PERCURSOS EM ÁRVORES

Percursos em árvores: formalização
Principais formas de percurso de uma árvore

Um percurso é a visita sistemática aos elementos de uma estrutura de dados. Vale destacar que visitar significa realizar uma operação e não acessar o conteúdo armazenado no elemento da estrutura de dados. Ou seja, especificamente para árvores, acessar um nó da árvore não configura uma visita. A visita é uma operação que tem sentido na semântica da aplicação desejada. A visita mais simples que podemos definir é a impressão do rótulo da chave armazenada no nó visitado.

Observe que o conceito de árvore é recursivo, isto é, a estrutura foi definida em termos recursivos. Por tal razão, as definições dos percursos também são recursivas. Existem três maneiras clássicas de percursos em árvores:

1. **Pré-ordem**

   Visitar, primeiro, a raiz, depois a subárvore esquerda e, por último, a subárvore direita.

2. **Em ordem**

   Visitar, primeiro, a subárvore esquerda, depois a raiz e, por último, a subárvore direita.

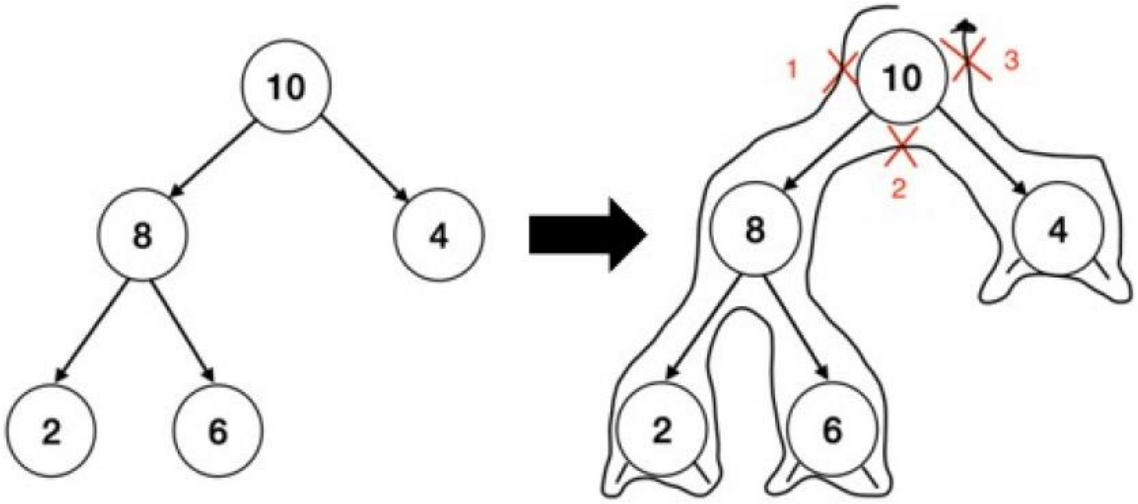
3. **Pós-ordem**

   Visitar, primeiro, a subárvore esquerda, depois a subárvore direita e, por último, a raiz.

Percurso pré-ordem

A partir da raiz r da árvore T, percorre-se a árvore da seguinte forma:

1. Visita-se a raiz.
2. Percorre-se a subárvore esquerda de T, em pré-ordem.
3. Percorre-se a subárvore direita de T, em pré-ordem.



Considerando como a operação de visita a impressão do rótulo contido no nó visitado, veremos o resultado do percurso em pré-ordem da árvore na imagem 7. Partindo-se da raiz, nó de rótulo “10”, temos que a primeira operação é a visita do nó, isto é, a impressão do seu rótulo.

Impressão: **10**

Em seguida, visita-se recursivamente a subárvore esquerda, cuja raiz é “8”, e a primeira operação é a visita do nó com rótulo “8”.

Impressão: **10, 8**

Após a visita do nó “8”, percorre-se recursivamente a subárvore esquerda do nó “8”, cuja raiz é o nó com rótulo “2”. A primeira operação desse percurso é a visita, isto é, o rótulo “2” é impresso.

