Anhanguera

Aqui o seu esforço ganha força.

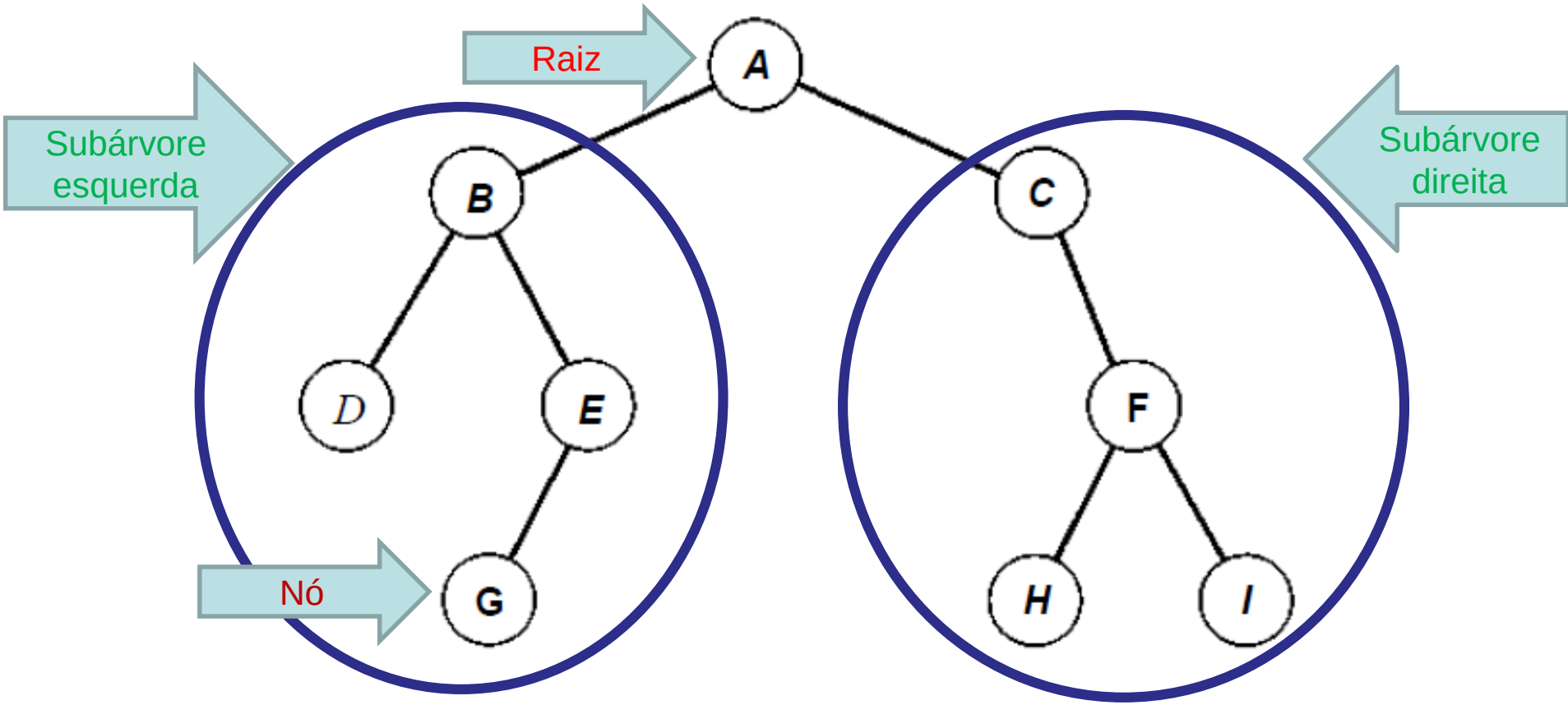
Árvores

Prof. Esp. Rodrigo Hentz

Árvores Binárias

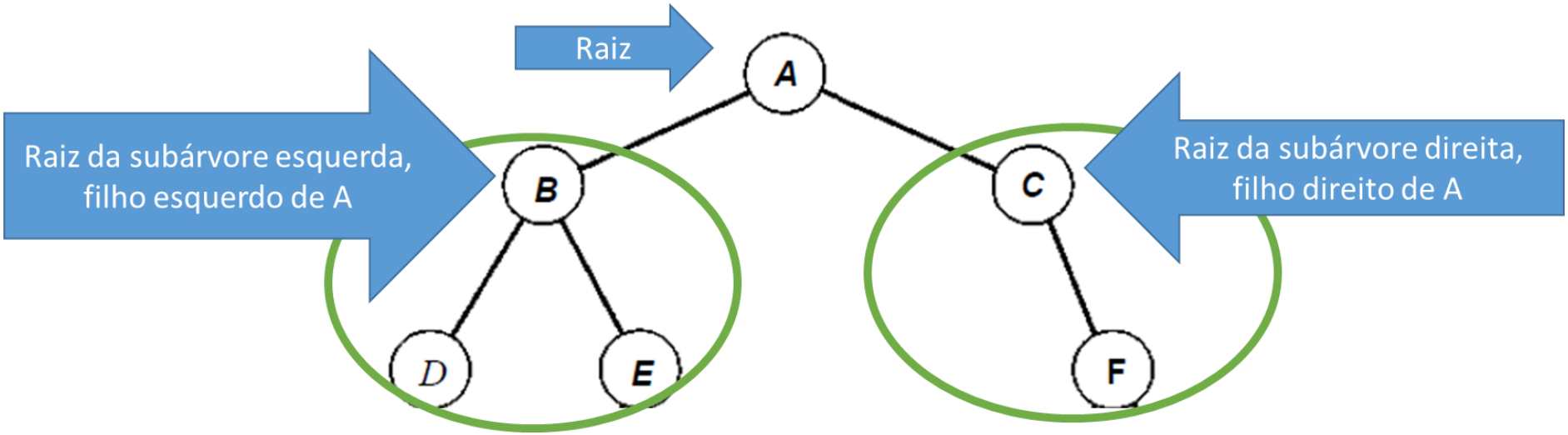
- Uma árvore binária é um conjunto finito de elementos que está vazio ou é particionado em três subconjuntos disjuntos.
- O primeiro subconjunto contém um único elemento, chamado raiz da árvore.
- Os outros dois subconjuntos são em si mesmos árvores binárias, chamadas subárvore esquerda e direita da árvore original.
- Uma subárvore esquerda ou direita pode estar vazia. Cada elemento de uma árvore binária é chamado nó da árvore.

