

Estrutura de Dados Avançada

Profa. Tatiane Fernandes Figueiredo



UNIVERSIDADE
FEDERAL DO CEARÁ

Percurso em Árvores Binárias

- Formas de percorrer árvores binárias?

Percurso em Árvores Binárias

- Formas de percorrer árvores binárias?
 - In ordem
 - Pós ordem
 - Pré ordem

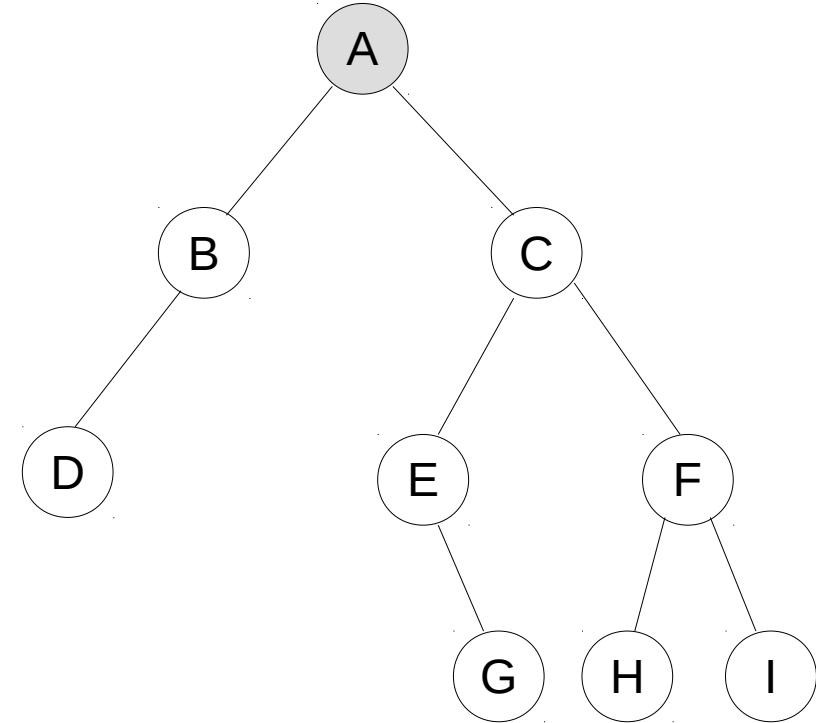
In-ordem

procedimento *in*(*pt*)

se *pt.esq* \neq *vazio* **então** *in*(*pt.esq*)

visita(*pt*)

se *pt.dir* \neq *vazio* **então** *in*(*pt.dir*)



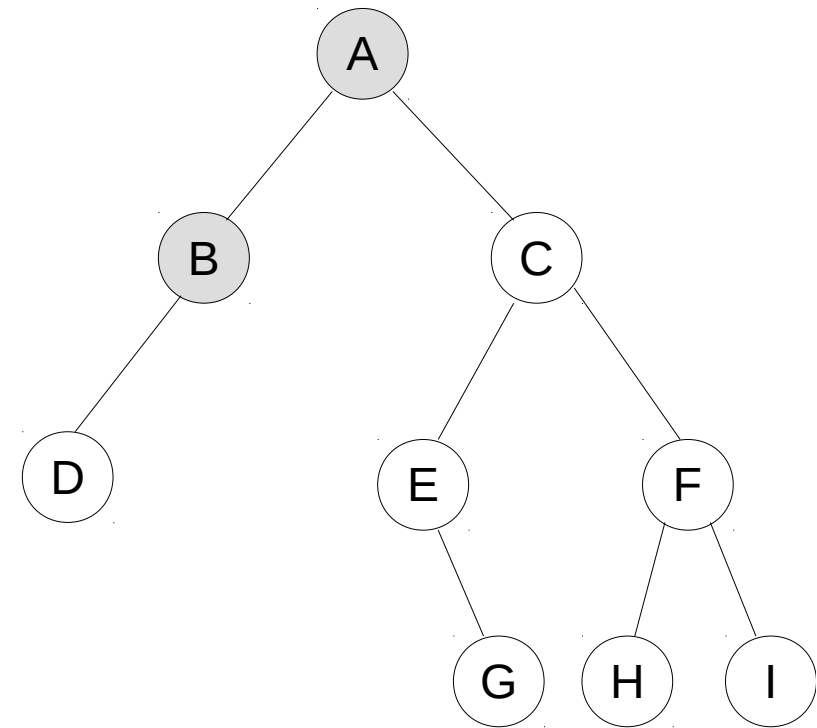
In-ordem

procedimento *in*(*pt*)

se *pt.esq* \neq *vazio* **então** *in*(*pt.esq*)

visita(*pt*)

se *pt.dir* \neq *vazio* **então** *in*(*pt.dir*)



