Lógica de Programação

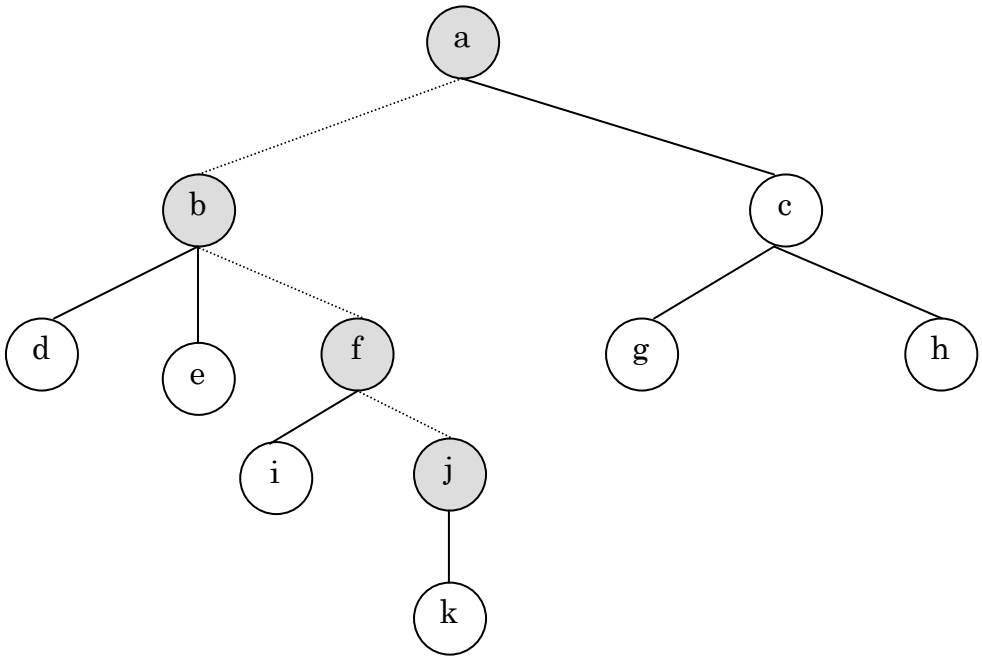
ÁRVORES

Uma árvore é uma estrutura de dados bidimensional, não linear, que possui propriedades espaciais e admite muitas operações de conjuntos dinâmicos, tais como: consulta, inserção, remoção, entre outros. É diferente das listas e pilhas, uma vez que estas são estruturas de dados lineares.

Uma árvore, de modo geral, possui as seguintes características:

- nó raiz: nó do topo da árvore, do qual descendem os demais nós. É o primeiro nó da árvore;
- nó interior: nó do interior da árvore (que possui descendentes);
- nó terminal: nó que não possui descendentes;
- trajetória: número de nós que devem ser percorridos até o nó determinado;
- grau do nó: número de nós descendentes do nó, ou seja, o número de subárvores de um nó;
- grau da árvore: número máximo de subárvores de um nó;
- altura da árvore: número máximo de níveis dos seus nós;
- altura do nó: número máximo de níveis dos seus nós;