Árvores Binárias - Exemplo



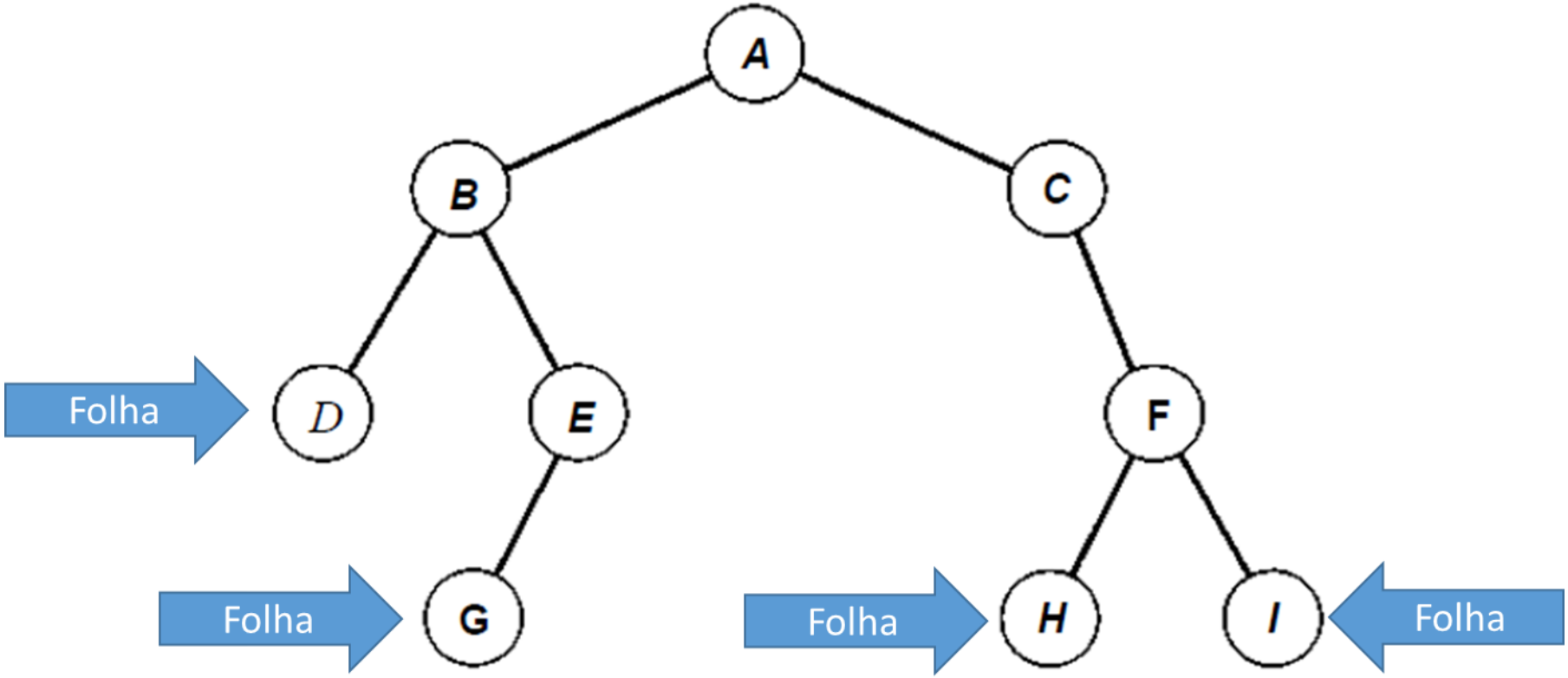
Árvores Binárias

- Se A é a raiz de uma árvore binária e B é a raiz de sua subárvore direita ou esquerda, então diz-se que A é o pai de B e que B é o filho direito ou esquerdo de A.



