ESTRUTURA DE DADOS

Rafael Albuquerque

SAGAH
SOLUÇÕES EDUCACIONAIS INTEGRADAS

Caminhamento em árvore

Objetivos de aprendizagem

Ao final deste texto, você deve apresentar os seguintes aprendizados:

- Reconhecer o caminhamento pós-ordem.
- Aplicar o caminhamento pré-ordem.
- Usar o caminhamento *in*-ordem.

Introdução

Em uma estrutura de dados no estilo de árvore, seja ela binária ou *n*-ária, apresentar todos os seus elementos sempre presume que existirá um padrão a ser seguido para percorrê-la, o que independe de existir alguma ordem ou não de inserção dos elementos na árvore.

Dentre os modos de exibição, serão explorados os tipos de apresentação em três seguimentos: pré-ordem, *in*-ordem e pós-ordem. Essas três formas apresentam uma distinção entre formas de percursos, pela ordem em que os nós são visitados.

Neste capítulo, veremos como implementar essas três formas de caminhamento em Python 3.X, com uma forma mais simplificada de árvore, com poucos elementos com o tipo preestabelecido em inteiro, e os resultados do percurso ilustrados em imagens.

1 Caminhamento pós-ordem

O caminhamento em pós-ordem refere-se às formas de visitas, em que a raiz passa a ser o último elemento a ser observado. Dessa forma, os nós de uma árvore binária passam a percorrer tanto a subárvore à esquerda quanto à direita e tratam o nó raiz da árvore, de forma recursiva (CELES; CERQUEIRA; RANGEL, 2004).

Utilizando a árvore da Figura 1, vejamos como ficaria a ordem de apresentação dos elementos da árvore fazendo o percurso pós-ordem. Antes de verificar a resposta, tente responder: qual seria a ordem de impressão dos elementos, seguindo as regras acima e a intuição adquirida?

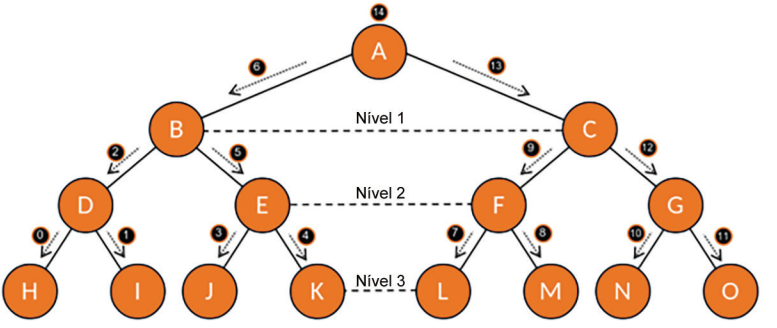

Figura 1. Caminhamento pós-ordem.

O resultado encontrado foi:

H-I-D-J-K-E-B-L-M-F-N-O-G-C-A

Note que, em relação aos modos de leitura ordem e *in*-ordem, a ordem da sequência numérica dos nós visitados retornou um valor bem diferente, no qual o nó raiz foi o último a ser visitado, simplesmente pelo fato de a raiz ser lida após os elementos das subárvores à esquerda e à direita. A seguir, veremos como implementar a função/método para o tipo de impressão pós-ordem, dando continuidade às funções já implementadas.

Implementação pós-ordem

A próxima função a ser implementada é a de pós-ordem, na qual tem seu valor raiz como o último valor a ser impresso, dado uma estrutura de dados em árvore. A seguir, na Figura 2, é apresentada a implementação de sua função, dando continuidade às funções já apresentadas neste capítulo.

```
def pos_order(atual):
    if atual != None:
        # explora a subarvore a
        # esquerda recursivamente
        pos_order(atual.esq)
        # explora a subarvore a
        # direita recursivamente
        pos_order(atual.dir)
        print(atual.item,end=" ")
```

Figura 2. Função de impressão da árvore pós-ordem.

O que mudou em relação à função de apresentação *in*-ordem? Note que o elemento só será impresso após ocorrer uma visita a cada nó de cada subárvore, e após identificar que não existe nenhum elemento tanto para a subárvore à esquerda quanto para à direita, é que o valor do nó observado será impresso. Por fim, na Figura 3, apresentamos o teste da função criada, e antecedendo ao seu resultado, a impressão da árvore no formato 2D.

```
if __name__ == '__main__':
    arvore = NoArvore("A")
    arvore.esq = NoArvore("B")
    arvore.dir = NoArvore("C")

    arvore.esq.esq = NoArvore("D")
    arvore.esq.dir = NoArvore("E")
    arvore.dir.esq = NoArvore("F")
    arvore.dir.dir = NoArvore("G")

    arvore.esq.esq.esq = NoArvore("H")
    arvore.esq.esq.dir = NoArvore("I")
    arvore.esq.dir.esq = NoArvore("J")
    arvore.esq.dir.dir = NoArvore("K")
    arvore.dir.esq.esq = NoArvore("L")
    arvore.dir.esq.dir = NoArvore("M")
    arvore.dir.dir.esq = NoArvore("N")
    arvore.dir.dir.dir = NoArvore("O")

    # imprime arvore da direita pra esquerda
    print("Imprime a árvore: \n")
    imprimir_arvore(arvore,0)
    print("\n")
    print("pós ordem: ")
    pos_order(arvore)
    print("\n")
```

```
Imprime a árvore:

                    O
                G   N
            C       M
                F   L
A
                    K
                E   J
            B
                D   I
                    H

pós ordem:
H I D J K E B L M F N O G C A
> >
```

Figura 3. Adição de elementos à arvore e resultados pós-ordem.

2 Caminhamento pré-ordem

Caminhamentos em árvores, conforme Tenenbaum, Langsam e Augenstein (1995), são formas de percorrê-las, enumerando cada um de seus nós uma vez. A enumeração ocorre de acordo com a necessidade, podendo o programador processar essa visita ao nó ou ao elemento da árvore como bem necessitar. Como visto em sequências lineares (p. ex., listas, pilhas, filas), cuja visita a cada elemento é realizada um a um, iniciando, de modo geral, no primeiro elemento, iterando até o último.

No entanto, tratando-se de árvores, não existe um caminho natural a se seguir, uma vez que existem diferentes formas para se percorrer uma árvore. Logo, podemos inferir que existem diferentes tipos de ordenamento de percurso em diferentes casos. Evidentemente, uma árvore somente será percorrida caso haja elementos contidos nela. Em árvores, o modo mais eficiente de percorrer os seus elementos é de forma recursiva, visto que percorrer uma árvore envolve visitar sua raiz e percorrer suas subárvores à esquerda e à direita (TENENBAUM; LANGSAM; AUGENSTEIN, 1995).

Conforme Piva Junior *et al.* (2014), para visitar uma árvore binária no percurso em pré-ordem, os nós são visitados em forma recursiva, na seguinte ordem:

- apresenta o item do nó visitado, nomeado, muitas vezes, de raiz;
- passa para o elemento do nó filho ou da subárvore, ambos à esquerda;
- passa para o elemento do nó filho ou da subárvore, ambos à direita.

Na Figura 4, é apresentada uma estrutura de árvore binária com 15 elementos. Seguindo as orientações de como realizar a leitura em pré-ordem da árvore, teremos a seguinte sequência:

A-B-D-H-I-E-J-K-C-F-L-M-G-N-O

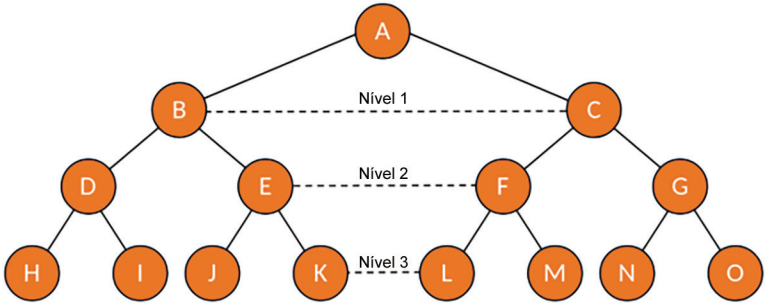

Figura 4. Exemplo de estrutura de árvore binária.

Aplicando as regras desse modelo de leitura, foi possível observar o porquê de termos obtido a sequência acima de letras? Perceba que, pelas regras de leitura, o primeiro elemento a ser visualizado será a raiz, logo após, o elemento à esquerda, e, por fim, o elemento à direita.

Nesse caso, o primeiro elemento a ser visualizado foi o elemento (A), localizado no nível 0 da árvore. Como o próximo elemento a ser visto estará localizado à esquerda, caso exista alguma subárvore, o elemento a ser visto será o (B). Até aqui, atendemos as duas primeiras regras de percurso em pré-ordem. Visto que o percurso é realizado em modo recursivo, será realizada uma próxima visita à subárvore à esquerda, no nível abaixo (nível 2). Perceba que, seguindo essa orientação, o próximo elemento a ser visualizado será o elemento (D). Este elemento contém duas ramificações, uma para o lado esquerdo e outra para o lado direito. Como o elemento (D) é visto como um nó raiz para essas ramificações, o próximo elemento a ser visualizado será o (H) – nível 3. E agora, qual será o próximo elemento a ser visualizado? Perceba que o elemento (H) não contém ramificações. Logo, teremos que visualizar a próxima ramificação de (D), localizada à sua direita do nível 3, o elemento (I).

Após visitarmos os elementos à esquerda de (D), vamos visitar a sua próxima subárvore, que está à direita. O primeiro elemento da subárvore à direita de (D) é o elemento (E). Como o nó (E) contém mais duas ramificações, visitamos primeiro a ramificação à esquerda (J) e, então, a ramificação (K). Terminamos de visitar todos os nós da subárvore à esquerda do nó raiz (A). Agora, precisamos visitar a subárvore à direita do nó raiz (A). A Figura 5 apresenta o percurso percorrido indicado pelo caminho de setas.

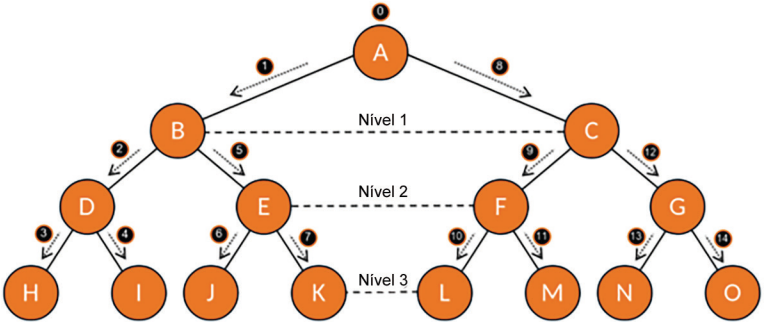

Figura 5. Caminhamento em pré-ordem.

Implementação do percurso pré-ordem

Para uma melhor compreensão do modo de leitura em pré-ordem, vamos apresentar uma implementação simples de árvore com alguns elementos em suas subárvores, tanto à esquerda como à direita, para, então, podermos imprimir seus elementos na ordem indicada.

A Figura 6 apresenta o código para a criação da árvore, assim como os métodos de impressão da rede em um formato 2D, cujo nome do método é `imprimir_arvore`, imprimindo a partir do nó raiz e expandindo as suas ramificações para a direita da tela. Além disso, é apresentado o código de impressão em pré-ordem.

```
class NoArvore(object):
  def __init__(self, item):
    self.item = item
    self.esq = None
    self.dir = None
```

```
def pre_ordem(atual):
  if atual != None:
    print(atual.item,end=" ")
    # explora a subarvore a
    #esquerda recursivamente
    pre_ordem(atual.esq)
    # explora a subarvore a
    # direita recursivamente
    pre_ordem(atual.dir)
```

```
def imprimir_arvore(raiz, espaco):
  # Caso base
  if (raiz == None) :
    return
  # Aumenta a distancia entre os niveis em 10
  espaco += 10

  # Processa a subarvore à direita
  imprimir_arvore(raiz.dir, espaco)

  # Imprime o nó atual após o espaço,
  #por meio da chave end
  print(end = " "*(espaco - 10))
  print(raiz.item)
  # Processa a subarvore  a esquerda
  imprimir_arvore(raiz.esq, espaco)
```

Figura 6. Métodos para criação e impressão da árvore.

No entanto, a Figura 7 apresenta uma árvore de forma simplificada, na qual são inseridos de forma manual os elementos, como os apresentados nas Figuras 1 e 2. Ainda na mesma imagem, é possível visualizar os resultados tanto da impressão dos elementos da árvore em formato de árvore como a sua ordem em pré-ordem.

Figura 7. Adição de elementos à arvore e resultados em pré-ordem.

Fique atento

Observe o modo de impressão para distinguir os tipos de ordenações impressas e perceba que cada formato de impressão está relacionado à posição do nó raiz.

Por exemplo: para o modo de impressão em pré-ordem, o primeiro elemento a ser impresso é o pertencente ao nó raiz, seguido dos elementos da subárvore à esquerda e à direita. Todavia, para a impressão *in*-ordem, o nó raiz é o segundo elemento a ser impresso, de acordo com a seguinte sequência: subárvore à esquerda, raiz e subárvore à direita; por último, a impressão pós-ordem mantém a seguinte sequência: subárvore à esquerda, subárvore à direita e raiz. Dessa forma, fica fácil relacionar cada tipo de impressão da árvore. Lembre-se sempre da posição do nó raiz.

3 Caminhamento *in*-ordem

O caminhamento *in*-ordem tem as mesmas características do modo de impressão visto na seção de pré-ordem, exceto pelo fato do nó raiz ser visitado após a visita na subárvore à esquerda, e anterior à visita na subárvore à direita. Nessa forma de percurso, conforme Tenenbaum, Langsam e Augenstein (1995), para percorrer uma árvore binária não vazia é preciso:

- percorrer a subárvore à esquerda em ordem simétrica;
- visitar o nó raiz;
- percorrer a subárvore à direita em ordem simétrica.

Para ilustrarmos esse caminho, voltamos para o exemplo apresentado na Figura 4, no qual temos uma sequência de caracteres organizada em formato de árvore, na qual precisaremos realizar uma nova leitura, não obstante, visitando conforme a sequência aqui estabelecida. Para simplificar essa leitura, a Figura 8 apresenta a sequência de nós visitados. Antes de visualizar, você consegue pensar qual será essa sequência, sem observar a imagem a seguir?

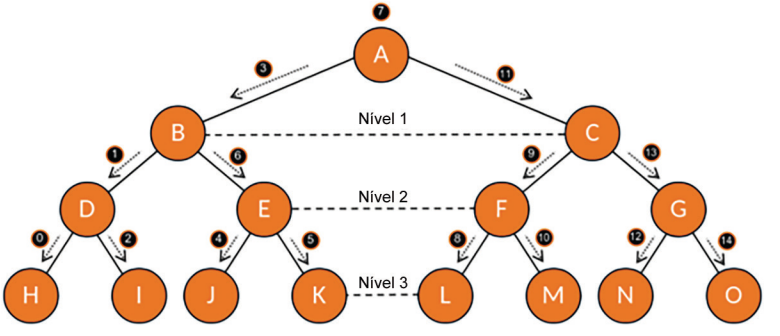

Figura 8. Caminhamento *in*-ordem.

O resultado encontrado foi:

H-D-I-B-J-E-K-A-L-F-M-C-N-G-O

Note que, em relação ao modo de leitura pré-ordem, a ordem da sequência numérica dos nós visitados foi alterada simplesmente pelo fato de a raiz ser lida após os elementos da subárvore à esquerda. A seguir, veremos como implementar a função/método para esse tipo de impressão, dando continuidade às funções já implementadas.

