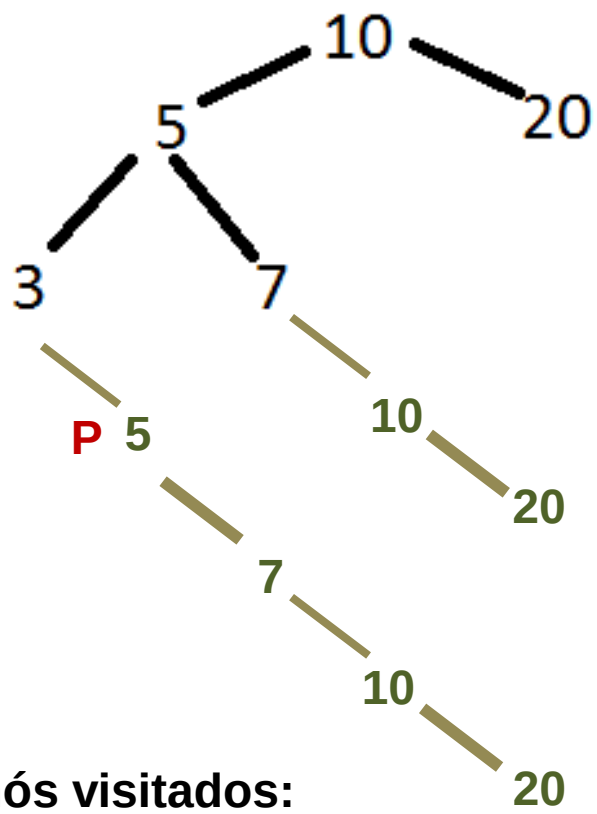



```
void MorrisInorder() {
    ArvoreNo<T> *p=root, *tmp;
    while (p!=0){
        if (p->left==0){
            visit(p);
            p=p->right;
        }
        else {
            tmp = p->left;
            while (tmp->right!=0 && tmp->right!=p)
                tmp=tmp->right;
            if (tmp->right==0){
                tmp->right=p;
                p=p->left;
            }
            else{
                visit(p);
                tmp->right=0;
                p=p->right;
            }
        }
    }//fim while
}
```



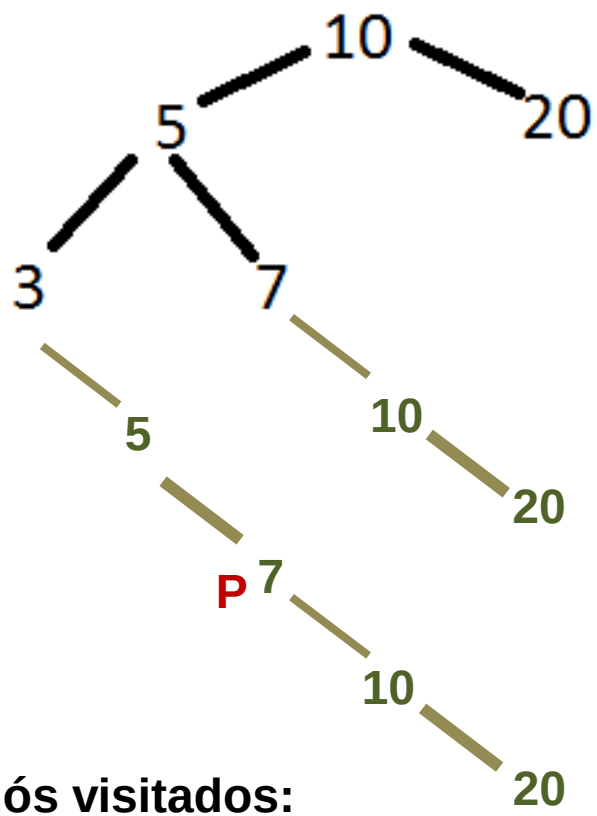
nós visitados:
3

```
void MorrisInorder() {
    ArvoreNo<T> *p=root, *tmp;
    while (p!=0){
        if (p->left==0){
            visit(p);
            p=p->right;
        }
        else {
            tmp = p->left;
            while (tmp->right!=0 && tmp->right!=p)
                tmp=tmp->right;
            if (tmp->right==0){
                tmp->right=p;
                p=p->left;
            }
            else{
                visit(p);
                tmp->right=0;
                p=p->right;
            }
        }
    }//fim while
}
```