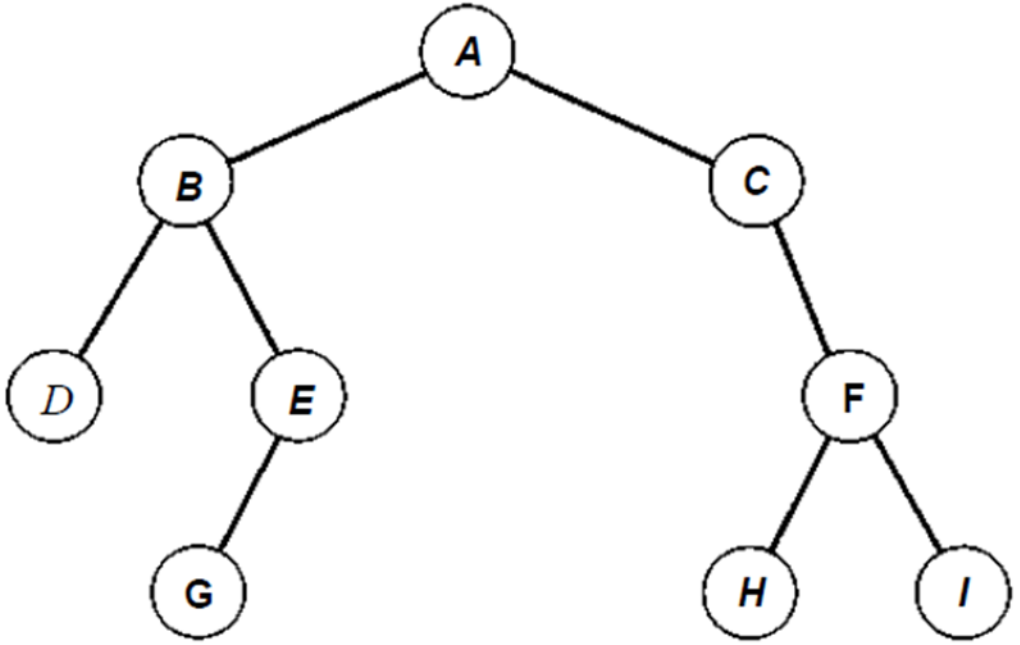
Árvores Binárias

- Um nó sem filhos (como D, G, H ou I) é chamado ***folha***.



Árvores Binárias

- O nó n1 é um ancestral do nó n2 (e n2 é um descendente de nl), se n1 for o pai de n2 ou o pai de algum ancestral de n2.



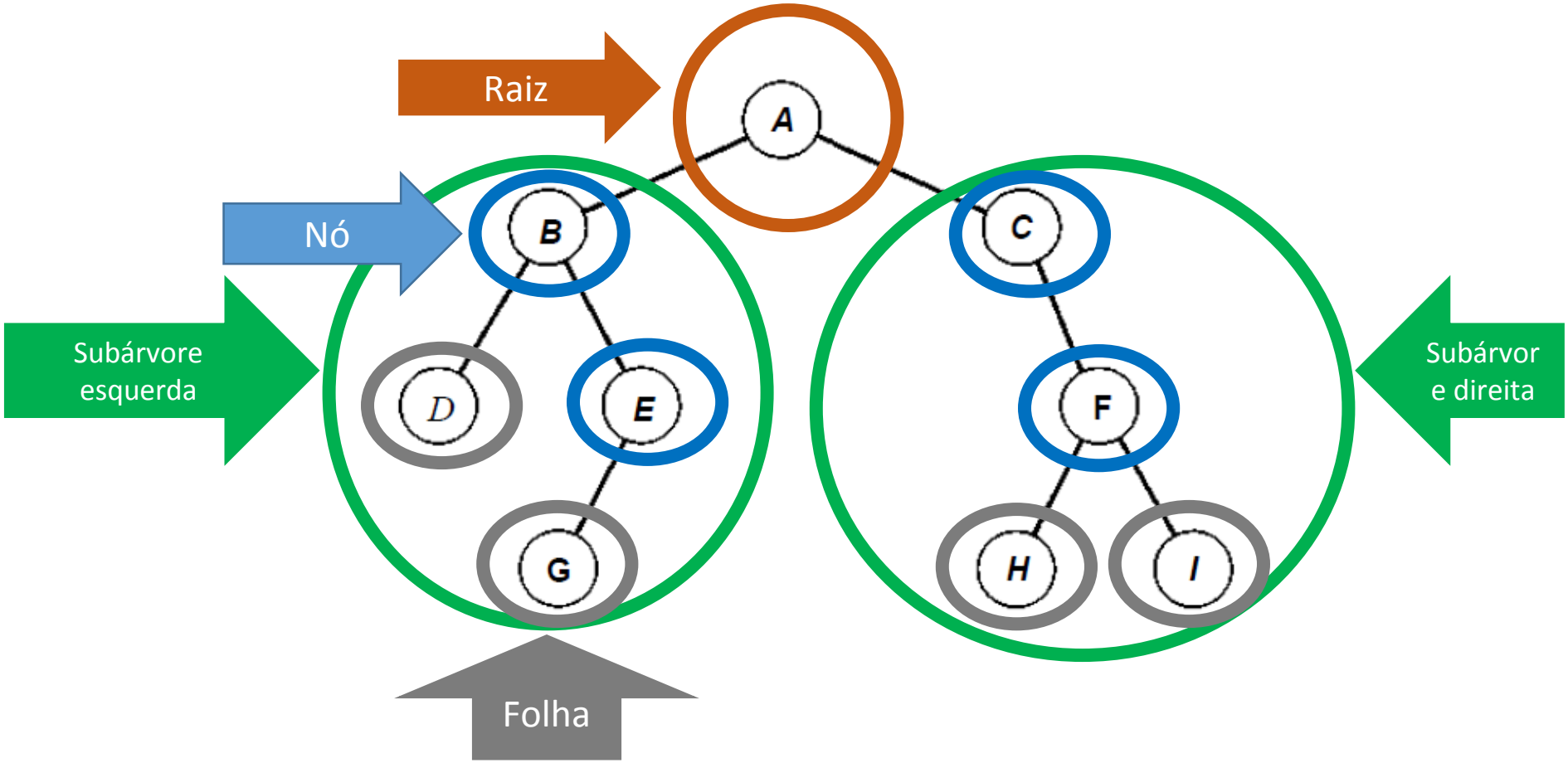
- A é um ancestral de G
- H é um descendente de C
- E não é nem ancestral nem descendente de C

Árvores Binárias

- Embora as árvores naturais cresçam com suas raízes fincadas na terra e suas folhas no ar, os cientistas de computação retratam quase universalmente as estruturas de dados em árvore com a raiz no topo e as folhas no chão.
- O sentido da raiz para as folhas é "para baixo" e o sentido oposto é "para cima".
- Quando você percorre uma árvore a partir das folhas na direção da raiz, diz-se que você está "subindo" a árvore, e se partir da raiz para as folhas, você está "descendo" a árvore.