Implementação da função *in*-ordem

Aproveitando boa parte da implementação da árvore vista na Figura 6, vamos apenas adicionar a função de impressão da árvore *in*-ordem ao código de árvore já implementado. Para isso, a Figura 9 apresenta a forma correta de sua implementação.

```
def em_ordem(atual):
    # enquanto um nó não for nulo
    # mandar uma referência nula
    if atual != None:
        em_ordem(atual.esq)
        print(atual.item, end=" ")
        em_ordem(atual.dir)
```

Figura 9. Função de impressão da árvore *in*-ordem.

Perceba que na função implementada na Figura 9, antes da verificação de valor nulo, não existe nenhum código, diferentemente da função de pré-ordem apresentada na Figura 6. Por fim, na Figura 10, apresentamos o teste da função criada e, antecedendo ao seu resultado, a impressão da árvore em 2D.

```
if __name__ == '__main__':
  arvore = NoArvore("A")
  arvore.esq = NoArvore("B")
  arvore.dir = NoArvore("C")

  arvore.esq.esq = NoArvore("D")
  arvore.esq.dir = NoArvore("E")
  arvore.dir.esq = NoArvore("F")
  arvore.dir.dir = NoArvore("G")

  arvore.esq.esq.esq = NoArvore("H")
  arvore.esq.esq.dir = NoArvore("I")
  arvore.esq.dir.esq = NoArvore("J")
  arvore.esq.dir.dir = NoArvore("K")
  arvore.dir.esq.esq = NoArvore("L")
  arvore.dir.esq.dir = NoArvore("M")
  arvore.dir.dir.esq = NoArvore("N")
  arvore.dir.dir.dir = NoArvore("O")

  # imprime arvore da direita pra esquerda
  print("Imprime a árvore: \n")
  imprimir_arvore(arvore,0)
  print("\n")
  print("em ordem: ")
  em_ordem(arvore)
  print("\n")
```

```
Imprime a árvore:

                    O
               G
                    N
          C
                    M
               F
                    L
A
                    K
               E
                    J
          B
                    I
               D
                    H

em ordem:
H D I B J E K A L F M C N G O

>
```

Figura 10. Adição de elementos à arvore e resultados *in*-ordem.

Saiba mais

Os caminhamentos aqui vistos fazem parte de um seleto grupo de métodos de busca por profundidade, diferentemente das buscas por extensão. Ademais, as árvores são estruturas muito utilizadas em controle de diretórios, como calculadoras no cálculo de expressões aritméticas, etc. Portanto, existem vários tipos com várias finalidades.

Exemplo

Perceba que a ordem em que se percorre uma árvore pode fazer total diferença, dependendo do tipo de aplicação. Em uma expressão aritmética, por exemplo: 5 * 5 + 5, se considerarmos que as operações pertencem às raízes das subárvores, uma leitura em pré-ordem poderia retornar uma sequência de: *+555, enquanto uma leitura em pós-ordem retornaria: 555*+. Portanto, uma leitura simétrica se apresentaria, neste caso hipotético, como a melhor forma de leitura.

Referências

CELES, W.; CERQUEIRA, R.; RANGEL, J. L. *Introdução e estrutura de dados*: com técnicas de programação em C. 4. ed. Rio de Janeiro: Campus, 2004.

PIVA JUNIOR, D. *et al. Estrutura de dados e técnicas de programação*. Rio de Janeiro: Elsevier, 2014.

TENENBAUM, A. M.; LANGSAM, Y.; AUGENSTEIN, M. J. *Estruturas de dados em C*. São Paulo: MAKRON Books, 1995.

Encerra aqui o trecho do livro disponibilizado para esta Unidade de Aprendizagem. Na Biblioteca Virtual da Instituição, você encontra a obra na íntegra.

Conteúdo:

SAGAH
SOLUÇÕES EDUCACIONAIS INTEGRADAS