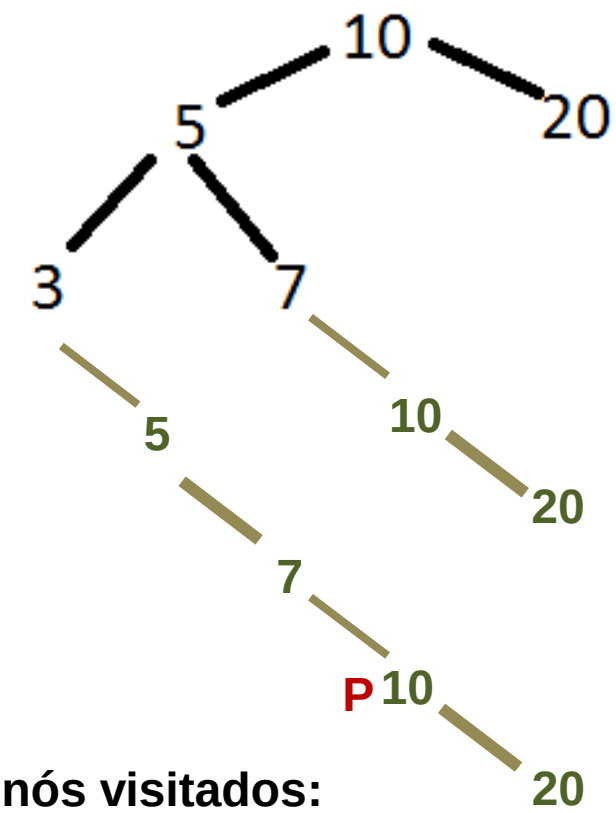
nós visitados:
3 5

```
void MorrisInorder() {
    ArvoreNo<T> *p=root, *tmp;
    while (p!=0){
        if (p->left==0){
            visit(p);
            p=p->right;
        }
        else {
            tmp = p->left;
            while (tmp->right!=0 && tmp->right!=p)
                tmp=tmp->right;
            if (tmp->right==0){
                tmp->right=p;
                p=p->left;
            }
            else{
                visit(p);
                tmp->right=0;
                p=p->right;
            }
        }
    }//fim while
}
```



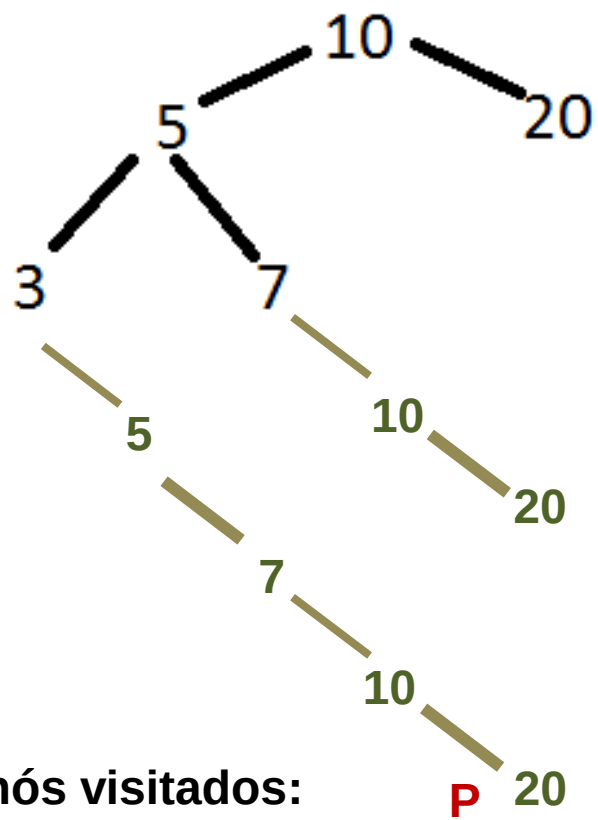
nós visitados:
3 5 7

```
void MorrisInorder() {
    ArvoreNo<T> *p=root, *tmp;
    while (p!=0){
        if (p->left==0){
            visit(p);
            p=p->right;
        }
        else {
            tmp = p->left;
            while (tmp->right!=0 && tmp->right!=p)
                tmp=tmp->right;
            if (tmp->right==0){
                tmp->right=p;
                p=p->left;
            }
            else{
                visit(p);
                tmp->right=0;
                p=p->right;
            }
        }
    }//fim while
}
```