Anhanguera

Árvores Binárias

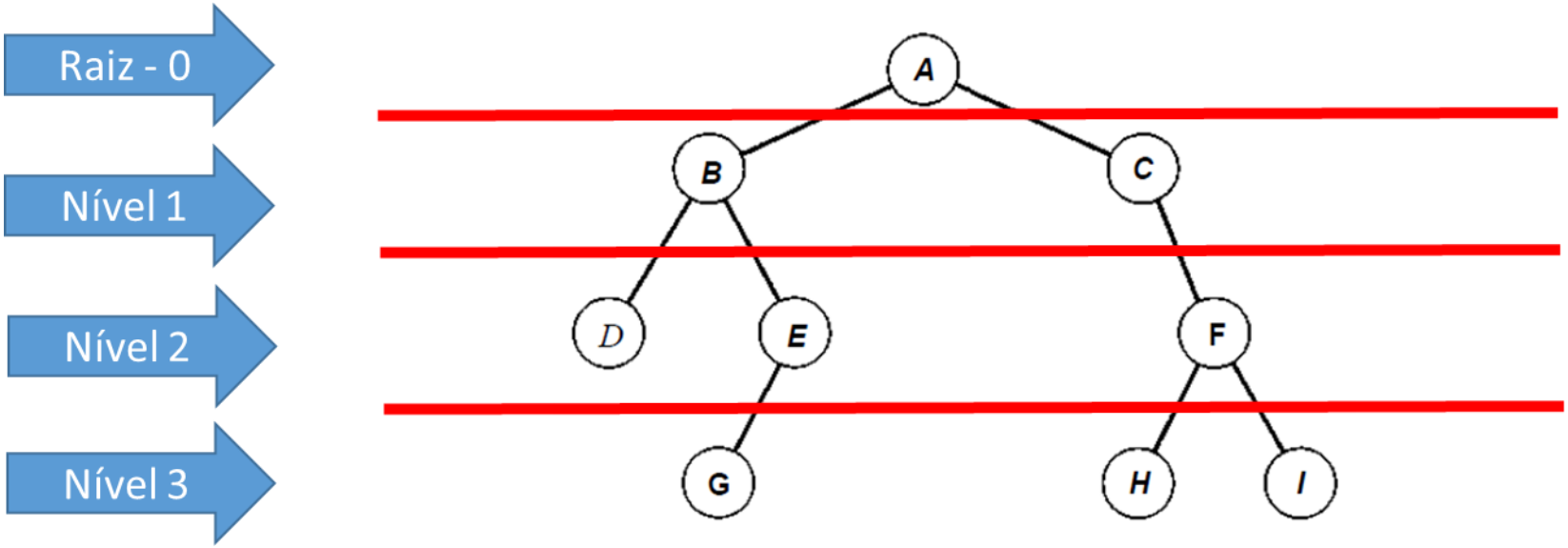


Anhanguera

Árvores Binárias

Nível.

- O nível de um nó numa árvore binária é definido como segue: a raiz da árvore tem nível 0, e o nível de qualquer outro nó na árvore é um nível a mais que o nível de seu pai.

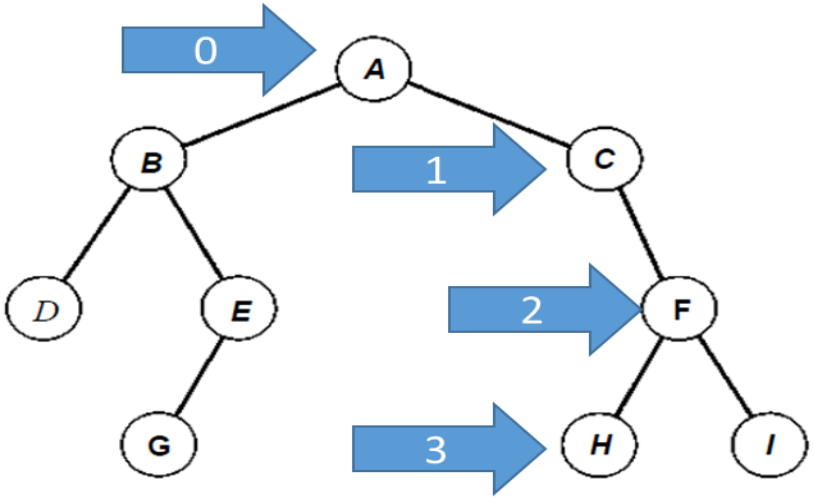


Anhanguera

Árvores Binárias

Profundidade.

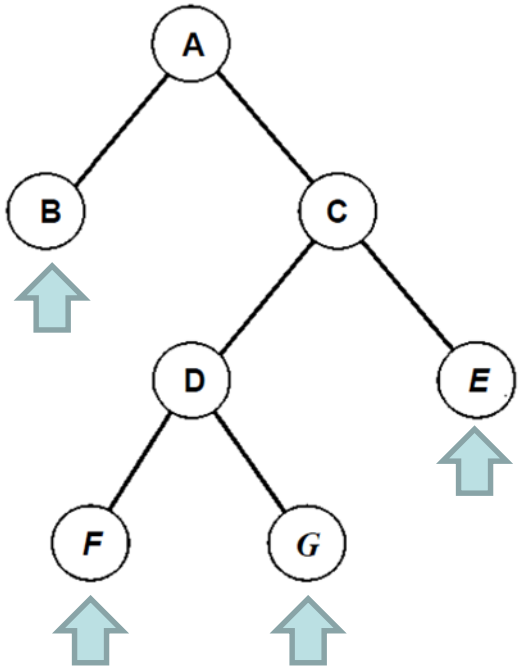
- A profundidade de uma árvore binária significa o nível máximo de qualquer folha na árvore. Isso equivale ao tamanho do percurso mais distante da raiz até qualquer folha. Sendo assim, a profundidade da árvore da figura é 3.



Árvores Binárias

Árvores Estritamente Binárias

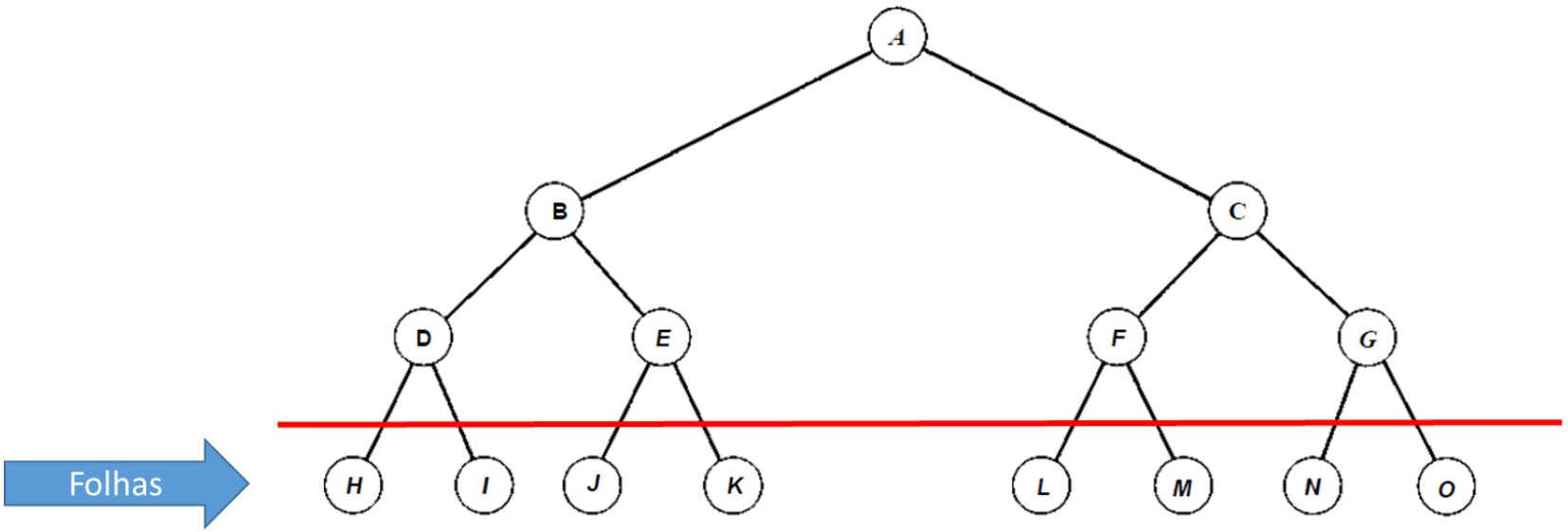
- Se todo **nó** que não é **folha** numa árvore binária tiver **subárvores esquerda e direita não-vazias**, a árvore será considerada uma árvore estritamente binária.
- Uma árvore estritamente binária com **n** folhas contém sempre 2n - 1 nós.
- A comprovação desse fato pode ser comprovada através do exemplo ao lado.
- Nós = 2n – 1 => 2(**4**) – 1 => 8 – 1 => 7



Árvores Binárias

Árvores Binárias Completa

- Uma árvore binária completa é aquela onde todas as folhas estejam no nível da profundidade da árvore.

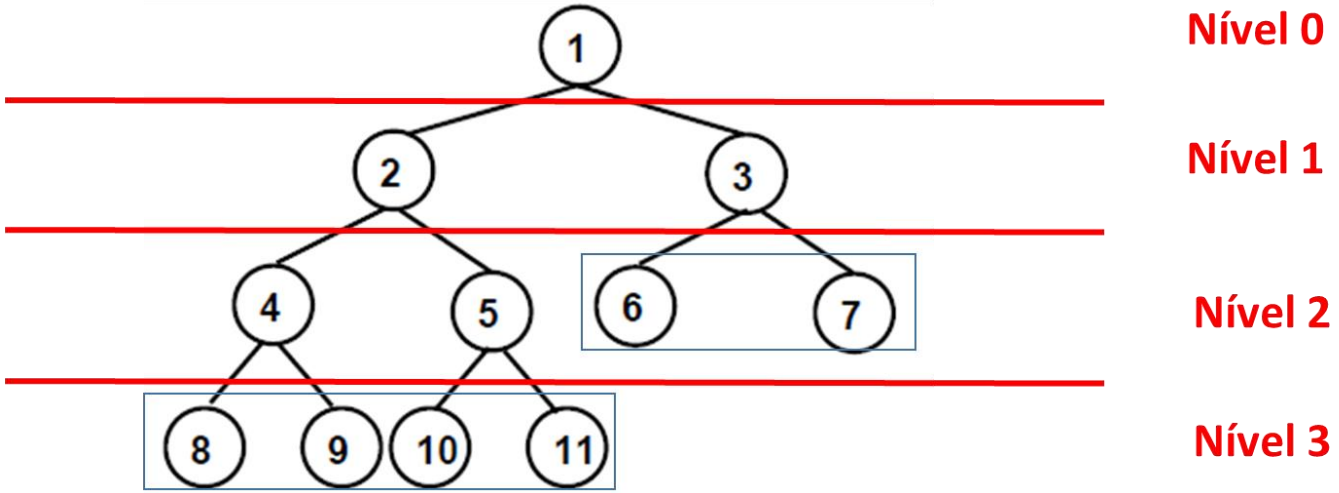


Anhanguera

Árvores Binárias

Árvores Binárias Quase Completa

- Cada folha na árvore deve estar no nível d ou no nível d - 1.
- Para cada nó da árvore com um descendente direito no nível d, todos os descendentes esquerdos do nó que forem folhas estiverem também no nível d.



Anhanguera

Árvores Binárias

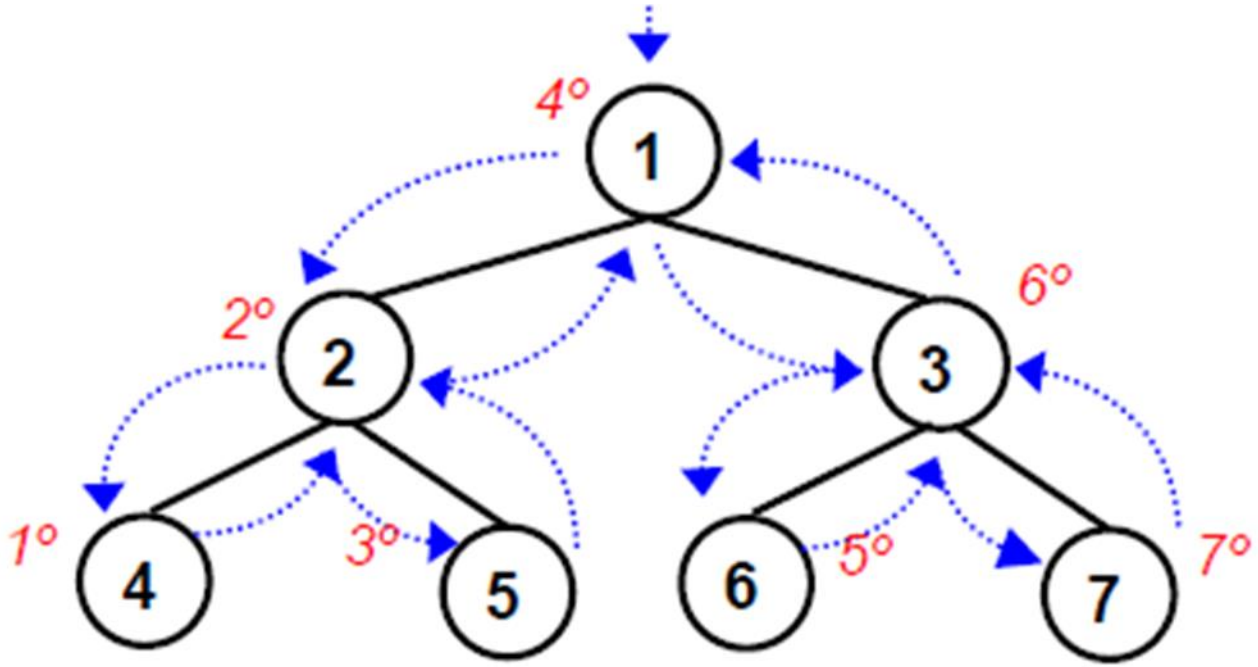
Percursos

- Existem três maneiras de percorrer uma árvore:
  - de forma ordenada (simétrica)
  - pré-ordenada (profundidade)
  - pós-ordenada.

Árvores Binárias

Percurso Ordenado

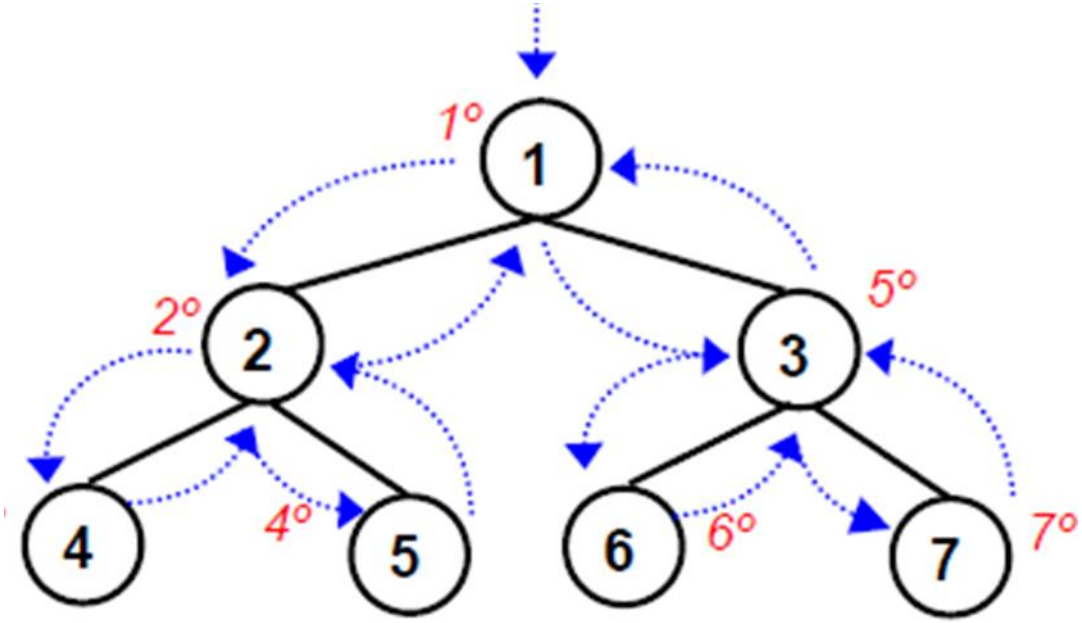
- Usando a forma ordenada, consultamos a subárvore da esquerda, a raiz e, em seguida, a subárvore direita.