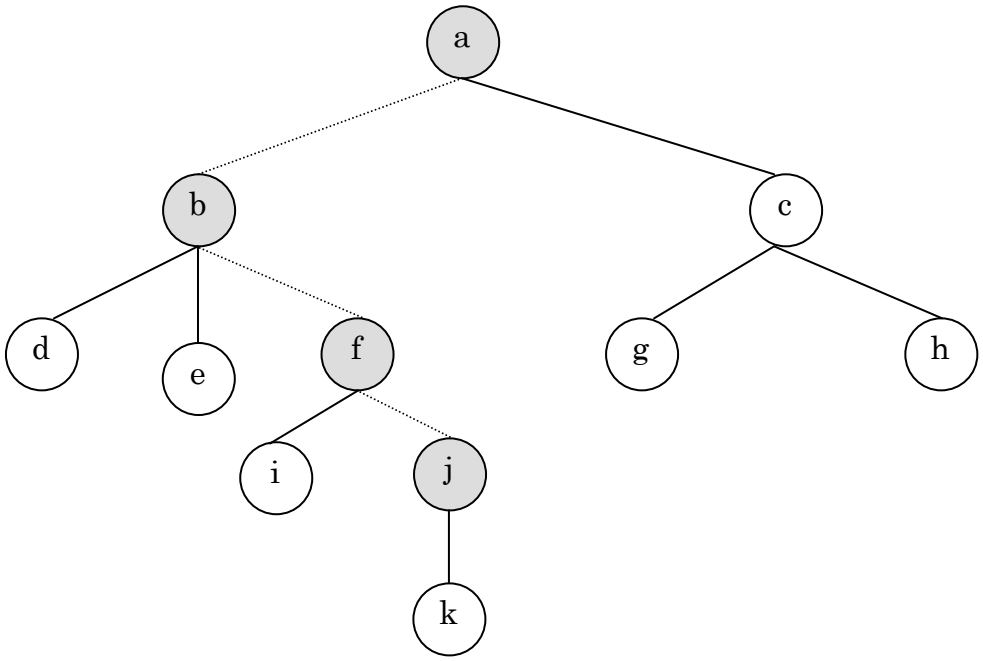
Para exemplificar a explicação sobre as características de uma árvore, vamos fazer uma análise da árvore apresentada na figura 1:



| Figura 1 | *Árvores*

- o nó a é determinado nó raiz, tem grau dois pois possui dois filhos, os nós b e c, que também podem ser chamados de subárvores ou nós descendentes;
- o nó b tem grau três pois possui três filhos : os nós d, e e f; o nó b também é denominado pai dos nós d, e e f;

- os nós d e e são nós terminais, isto é, não possuem descendentes e por isso têm grau zero;
- o nó f tem grau dois e tem como filhos os nós i e j;
- o nó i é um nó terminal e possui grau zero;
- o nó j tem grau um e é pai do nó k, que é terminal;
- o nó c tem grau dois e é pai dos nós g e h, que são nós terminais;
- a árvore possui grau três, pois este é o número máximo de nós descendentes de um único pai;
- a árvore tem altura igual a 5, já o nó b tem altura igual a 4, o nó c tem altura igual a 2, o nó k tem altura igual a 1 e assim por diante;
- para definirmos a trajetória a ser percorrida vamos supor que se deseje chegar ao nó j, então o caminho a ser percorrido será a, b, f, j, conforme ilustrado na figura 2.



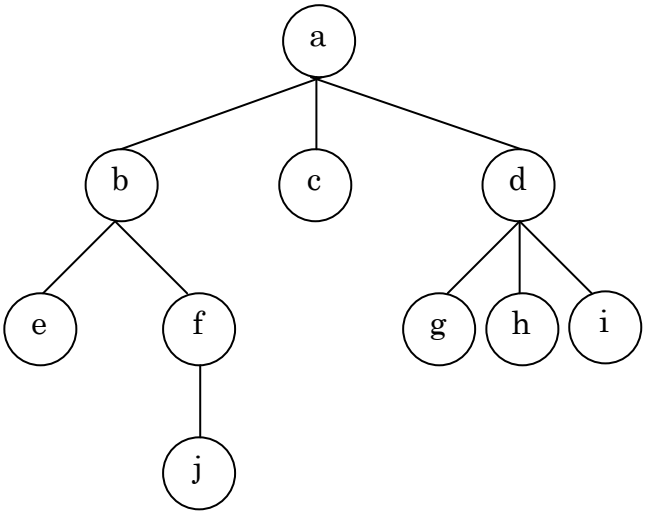
| Figura 2 | *Trajetória*

As árvores podem ser do tipo listas generalizadas ou binárias. As árvores do tipo listas generalizadas possuem nós com grau maior ou igual a zero, enquanto uma árvore do tipo binária sempre possui nós com grau menor ou igual a 2. Veja os exemplos de árvores apresentados na Figura 3.