nós visitados:
3 5 7 10

```
void MorrisInorder() {
    ArvoreNo<T> *p=root, *tmp;
    while (p!=0){
        if (p->left==0){
            visit(p);
            p=p->right;
        }
        else {
            tmp = p->left;
            while (tmp->right!=0 && tmp->right!=p)
                tmp=tmp->right;
            if (tmp->right==0){
                tmp->right=p;
                p=p->left;
            }
            else{
                visit(p);
                tmp->right=0;
                p=p->right;
            }
        }
    }//fim while
}
```



nós visitados:
3 5 7 10 20

```
void MorrisInorder() {
    ArvoreNo<T> *p=root, *tmp;
    while (p!=0){
        if (p->left==0){
            visit(p);
            p=p->right;
        }
        else {
            tmp = p->left;
            while (tmp->right!=0 && tmp->right!=p)
                tmp=tmp->right;
            if (tmp->right==0){
                tmp->right=p;
                p=p->left;
            }
            else{
                visit(p);
                tmp->right=0;
                p=p->right;
            }
        }
    }//fim while
}
```

```
void MorrisPreorder(){
    ArvoreNo<T> *p=root, *tmp;
    while (p!=0){
        if (p->left==0){
            visit(p);
            p=p->right;
        }
                tmp->right=p;
                p=p->left;
            }
            else{
                tmp->right=0;
                p=p->right;
            } /*fim else*/  }
    } /*fim while*/ }
```

Morris Percurso PreOrder (VLR)

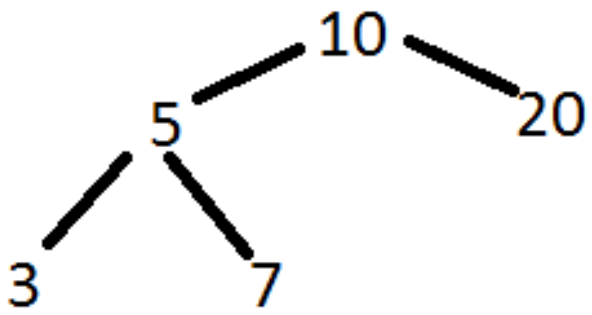
"...fácil de obter, a partir do percurso in order, movendo o "visit" da cláusula "else" mais interna para a cláusula "if" mais interna." (pg. 207)

```
void MorrisPreorder(){
    ArvoreNo<T> *p=root, *tmp;
    while (p!=0){
        if (p->left==0){
            visit(p);
            p=p->right;
        }
        else{
            tmp = p->left;
            while (tmp->right!=0 && tmp->right!=p)
                tmp=tmp->right;
                if (tmp->right==0){
                    visit(p);
                    tmp->right=p;
                    p=p->left;
                }
                else{
                    tmp->right=0;
                    p=p->right;
                } /*fim else*/  }
        } /*fim while*/ }
```

Morris Percurso PreOrder (VLR)

O algoritmo está correto??

Morris Percurso PreOrder (VLR)



Qual o resultado do percurso na árvore?