Árvores Binárias

Percurso Pré-Ordenado

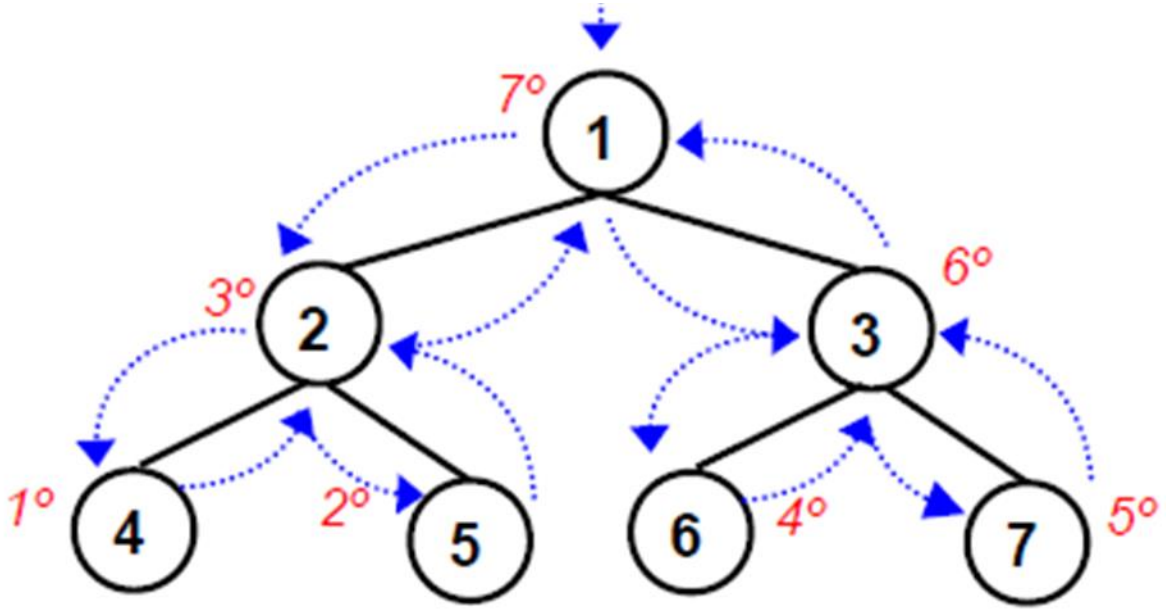
- Na maneira pré-ordenada, visitamos a raiz, a subávore da esquerda e em seguida a subárvore da direita.



Árvores Binárias

Percurso Pós-Ordenado

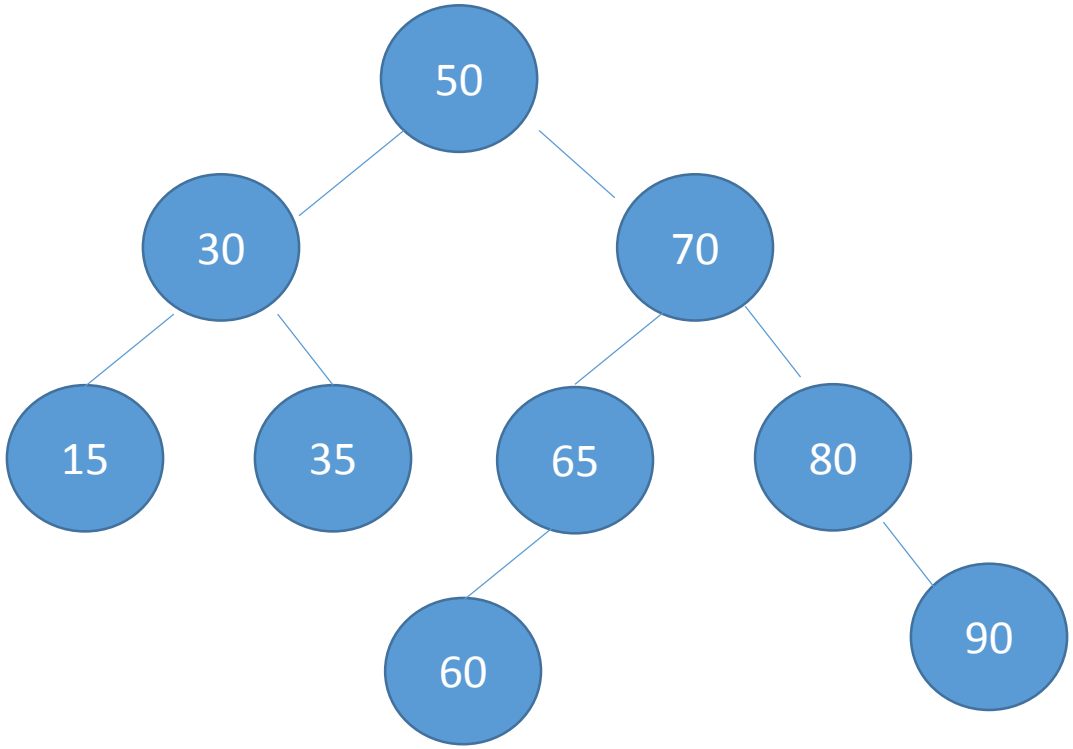
- Na forma pós-ordenada, visitamos a subárvore esquerda, a subárvore direita e depois a raiz.