Impressão: **10, 8, 2**

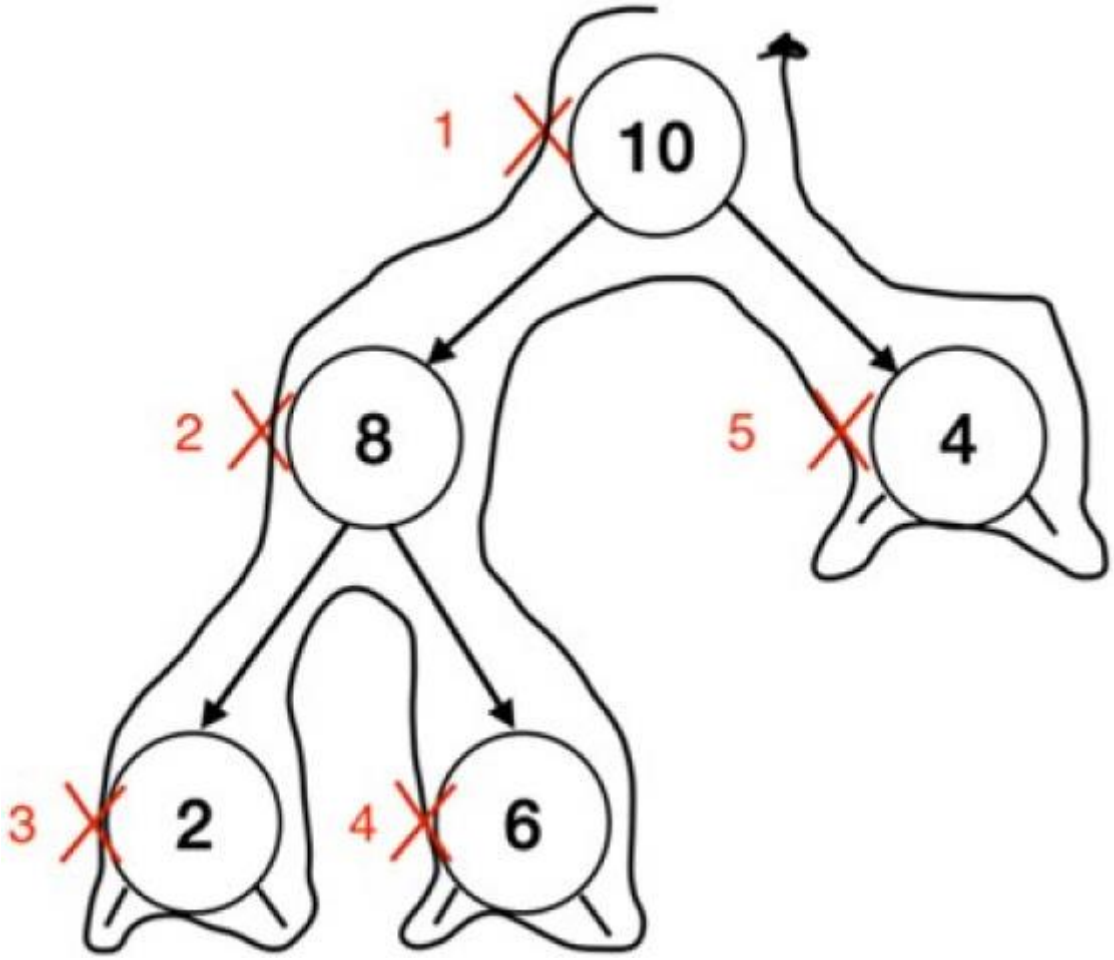
O nó “2” é folha, isto é, não possui subárvores, assim, o percurso da subárvore esquerda do nó “8” é concluído. O próximo passo é percorrer a subárvore direita do nó “8”, cuja raiz é o nó “6”. O primeiro passo é a impressão (visita).

Impressão: **10, 8, 2, 6**

Observe que o nó “6” é folha, o que encerra o percurso da subárvore de raiz “6”. Assim, retornamos ao seu pai, o nó “8”, que é raiz da subárvore, e que já teve seu percurso completo. O pai de “8” é o nó “10”, que já foi visitado e teve sua subárvore esquerda visitada. O próximo passo é visitar sua subárvores direita em pré-ordem. A raiz da subárvore direita é “4”, que é folha, sendo assim, o percurso pré-ordem da árvore é:

Impressão: **10, 8, 2, 6, 4**

Observe que, para cada nó, acessamos seu conteúdo três vezes, porém, somente uma dessas corresponde à visita. No caso do percurso em pré-ordem, a primeira vez. Observe também que, imprimindo o conteúdo do nó (operação de visita), sempre no acesso “1”, teremos o percurso em pré-ordem.



Percurso em ordem

A partir da raiz r da árvore T, percorre-se a árvore da seguinte forma:

1. **Percorre-se a subárvore esquerda de T em ordem simétrica.**
2. **Visita-se a raiz.**
3. **Percorre-se a subárvore direita de T em ordem simétrica.**

Aplicando a definição na árvore da imagem 7, temos a seguinte sequência de visitas, que percorre-se a subárvore direita de T, em ordem simétrica, pode ser obtida, passo a passo, a partir do nó “10”, raiz de T.

Inicialmente, percorremos a subárvore esquerda de “10” em ordem simétrica, em que o nó “8” é a raiz dessa árvore. Mais uma vez, o primeiro passo é percorrer a subárvore esquerda do nó “8” em ordem simétrica.

Mais uma vez, o primeiro passo é percorrer a subárvore esquerda do nó “8” em ordem simétrica.

O nó “2” é a raiz dessa árvore; “2” é folha, assim, não possui subárvore esquerda nem direita. Logo, “2” é visitado após a tentativa de percurso de sua subárvore esquerda.

Impressão: **2**

Após a visita do nó “2”, seria feita a visita do ramo direito de “2”, que não existe, concluindo, assim, o percurso em ordem simétrica da subárvore de raiz “2”. O próximo passo é a visita do nó “8”.

Impressão: **2, 8**

Após a visita de “8”, percorre-se a subárvore direita de “8” em ordem simétrica. A raiz dessa subárvore é “6”, que é folha; logo, ocorre a visita de “6”, resultando na sequência de impressão.

Impressão: **2, 8, 6**

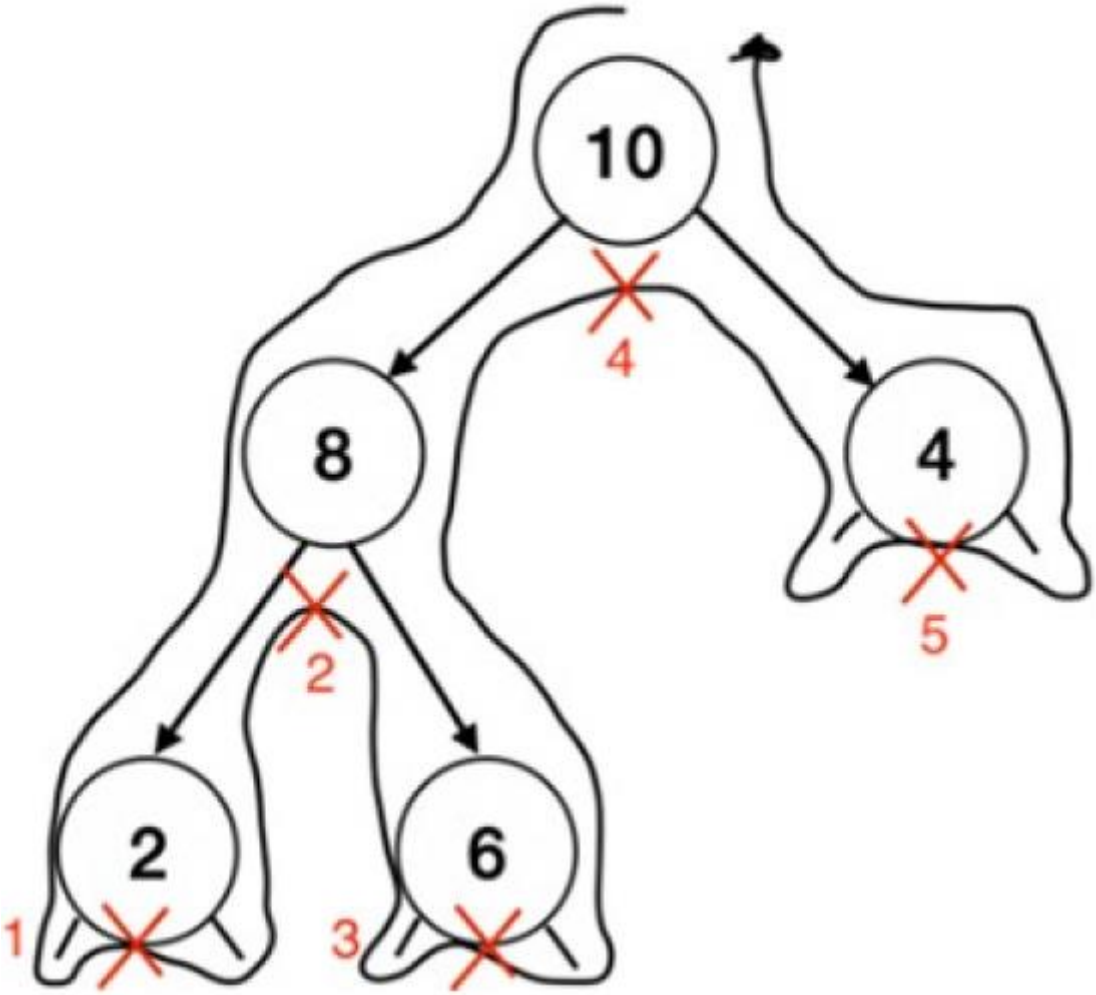
A impressão do rótulo da folha “6” e a tentativa de percurso da subárvore direita do nó “6” termina o percurso da subárvore de raiz “8”. Voltando, na reclusão, ao seu pai, já percorremos a subárvore esquerda do nó “10”. O próximo passo é a vista do nó, resultando na impressão.

Impressão: **2, 8, 6, 10**

O próximo passo é o percurso da subárvore direita de “10”, que contém a folha “4”, resultando na impressão.

Impressão: **2, 8, 6, 10, 4**

Concluindo o percurso em ordem simétrica de *T*, temos o seguinte:



Percurso pós-ordem

A partir da raiz *r* da árvore *T*, percorre-se a árvore da seguinte forma:

1. Percorre-se a subárvore esquerda de *T* em pós-ordem.
2. Percorre-se a subárvore direita de *T* em pós-ordem.
3. Visita-se a raiz.

Aplicando-se a definição na árvore da imagem 7, temos a sequência de visitas apresentada a seguir. Partindo-se da raiz “10” de T, percorre-se a subárvore esquerda de “10” em pós-ordem. O nó “8” é a raiz desta subárvore. Para esta subárvore, o primeiro passo é visitar seu ramo esquerdo em pós-ordem. O nó “2” é a raiz deste ramo, que também é folha, assim, “2” não tem subárvore esquerda e direita, sendo o primeiro nó visitado.

Impressão: **2**

Retorna-se, então, ao seu pai, o nó “8”, a subárvore esquerda de “8” já foi visitada, o próximo passo é visitar sua subárvore direita. O nó “6” é raiz desta subárvore e folha, sendo assim, o nó “6” é visitado.

Impressão: **2, 6**

Com o término do percurso na subárvore de raiz “6”, retorna-se ao pai de “6”, que é o nó “8”. As subárvores esquerda e direita de “8” já foram percorridas, assim, o próximo passo é visitar “8”.

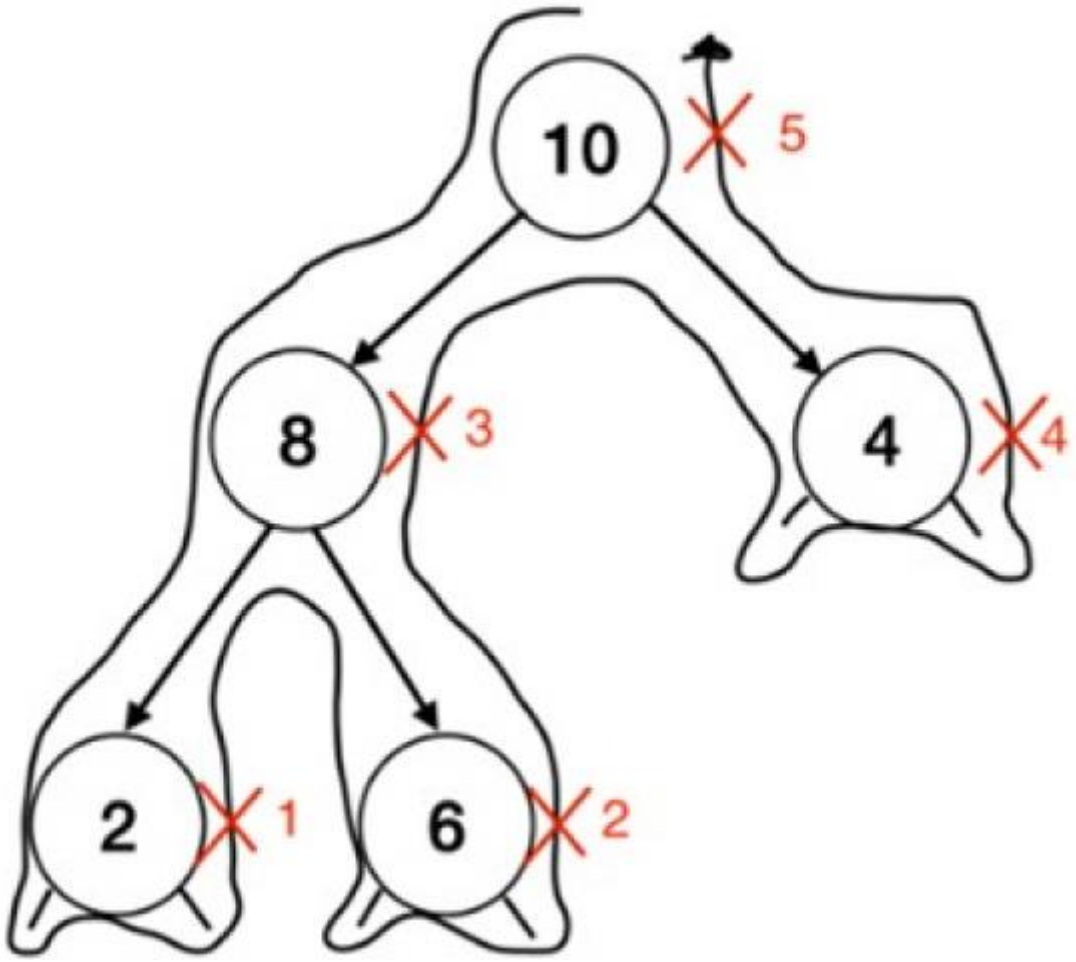
Impressão: **2, 6, 8**

A visita ao nó “8” termina o percurso da subárvores de raiz “8”. Ao retornar ao pai do nó “8”, o nó “10”, devemos percorrer em pós-ordem a subárvore direita do nó “10”. O nó “4” é a raiz da subárvore é folha, sendo assim, visita-se o nó “4”.

Impressão: **2, 6, 8, 4**

Com o término do percurso da subárvore de raiz “4”, retorna-se ao pai de “4”, que é o nó “10”. Já foram percorridas as subárvores esquerda e direita de “10”, assim, o próximo passo é visitar o nó “10”, o que finaliza o percurso.

Impressão: **2, 6, 8, 4, 10**, que é o percurso em pós-ordem.



Algoritmos em Python para percursos em árvores

Algoritmo de percurso pré-ordem

Descrição:

```
função pre-ordem (registro no *p)
inicio
  visita (p);
  se (p->dir != NULO)
     pre-ordem (p->dir);
  se (p->esq != NULO)
     pre-ordem (p->esq);
fim
```

Pseudocódigo:

```
def VisitaPreOrdem(raiz):
   if (raiz):
```

```
    print(raiz.chave)

    VisitaPreOrdem(raiz.esquerda)

    VisitaPreOrdem(raiz.direita)
```

Para realizar o percurso em pré-ordem, são necessários três acessos ao nó. No caso da pré-ordem, no primeiro, executamos a visita; no segundo, chamamos recursivamente o algoritmo para a subárvore esquerda e, no terceiro, ocorre a chamada do percurso em pré-ordem do ramo direito. Assim, a complexidade computacional do percurso em pré-ordem é **O**(**n**).

Algoritmo de percurso em ordem

Descrição:

```
função simetrica (registro no *p)
inicio
  se (p->dir != NULO)
     simetrica (p->dir);
  visita (p);
  se (p->esq != NULO)
     simetrica (p->esq);
fim
```

Pseudocódigo:

```
def VisitaInOrdem(raiz):

   if (raiz):

     VisitaInOrdem(raiz.esquerda)

     print(raiz.chave)

     VisitaInOrdem(raiz.direita)
```

A análise de complexidade é análoga à realizada no algoritmo do percurso em pré-ordem. Observe que a única diferença é a ordem das visitas. Sendo assim, a complexidade computacional do algoritmo para percurso em ordem simétrica é ***O(n)*.**

Algoritmo de percurso pós-ordem

Descrição:

```
função pos-ordem (registro no *p)

inicio

  se (p->dir != NULO)

     pos-ordem (p->dir);

  se (p->esq != NULO)

     pos-ordem (p->esq);

  visita (p);

fim
```

Pseudocódigo:

```
def VisitaPosOrdem(raiz):
  if (raiz):
    VisitaPosOrdem(raiz.esquerda)
    VisitaPosOrdem(raiz.direita)
    print(raiz.chave)
```

A análise da complexidade é totalmente análoga à análise feita para pré-ordem e ordem simétrica, o que faz com que o algoritmo tenha complexidade ***O(n)*.**