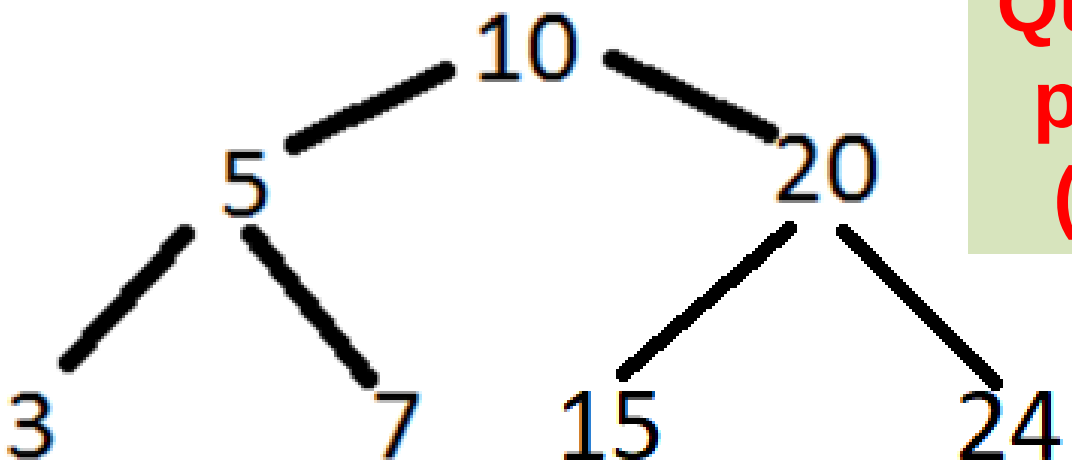
10 5 3 7 20

Sobre as implementações

- O autor afirma que:

- “As árvores alinhadas usam nós maiores, mas não chega a ser um problema. Mas tanto a implementação iterativa quanto a alinhada e a implementação de MORRIS são **MUITO MENOS intuitiva** do que as recursivas. Em consequência, a clareza da implementação e os tempos de execução comparáveis **claramente favorecem, na maioria das situações, as implementações recursivas. “**

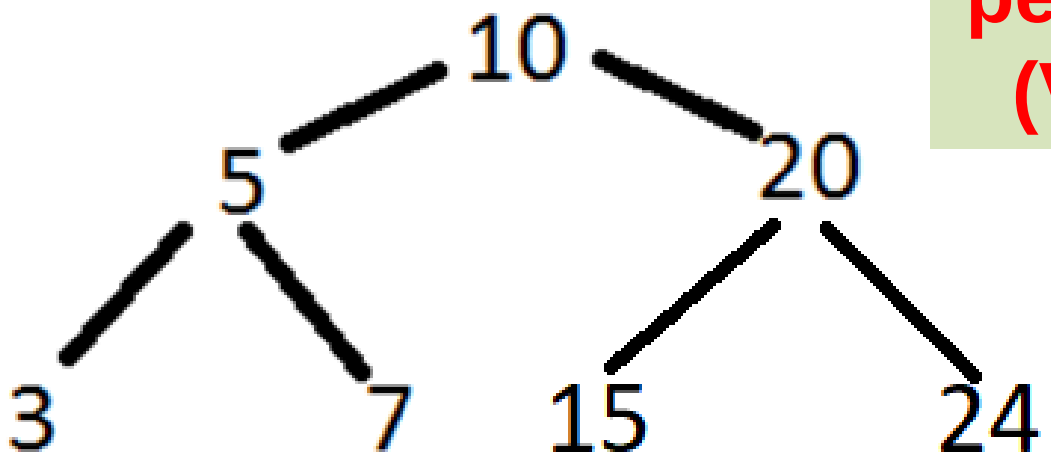
Atividade



Qual o resultado do percurso In-Order (LVR) na árvore?

- Explique quais os passos realizados pelo algoritmo de Morris – Percurso **inORDER**:

Atividade



Qual o resultado do percurso Pré-Order (VLR) na árvore?

- Explique quais os passos realizados pelo algoritmo de Morris – Percurso pre-ORDER: