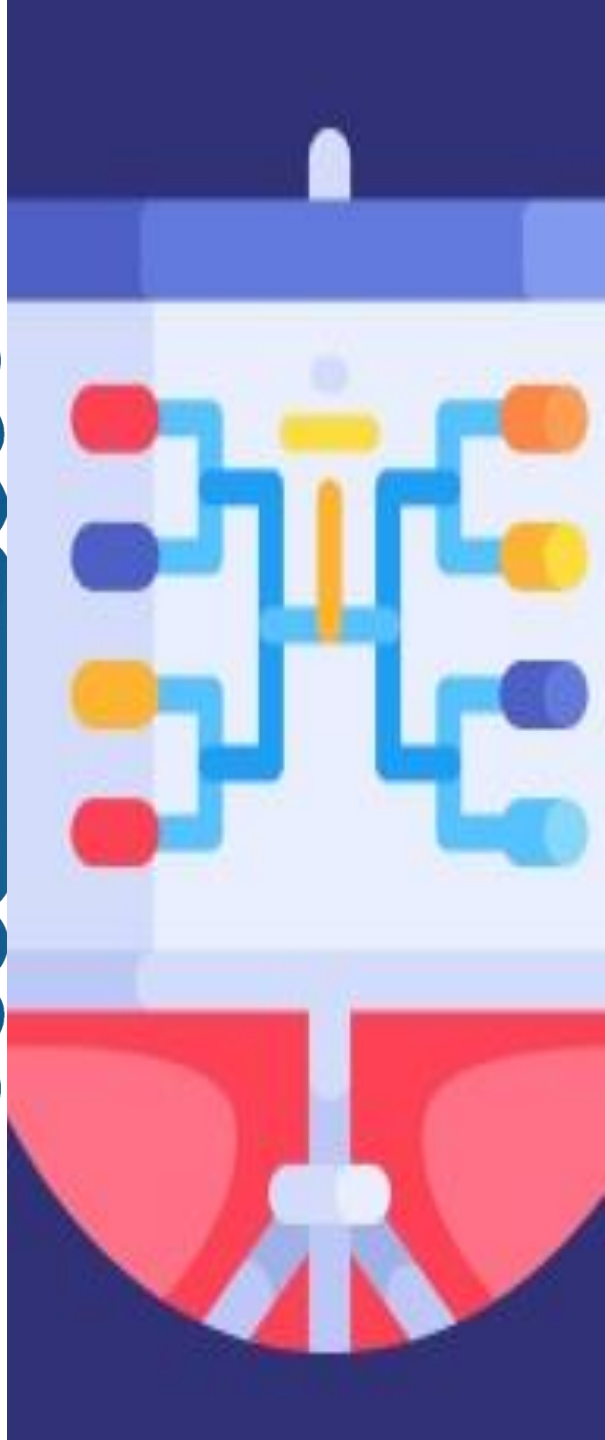
Unicesumar

Estrutura de Dados II



Unidade 4 – Caminhamento em Árvores Binárias

Prof. Sandro T. Pinto

Uma árvore binária de busca é uma estrutura de dados de árvore binária baseada em nós, onde todos os nós da sub árvore esquerda possuem um valor numérico inferior ao nó raiz e todos os nós da sub árvore direita possuem um valor superior ao nó raiz.

Wikipédia (2021)

Unidade 4

Caminhamento em Árvores Binárias

Em algum momento, dependendo dos requisitos de uma aplicação, pode ser necessário percorrer todos os elementos de uma árvore para, por exemplo, exibir todo o seu conteúdo ao usuário.

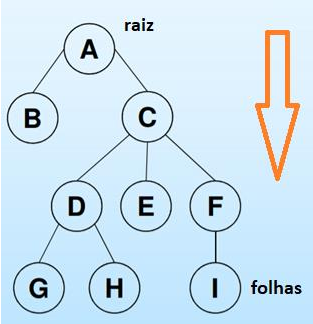
De acordo com a ordem de visitação dos nós, o usuário pode ter visões distintas de uma mesma árvore.

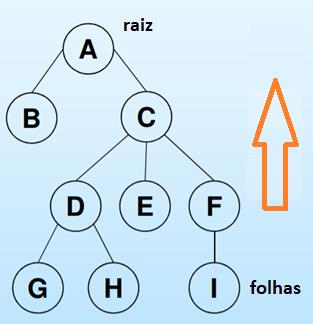
Unidade 4

Caminhamento em Árvores Binárias

Imagine que, para percorrer uma árvore, tomemos o nó raiz como nó inicial e, a partir dele, comecemos a visitar todos os nós adjacentes a ele para, só então, começar a investigar todos os nós da árvore.

Por outro lado, imagine que tomemos um nó folha como ponto de partida e caminhemos em direção à raiz, visitando apenas o ramo da árvore que leva o nó folha à raiz.





Unidade 4

Percurso Pré-Ordem

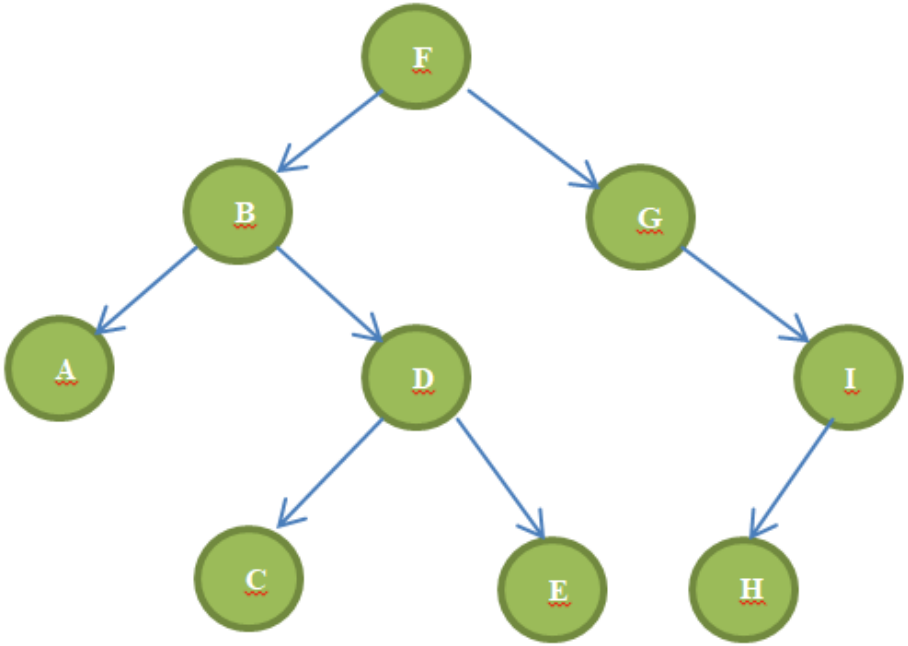
O caminhamento pré-ordem conhecido por caminhamento pré-fixado, marca primeiramente a raiz como visitada, e só depois visitamos as sub árvores esquerda e direita, respectivamente.

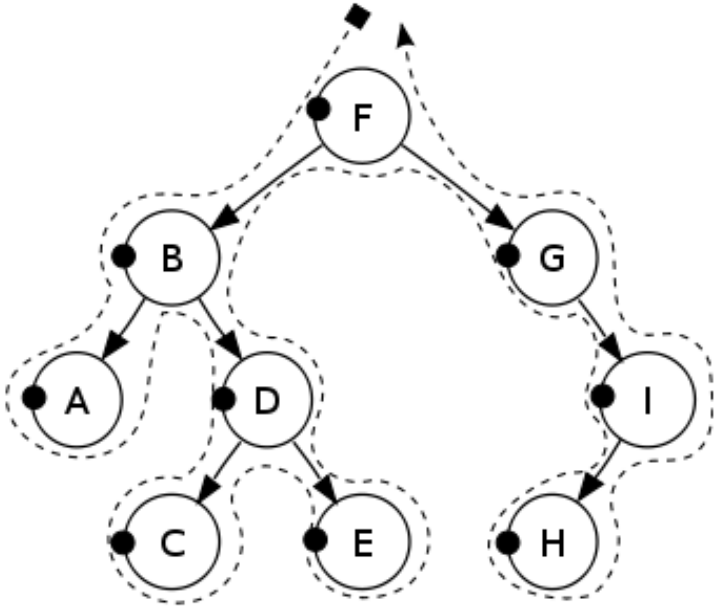
```
void preordem (no *arvore){
        if(arvore != NULL){
                printf("%d\n",arvore->chave);  //visita o nó atual
                preordem(arvore->esq);
                preordem(arvore->dir);
        }
}
```

Unidade 4

Percurso Pré-Ordem

Considere o caminhamento pré-ordem na árvore:

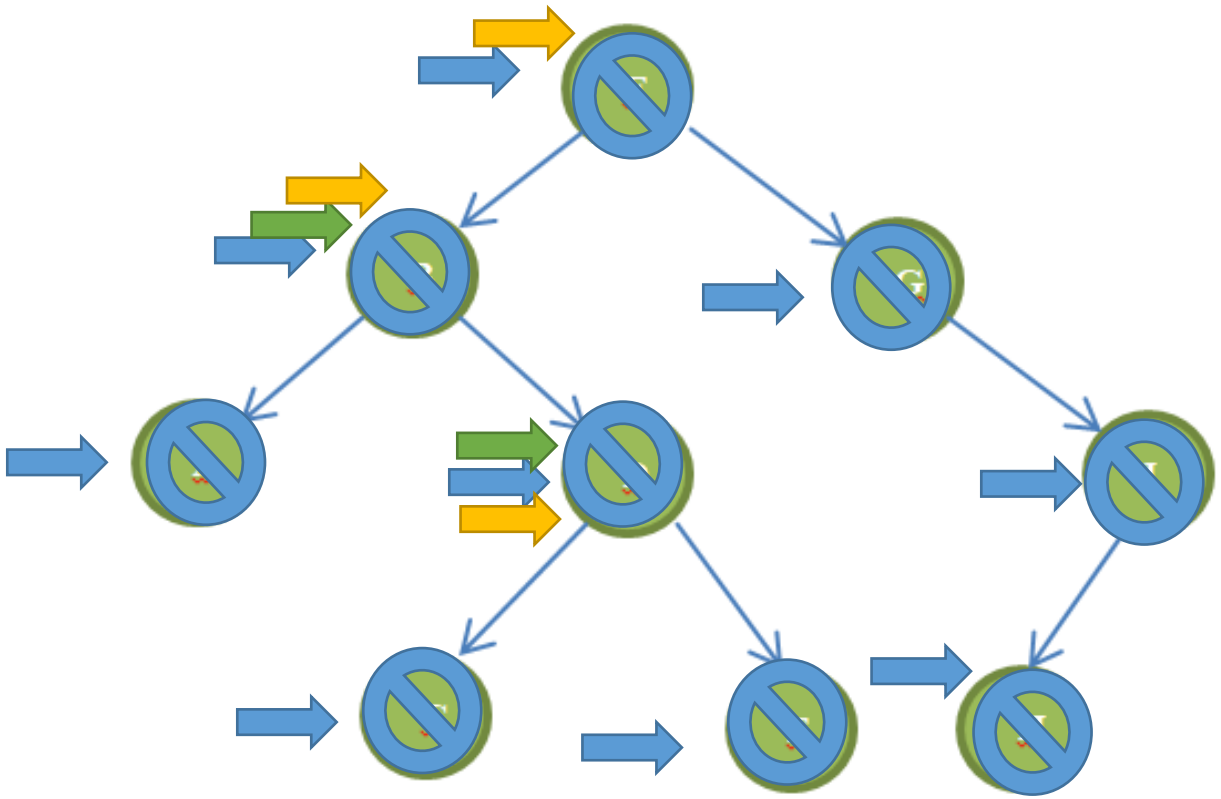




A ordem de visitação produzida pela função pré-ordem(), onde F é o nó raiz, seria a seguinte: F,B,A,D,C,E,G,I,H.

Unidade 4

Percurso Pré-Ordem



F,B,A,D,C,E,G,I,H

Unidade 4

Percurso Em-Ordem

No Caminhamento em-ordem, também conhecido como caminhamento inter-fixado, primeiramente visitamos toda a árvore esquerda e, só então, a raiz é marcada como visitada.

Em seguida, o método em-ordem faz a visitação em toda a sub árvore direita.

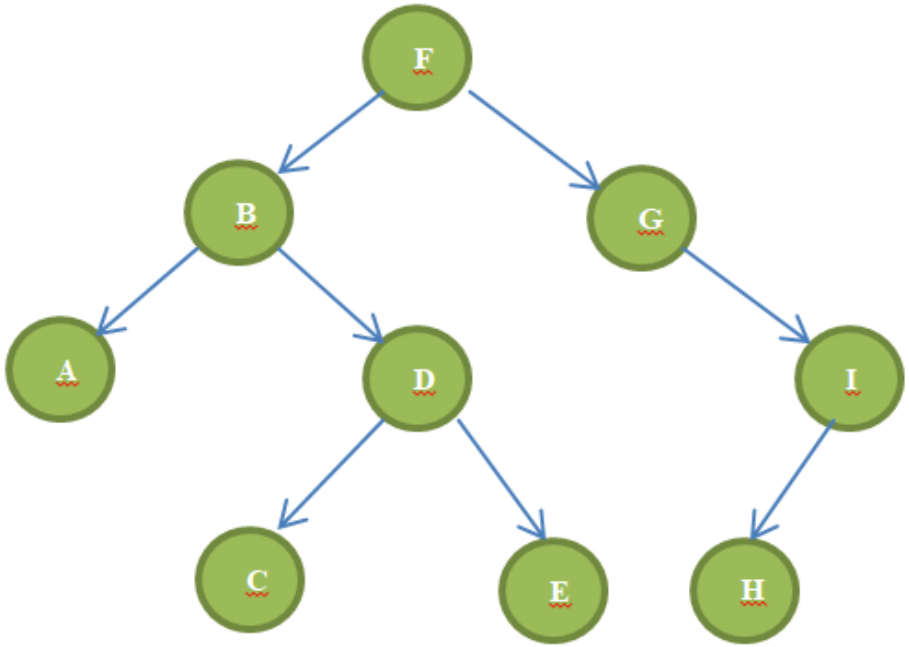
```
void emordem(no *arvore){
        if(arvore != NULL){
                emordem(arvore->esq);
                printf("%d\n",arvore->chave);
                emordem(arvore->dir);
        }
}
```

Unidade 4

Percurso Em-Ordem

Neste caso quando não tem nó a esquerda, lê o nó pai para depois o nó a direita.

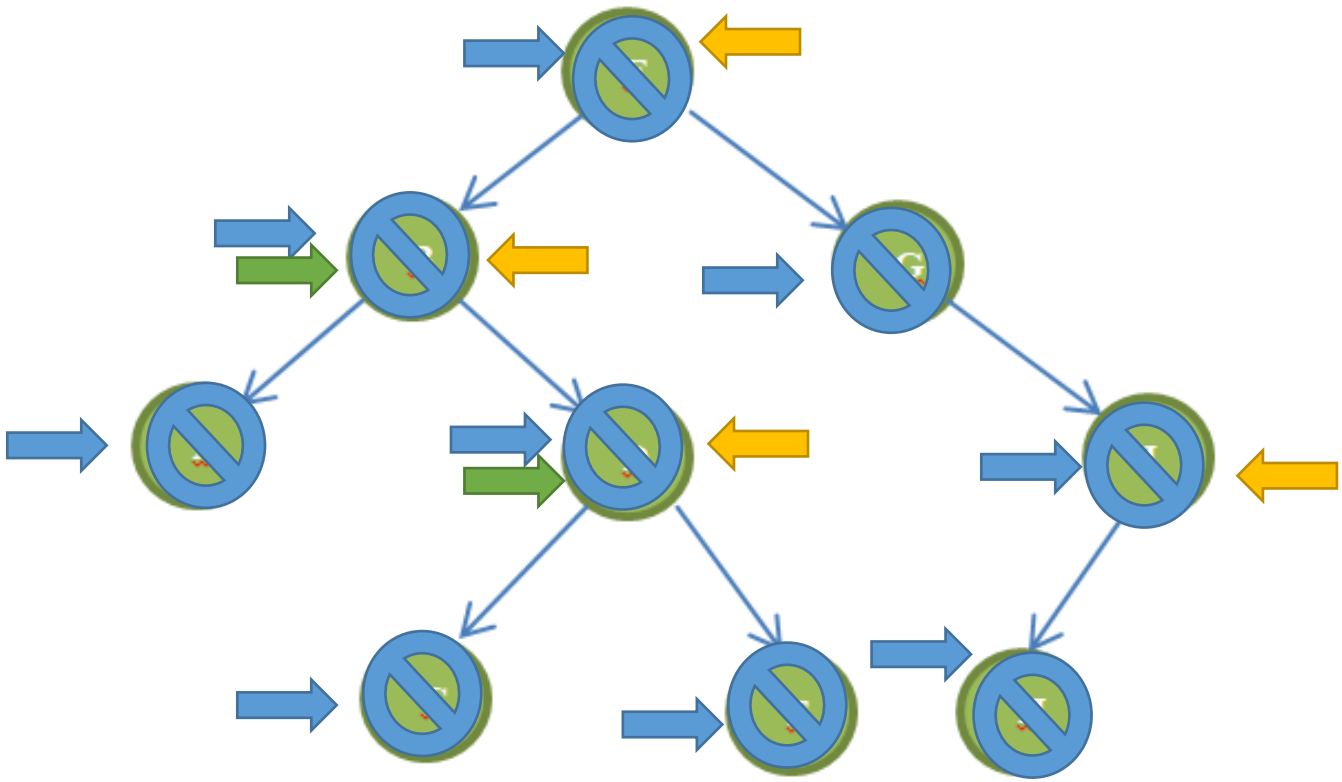
Considere o caminhamento em-ordem na árvore:



A ordem de visitação produzida pela função em-ordem(), onde F é o nó raiz, seria a seguinte: A,B,C,D,E,F,G,H,I.

Unidade 4

Percurso Em-Ordem



A,B,C,D,E,F,G,H,I

Unidade 4

Percurso Pós-Ordem

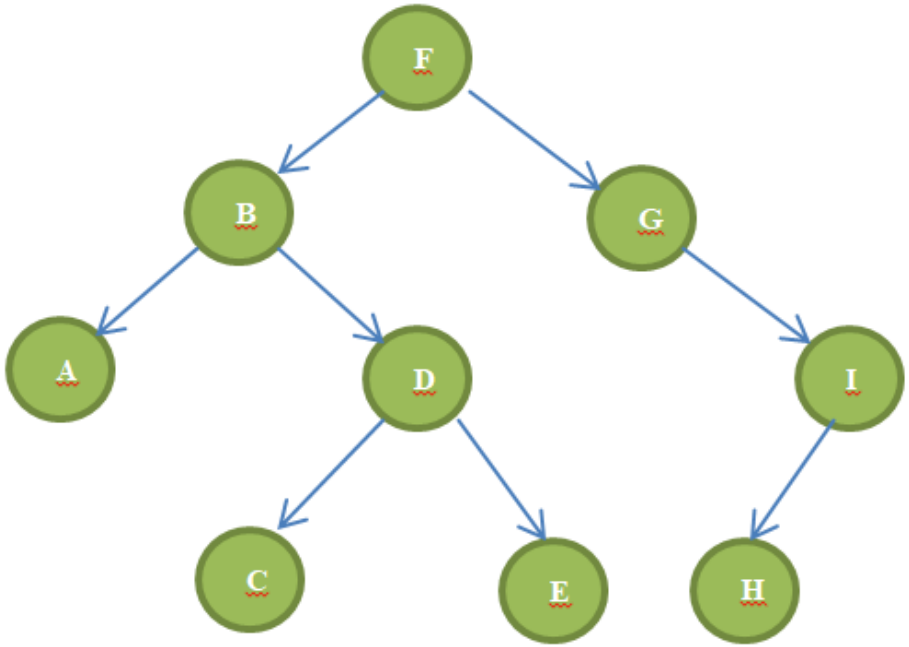
No Caminhamento pós-ordem, também conhecido como caminhamento pós-fixado. Nesse caso, primeiramente visitamos toda a sub árvore esquerda, depois, toda sub árvore a direita. Só após ter visitado as duas sub árvores, é que marcamos o nó corrente como visitado.

```
void posordem(no *arvore){
        if(arvore != NULL){
                posordem(arvore->esq);
                posordem(arvore->dir);
                printf("%d\n",arvore->chave);
        }
}
```

Unidade 4

Percurso Pós-Ordem

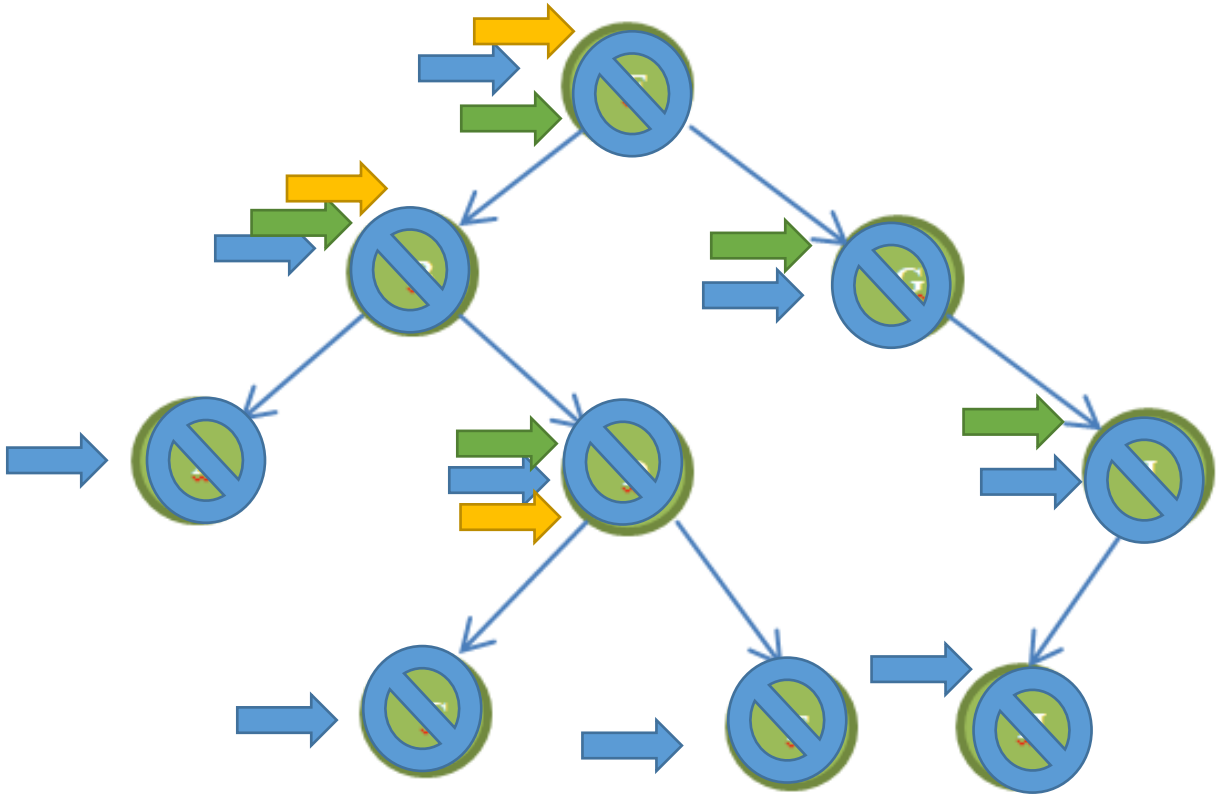
Considere o caminhamento pós-ordem na árvore:



A ordem de visitação produzida pela função pós-ordem(), onde F é o nó raiz, seria a seguinte: A,C,E,D,B,H,I,G,F.

Unidade 4

Percurso Pós-Ordem

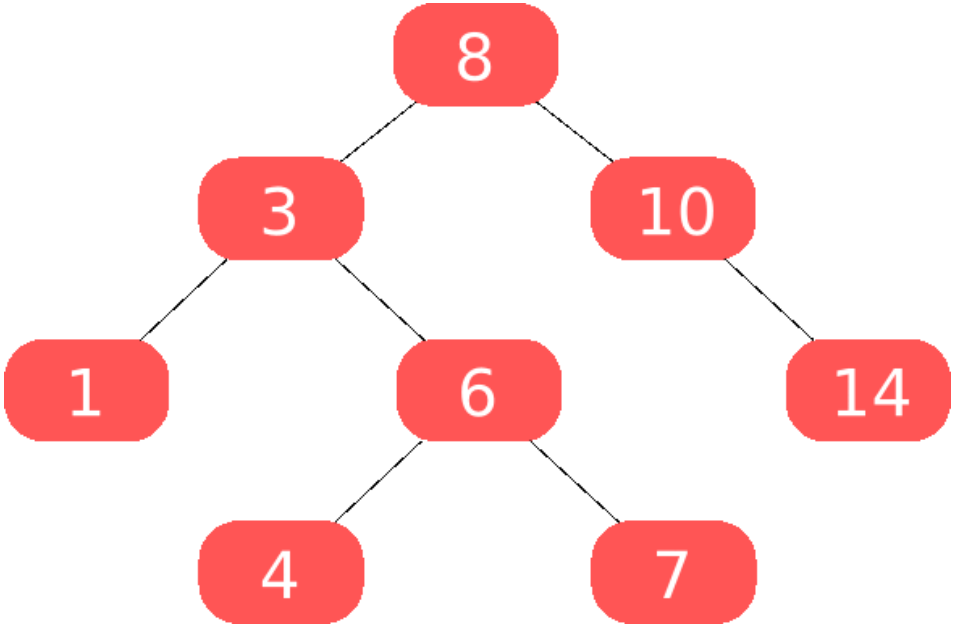


A,C,E,D,B,H,I,G,F

Unidade 4

Busca em Árvores Binárias

Árvores Binárias são muito utilizadas para organizar informações na memória devido ao seu grande potencial de busca em um tempo relativamente curto. Para isso precisamos criar uma árvore binária de busca.



Unidade 4

Busca em Árvores Binárias

Lembrando.....

Uma Árvore Binária ou é vazia ou tem pelo menos um nó raiz. Todo nó tem um pai (menos a raiz) e no máximo dois filhos, um esquerda e um direita. Tanto o filho a esquerda quanto o da direita podem ser sub árvores vazias.

Unidade 4

Busca em Árvores Binárias

Nova regra para construção da árvore:

O nó raiz será o valor que estiver na posição do meio de um vetor (ordenado ou não). Ao adicionar um novo nó na árvore, verificamos se ele é menor do que a raiz.

Caso seja verdade, ele será adicionado na sub árvore a esquerda, caso contrário na sub árvore a direita.

0	1	2	3	4	5	6	7	8	9
3	1	8	7	20	21	31	40	30	0

A posição central do vetor é encontra pela fórmula:

$$meio = \frac{(menor + maior)}{2}$$

Unidade 4

0	1	2	3	4	5	6	7	8	9
3	1	8	7	20	21	31	40	30	0

Busca em Árvores Binárias

Substituindo-se os valores temos:

$$meio = \frac{(menor + maior)}{2}$$

$$meio = \frac{(0 + 9)}{2} \qquad meio = \frac{(9)}{2} \qquad meio = 4{,}5$$

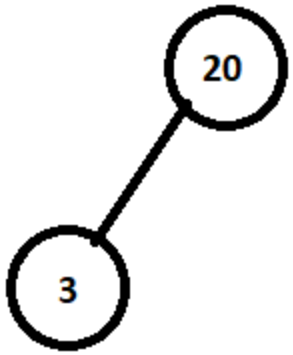
Como os valores de índices de um vetor são inteiros, meio será o 4. Sendo assim, o valor do vetor na posição 4 será a raiz da árvore.

Unidade 4

0	1	2	3	4	5	6	7	8	9
3	1	8	7	20	21	31	40	30	0

Busca em Árvores Binárias

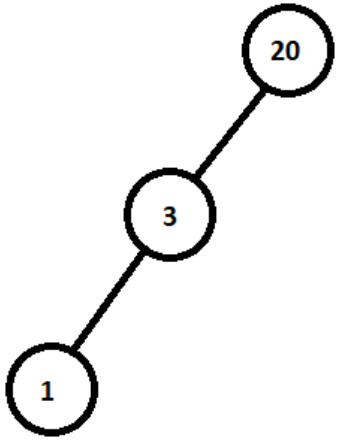
Agora que a nossa árvore possui uma raiz, vamos adicionar o primeiro elemento do vetor na árvore. Como o vetor[0] é menor do que a raiz, seu valor será adicionado na sub árvore esquerda.



0	1	2	3	4	5	6	7	8	9
3	1	8	7	20	21	31	40	30	0

Busca em Árvores Binárias

O próximo vetor[1] também é menor do que a raiz, mas a raiz já possui uma sub árvore esquerda. Fazemos então uma nova comparação com a raiz da sub árvore esquerda. Como o vetor[1] é menor do que a raiz dessa sub árvore, ele será adicionado como seu filho esquerdo.

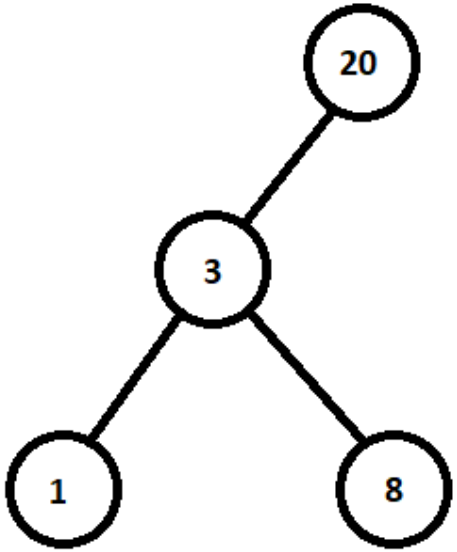


Unidade 4

0	1	2	3	4	5	6	7	8	9
3	1	8	7	20	21	31	40	30	0

Busca em Árvores Binárias

Dando sequência, vamos verificar o valor contido em vetor[2]. Seu valor é menor do que 20, então iremos verificar com o filho esquerdo da raiz. O valor do vetor[2] é maior do que 3, então ele entrará na árvore como seu filho da direita.

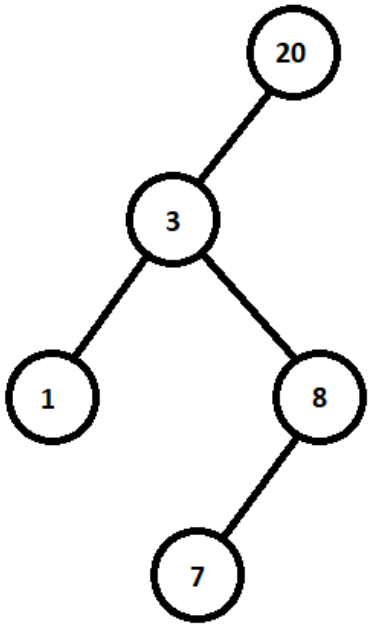


Unidade 4

0	1	2	3	4	5	6	7	8	9
3	1	8	7	20	21	31	40	30	0

Busca em Árvores Binárias

Agora é a vez do vetor[3], que tem o valor 7. Percorreremos a árvore em formação respeitando as regras da árvore binária de busca e um novo nó será adicionado como filho esquerdo de 8.

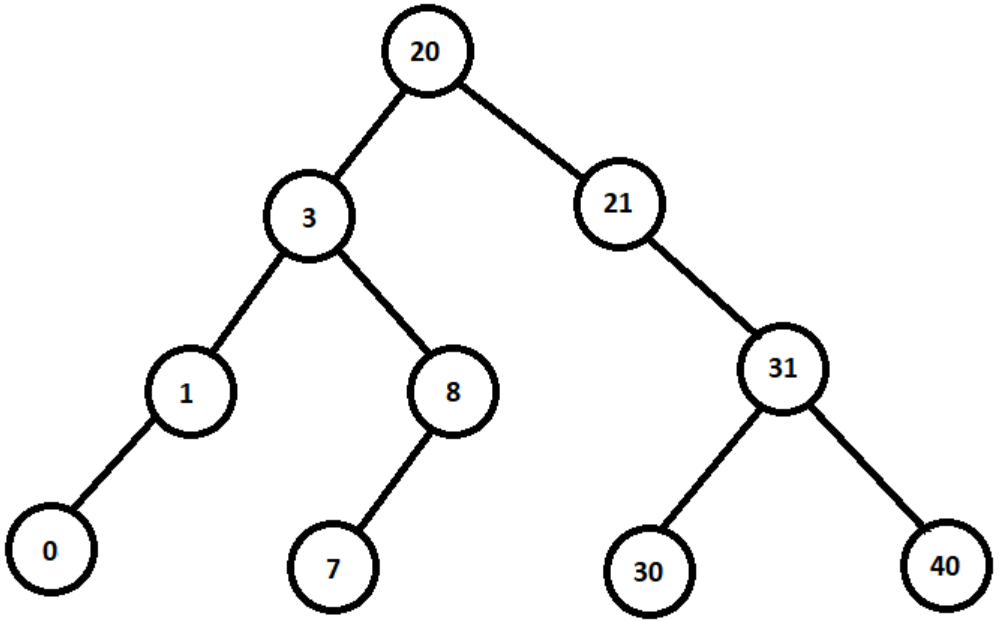


Unidade 4

0	1	2	3	4	5	6	7	8	9
3	1	8	7	20	21	31	40	30	0

Busca em Árvores Binárias

Chegamos agora á metade do vetor. Precisamos de mais cinco interações para finalizar a construção da árvore. Desta forma inserindo os valores 21, 31, 40, 30 e 0 ficam assim:



Unidade 4

0	1	2	3	4	5	6	7	8	9
3	1	8	7	20	21	31	40	30	0

Busca em Árvores Binárias

Agora vem o que nos interessa, que é a realização da busca.

Dado um argumento qualquer, se ele for menor do que a raiz:

- ou ele não existe,
- ou ele se encontra na sua sub árvore esquerda.

Se o valor for maior do que a raiz:

- ou ele não existe,
- ou ele se encontra na sua sub árvore direita.

Unidade 4

0	1	2	3	4	5	6	7	8	9
3	1	8	7	20	21	31	40	30	0

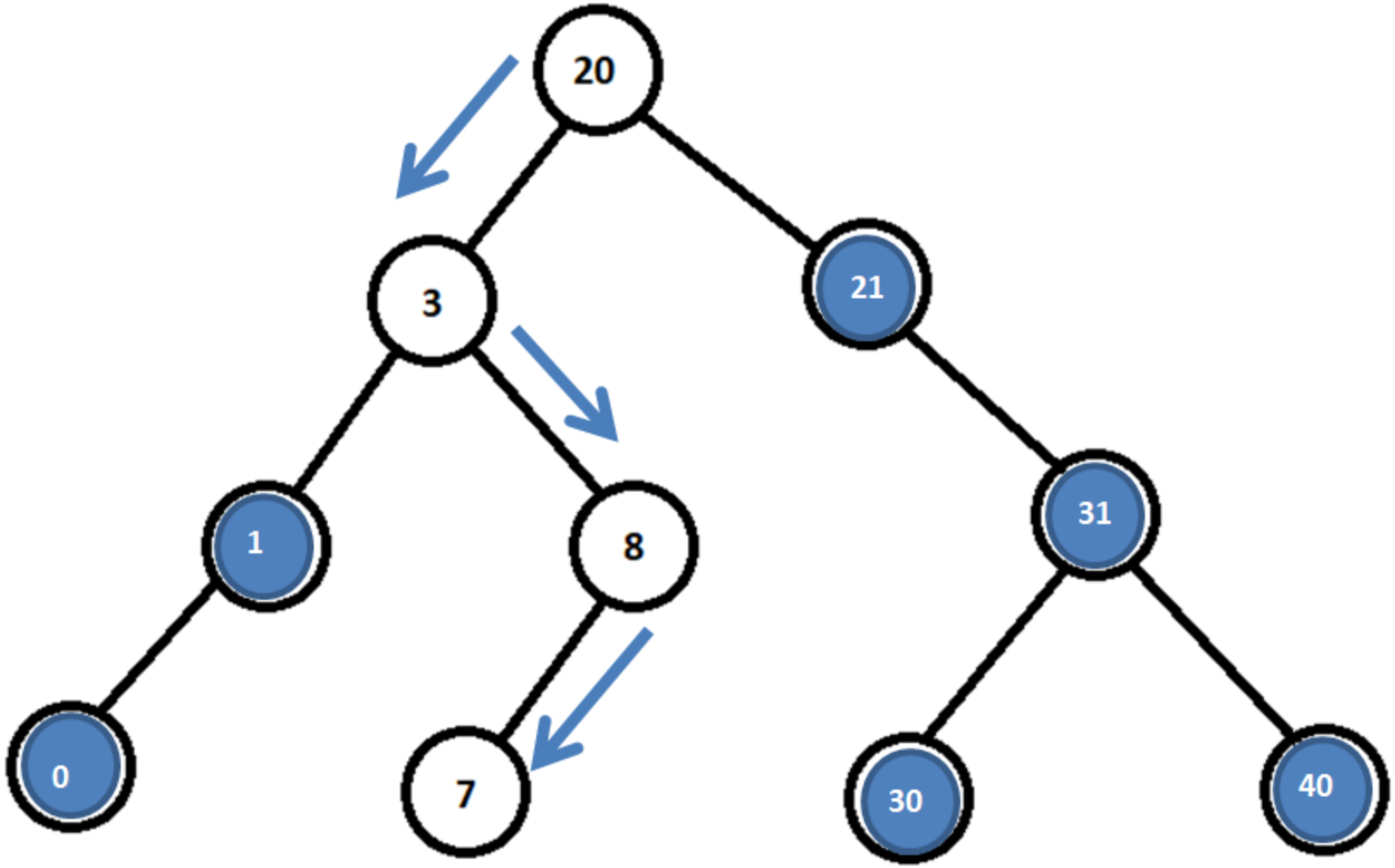
Busca em Árvores Binárias

A partir da raiz vamos descendo pela árvore binária de busca até que o valor seja encontrado ou que encontremos uma folha, ou um sub árvore vazia.

Exemplo: vamos fazer uma busca pelo valor 7. A raiz tem valor 20, então a busca prossegue na sua sub árvore esquerda. Como 7 > 3, a busca continua pela sub árvore direita do nó 3. Como 7 < 8, a busca desce pela sub árvore esquerda do nó 8, até que o valor 7 seja encontrado.

Unidade 4

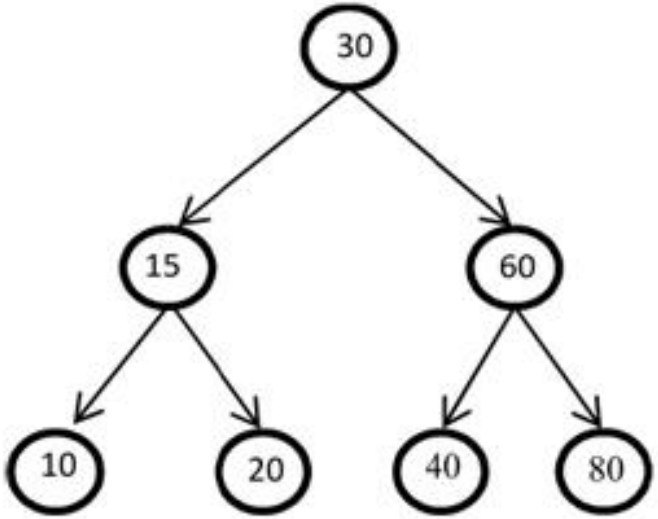
0	1	2	3	4	5	6	7	8	9
3	1	8	7	20	21	31	40	30	0



Unidade 4

Busca em Árvores Binárias

23 **Observe a Árvore Binária de Busca (ABB) a seguir.**



Assinale a alternativa que apresenta, corretamente, a sequência de inserção que gera essa ABB.

a) 30, 15, 40, 10, 20, 60, 80

b) 30, 15, 40, 10, 20, 80, 60

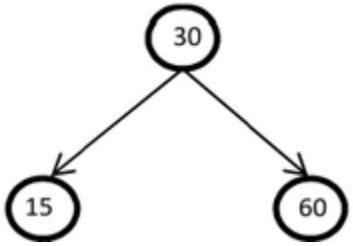
c) 30, 15, 60, 10, 20, 40, 80

d) 30, 60, 20, 80, 15, 10, 40

e) 30, 60, 40, 10, 20, 15, 80

Unidade 4

Busca em Árvores Binárias



Resolvendo a questão:

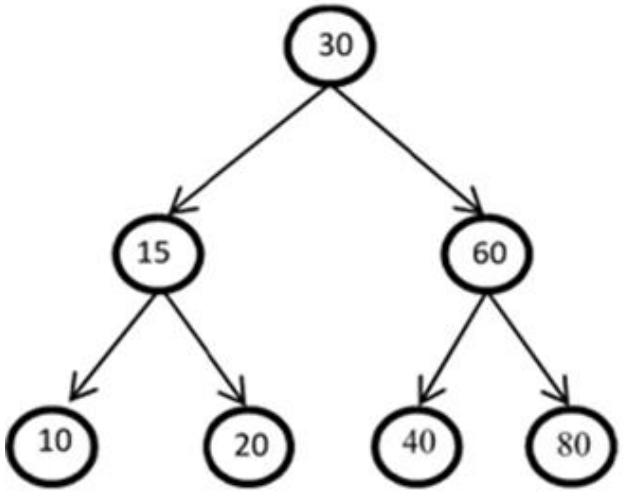
O primeiro nó a adicionar é o 30, já que ele é a raiz;

O segundo e o terceiro devem ser o 15 e o 60 independente da ordem, já que os valores numéricos maiores que a raiz vão para a sub-árvore direita e os menores para a sub-árvore esquerda. Assim temos a árvore binária ao lado;

Para gerarmos a arvore descrita na questão, temos agora que adicionar a nossa árvore mais quatro nós, são eles: 10, 20, 40 e 80. Devemos então continuar a adicionar a partir do nó com menor valor, no nosso caso o 10, pois como ele é menor que 30, descemos para a sub-árvore a esquerda e como é menor que 15, ele se tornará seu filho a esquerda;

Unidade 4

Busca em Árvores Binárias



Resolvendo a questão:

Em seguida adicionamos o 20, que assim como o 10, desce para a sub-árvore a esquerda de 30, porém, em seguida, como é maior que 15, se torna seu filho a direita;

Por fim adicionamos os nós 40 e 80, nessa ordem, pois, de acordo com a explicação anterior, respectivamente tornam-se o filho à esquerda e o filho à direita de 60;

Existem outras ordens de inserção que gerariam a mesma árvore, porém, para essa questão a resposta certa é a “letra c”;

Unidade 4

Referências:

SZWARCFITER, Jayme Luiz; MARKENZON, Lilian. Estruturas de dados e seus algoritmos. 2ed. Rio de Janeiro: LTC, 1994. 320p.

TENENBAUM, Aaron M.; LANGSAM, Yedidyah; AUGENSTEIN, Moshé J.. Estruturas de dados usando C. São Paulo: Makron Books, 1995. 884p.

VELOSO, Paulo et al.. Estruturas de dados. Rio de Janeiro: Campus, 2001. 228p

Atividades - Unidade 4


Unicesumar