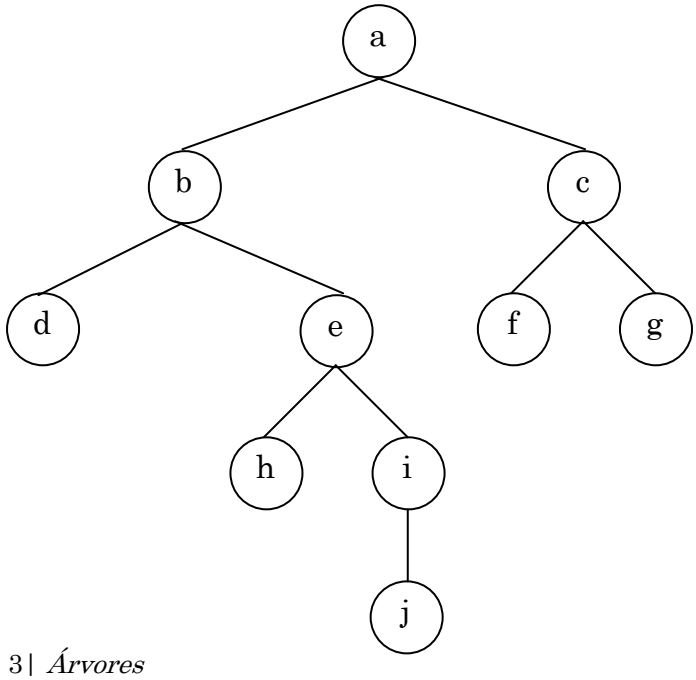
ÁRVORES BINÁRIAS

Conforme já dissemos anteriormente, uma árvore binária sempre possui nós com grau menor ou igual a dois, isto é, nenhum nó possui mais do que dois descendentes

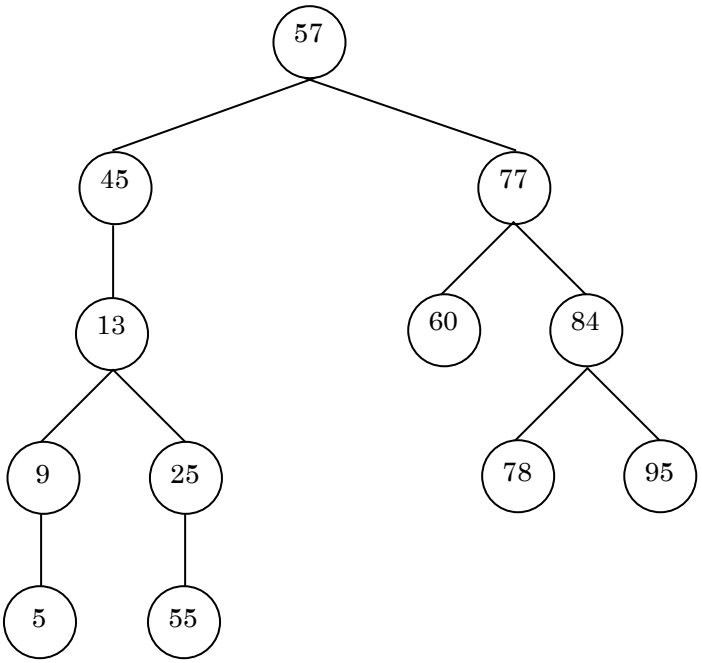
Árvore como lista generalizada



Árvore binária



| Figura 3 | *Árvores*



| Figura 4 | *Árvore Binária*

diretos (dois filhos). Nesse tipo de árvore também existe uma particularidade quanto à posição dos nós: os nós da direita sempre possuem valor superior ao do nó pai, e os nós da esquerda sempre possuem valor inferior ao do nó pai.

O algoritmo a seguir apresenta as variáveis que serão utilizadas para a manipulação da árvore – note que existe grande similaridade com os nós criados para manipulação das listas. O algoritmo tem a definição de um

registro que possui as variáveis *valor*, *esq* e *dir*, a variável *apontador*, que será utilizada para fazer a referência a nós localizados à direita e a esquerda (da raiz ou do nó pai), e a variável *raiz*, que guardará o valor do nó raiz da árvore.

EXEMPLO 1: PSEUDOCÓDIGO QUE REPRESENTA UMA ÁRVORE BINÁRIA.

```
1.  algoritmo BArvore
2.      tipo apontador: ^no_arvore
3.      tipo no_arvore: registro
4.                          valor: inteiro
5.                          esq: apontador
6.                          dir: apontador
7.                      fim
8.      var
9.          raiz: apontador
10. Função inserir (arvore: no_arvore, novoNo: inteiro): no_arvore
11.     var
12.         apoio: no_arvore
13.     inicio
14.         Se (arvore = nulo) então
15.             apoio.valor ← novoNo
16.             retorne (apoio)
17.         Senão
18.             Se (novoNo < arvore.valor) então
19.                 arvore^.esq ← inserir(arvore^.esq, novoNo)
20.             Senão
21.                 arvore^.dir ← inserir(arvore^.dir, novoNo)
22.             Fim-se
23.         Fim-se
24.         retorne (arvore)
25.     Fim
26.
27. Procedimento inserirNo (novoValor: inteiro)
28.     inicio
29.         raiz ← inserir (raiz,novoValor)
30.     fim
31.
32. Procedimento exibir_esquerdo
33.     arv: no_arvore
34.     inicio
35.         arv ← raiz
36.         Se (arv <> nulo) então
37.             mostre(arv ^.valor)
38.             exibir_esquerdo(arv ^.esq)
39.         fim
40.
41. Procedimento exibir_direito
42.     arv: no_arvore
43.     inicio
44.         arv ← raiz
45.         inicio
46.             Se (arv <> nulo) então
47.                 mostre (arv^.valor)
48.                 exibir_direito (arv^.dir)
49.             fim
50.
```

```
51. Procedimento exibir_raiz( )
52.     inicio
53.         mostre (“Raiz”, raiz)
54.     fim
```

Os comentários serão feitos juntamente com os comentários do programa.

EXEMPLO 2: PSEUDOCÓDIGO PARA REPRESENTAR O PROCEDIMENTO DA EXCLUSÃO DE NÓS COM ÁRVORES BINÁRIAS.

```
1.  Procedimento excluirNo (item: inteiro)
2.  var
3.      tempNo: no_arvore
4.      pai: no_arvore
5.      filho: no_arvore
6.      temp: no_arvore
7.  inicio
8.      tempNo ← raiz
9.      pai ← nulo
10.     filho ← raiz
11.     Enquanto (tempNo < > nulo .e. tempNo^.valor < > item) faça
12.         pai ← tempNo
13.         Se(item < tempNo^.valor) então
14.             tempNo ← tempNo^.esq
15.         Senão
16.             tempNo ← tempNo^.dir
17.         Fim-se
18.         Se (tempNo = nulo) então
19.             Mostre (“item não localizado”)
20.         Fim-se
21.         Se (pai = nulo) então
22.             Se (tempNo^.dir = nulo)
23.                 raiz ← tempNo^.esq
24.             Fim-se
25.             Se (tempNo.esq = nulo)
26.                 raiz ← tempNo^.dir
27.             Fim-se
28.         Senão
29.             temp ← tempNo
30.             filho ← tempNo^.esq
31.             Enquanto (filho.dir < > nulo) faça
32.                 temp ← filho
33.                 filho ← filho^.dir
34.             Fim-enquanto
35.             Se (filho < > tempNo^.esq) então
36.                 temp^.dir ← filho.^esq
37.                 temp.^esq ← raiz.^esq
38.             Fim-se
39.             filho.^dir ← raiz.^dir
40.             raiz ← filho
41.         Fim-se
42.         Se (tempNo.^dir = nulo) então
43.             Se (pai.^esq = tempNo.^esq
44.                 pai.^esq ← tempNo.^esq
45.             Senão
46.                 pai.^dir ← tempNo.^esq
47.             Fim-se
```

```
48.         Senão
49.             Se (tempNo^.esq = tempNo) então
50.                 Se (pai.^esq = tempNo) então
51.                     pai^.esq ← tempNo^.dir
52.                 Senão
53.                     pai^.dir ← tempNo^.dir
54.                 fim-se
55.             Senão
56.                 temp ← tempNo
57.                 filho ← tempNo^.esq
58.                 Enquanto (filho.dir < > nulo) faça
59.                     temp ← filho
60.                     filho ← filho^.dir
61.                 Fim-enquanto
62.                 Se (filho < > tempNo.esq) então
63.                     temp.^dir ← filho.esq
64.                     filho.^esq ← tempNo.esq
65.                 Fim-se
66.                 filho.^dir ← tempNo.^dir
67.                 Se (pai^.esq = tempNo) então
68.                     pai.^esq ← filho
69.                 Senão
70.                     pai.^dir ← filho
71.                 fim-se
72.             Senão
73. fim
```