Árvores Binárias

Inclusão de nós em árvores binárias

Valores: 50, 30, 70, 15, 35, 65, 80, 90 e 60



Anhanguera

Árvores Binárias

Implementação de árvore binária com alocação dinâmica

```
struct arvore
{
    int valor;
    struct arvore* esquerda;
    struct arvore* direita;
};
```

Anhanguera

Inclusão de nó

```
struct arvore* sarvore(struct arvore* raiz, struct arvore* filho, int valor)
{
    if(filho == NULL)
    {
        filho = (struct arvore*)malloc(sizeof(struct arvore));
        if (filho == NULL)
        {
            printf("\nErro alocando memoria.");
            exit(0);
        }
        filho->esquerda = NULL;
        filho->direita = NULL;
        filho->valor = valor;
        if(raiz == NULL) return filho;
        if(valor < raiz->valor) raiz->esquerda = filho;
        else raiz->direita = filho;
        return filho;
    }
    if (valor < filho->valor) sarvore(filho, filho->esquerda, valor);
    else sarvore(filho, filho->direita, valor);
}
```

Exclusão de nó

```
struct arvore* darvore(struct arvore* raiz, int valor)
{
    struct arvore *p, *p2;
    if(raiz == NULL) return raiz;
    if(raiz->valor == valor)
    {
        if(raiz->esquerda == raiz->direita)
        {
            free(raiz); return NULL;
        }
        else if (raiz->esquerda == NULL)
        {
            p = raiz->direita; free(raiz); return p;
        }
        else if (raiz->direita == NULL)
        {
            p = raiz->esquerda; free(raiz); return p;
        }
        else
        {
            p2 = raiz->direita; p = raiz->direita;
            while(p->esquerda) p = p->esquerda;
            p->esquerda = raiz->esquerda;
            free(raiz); return p2;
        }
    }
    if (raiz->valor < valor) raiz->direita = darvore(raiz->direita, valor);
    else raiz->esquerda = darvore(raiz->esquerda, valor);
    return raiz;
}
```

Anhanguera

Pesquisa de nó

```
void pesquisar(struct arvore* raiz, int valor)
{
    struct arvore* aux;
    aux = raiz;
    if (aux == NULL) printf("\nValor nao encontrado.");
    else
    {
      while(aux->valor != valor)
      {
        if (valor < aux->valor) aux = aux->esquerda;
        else aux = aux->direita;
        if (aux == NULL) break;
      }
      if (aux != NULL) printf("\nValor encontado.");
      else printf("\nValor nao encontrado.");
    }
}
```

Anhanguera

Impressão ordenada

```
void imprimirOrdenado(struct arvore* raiz)
{
    if (raiz == NULL) return;
    imprimirOrdenado(raiz->esquerda);
    if (raiz->valor != NULL) printf("%d ", raiz->valor);
    imprimirOrdenado(raiz->direita);
}
```

Anhanguera

Impressão pré-ordenada

```
void imprimirPreOrdenado(struct arvore* raiz)
{
    if (raiz == NULL) return;
    if (raiz->valor != NULL) printf("%d ", raiz->valor);
    imprimirPreOrdenado(raiz->esquerda);
    imprimirPreOrdenado(raiz->direita);
}
```

Anhanguera

Impressão pós-ordenada

```
void imprimirPosOrdenado(struct arvore* raiz)
{
    if (raiz == NULL) return;
    imprimirPosOrdenado(raiz->esquerda);
    imprimirPosOrdenado(raiz->direita);
    if (raiz->valor != NULL) printf("%d ", raiz->valor);
}
```

Anhanguera

Principal

```
int main(int argc, char** argv) {
    int num, opcao;
    struct arvore* raiz;
    do
    {
        printf("\n");
        printf(" 1 - Iniciar arvore\n");
        printf(" 2 - Incluir no\n");
        printf(" 3 - Pesquisar no\n");
        printf(" 4 - Excluir no\n");
        printf(" 5 - Impressao ordenada\n");
        printf(" 6 - Impressao pré-ordenada\n");
        printf(" 7 - Impressao pós-ordenada\n");
        printf("0 - SAIR\n");
        printf("\nEntre com a opcao: ");
        scanf("%d", &opcao);
```

```
switch (opcao)
{
    case 1:
        raiz = NULL;
        printf("\nArvore iniciada.");
        break;
    case 2:
        printf ("\nEntre com o numero para o novo no: ");
        scanf ("%d", &num);
        if (raiz == NULL) raiz = sarvore(raiz, raiz, num);
        else sarvore(raiz, raiz, num);
        break;
    case 3:
        printf ("\nEntre com o numero para pesquisa: ");
        scanf ("%d", &num);
        pesquisar(raiz, num);
        break;
    case 4:
        printf ("\nEntre com o numero para excluir: ");
        scanf ("%d", &num);
        raiz = darvore(raiz, num);
        break;
```

```
            case 5:
                imprimirOrdenado(raiz);
                break;
            case 6:
                imprimirPreOrdenado(raiz);
                break;
            case 7:
                imprimirPosOrdenado(raiz);
                break;
        }
        fflush(stdin);
    } while (opcao != 0);
    return 0;
}
```

Anhanguera

Aqui o seu esforço ganha força.