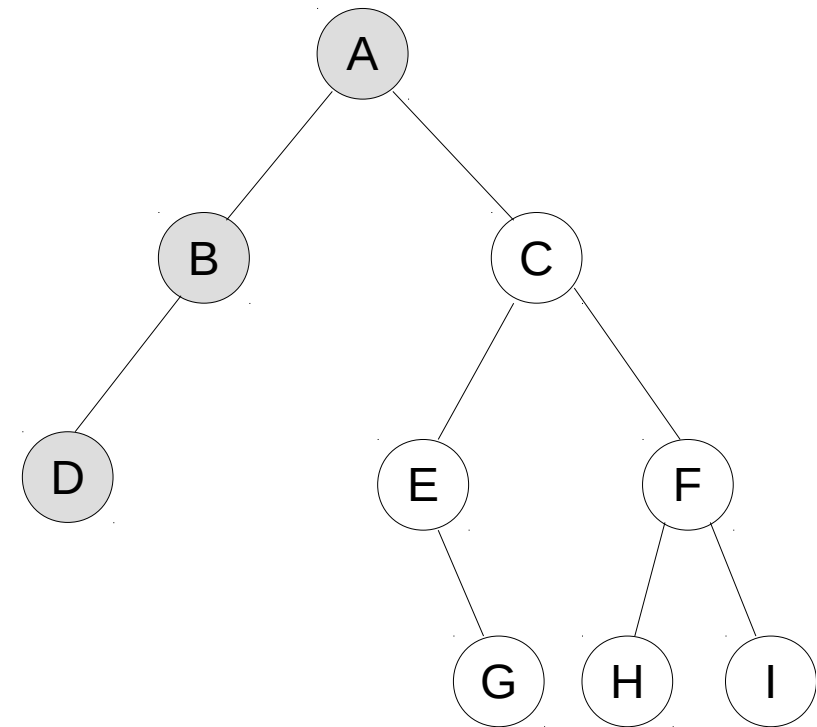
In-ordem

procedimento *in*(*pt*)

se *pt.esq* \neq *vazio* **então** *in*(*pt.esq*)

visita(*pt*)

se *pt.dir* \neq *vazio* **então** *in*(*pt.dir*)



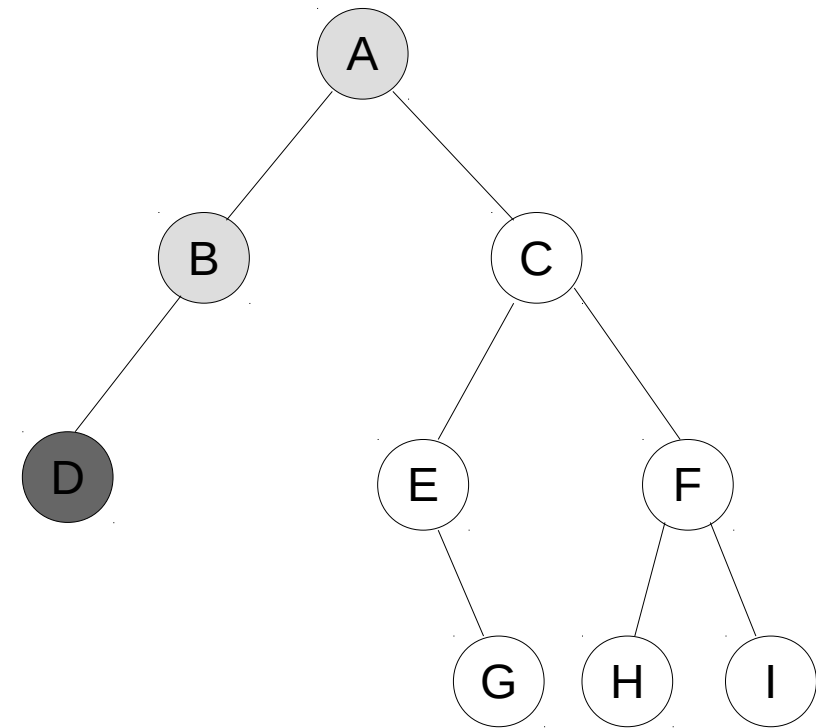
In-ordem

procedimento *in*(*pt*)

se *pt.esq* \neq *vazio* **então** *in*(*pt.esq*)

visita(*pt*)

se *pt.dir* \neq *vazio* **então** *in*(*pt.dir*)



D

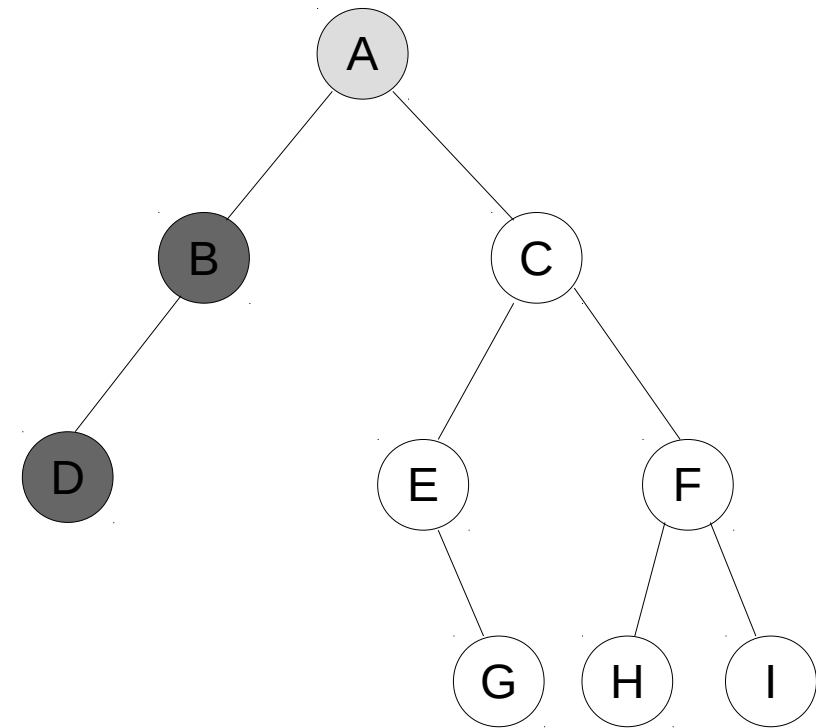
In-ordem

procedimento *in(pt)*

se *pt.esq* \neq *vazio* **então** *in(pt.esq)*

visita(pt)

se *pt.dir* \neq *vazio* **então** *in(pt.dir)*



D B

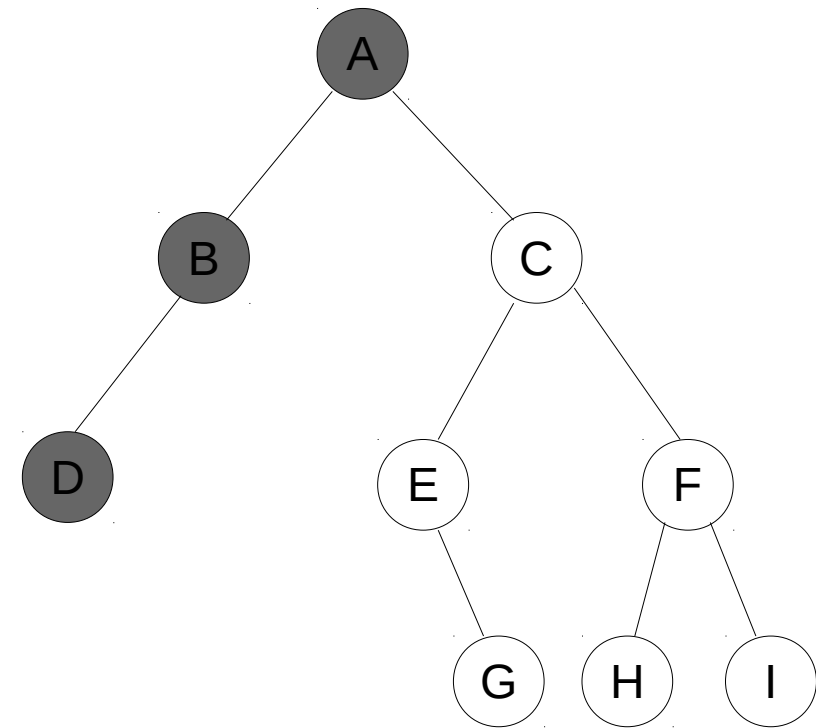
In-ordem

procedimento *in(pt)*

se *pt.esq* \neq *vazio* **então** *in(pt.esq)*

visita(pt)

se *pt.dir* \neq *vazio* **então** *in(pt.dir)*



D B A

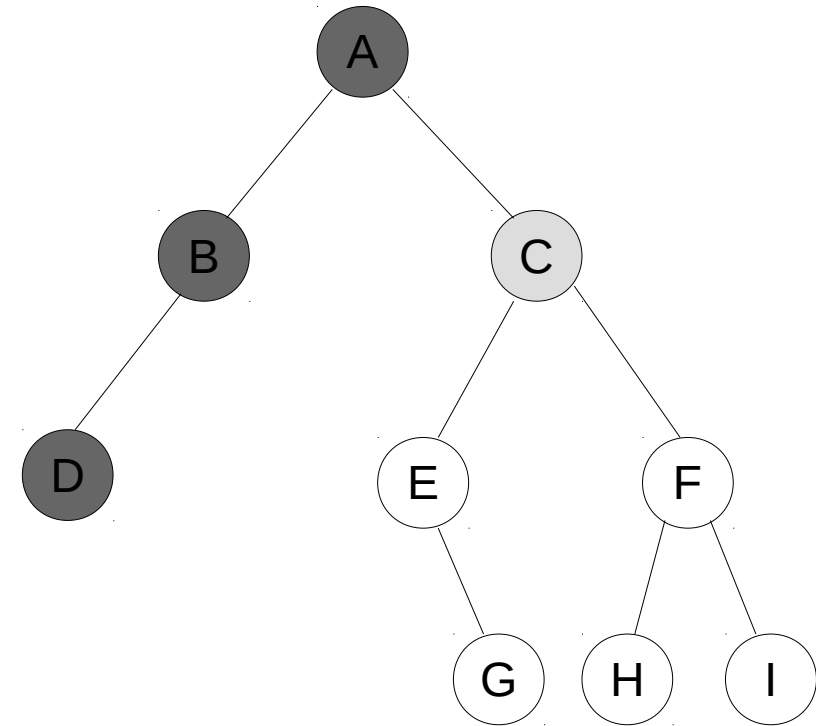
In-ordem

procedimento *in(pt)*

se *pt.esq* \neq *vazio* **então** *in(pt.esq)*

visita(pt)

se *pt.dir* \neq *vazio* **então** *in(pt.dir)*



D B A

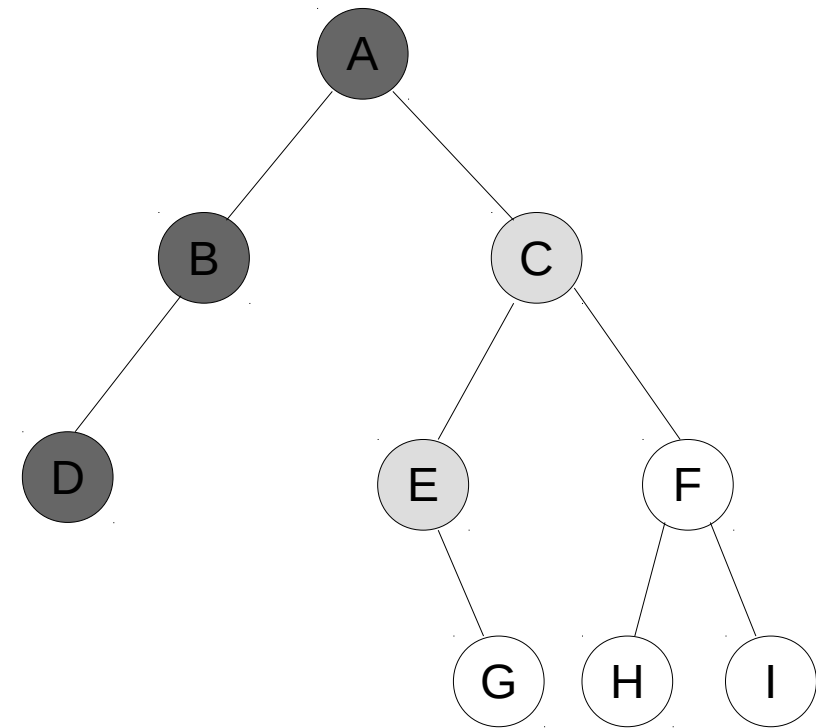
In-ordem

procedimento *in(pt)*

se *pt.esq* \neq *vazio* **então** *in(pt.esq)*

visita(pt)

se *pt.dir* \neq *vazio* **então** *in(pt.dir)*



D B A

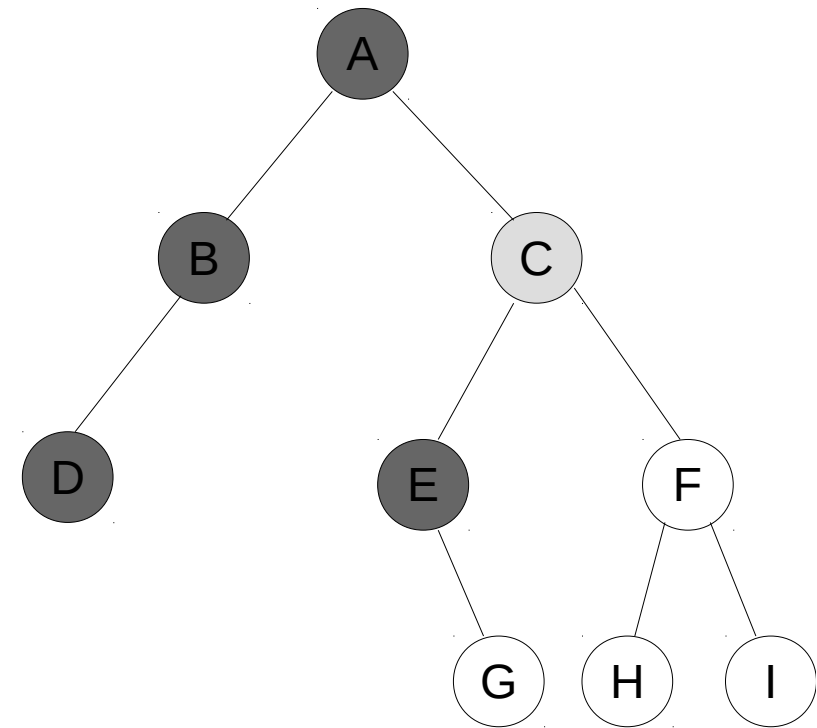
In-ordem

procedimento *in(pt)*

se *pt.esq* \neq *vazio* **então** *in(pt.esq)*

visita(pt)

se *pt.dir* \neq *vazio* **então** *in(pt.dir)*



D B A E

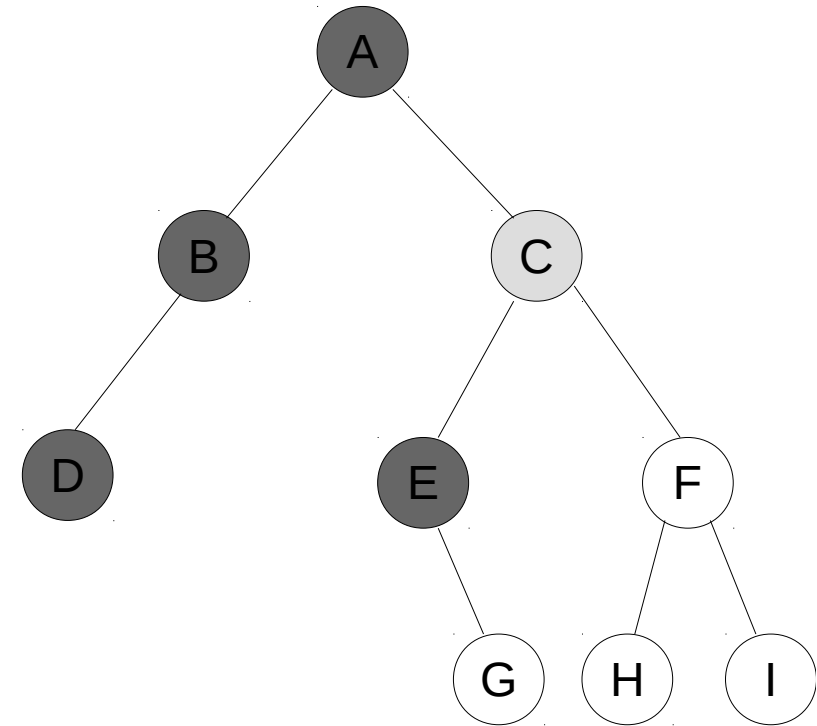
In-ordem

procedimento *in*(*pt*)

se *pt.esq* \neq *vazio* **então** *in*(*pt.esq*)

visita(*pt*)

se *pt.dir* \neq *vazio* **então** *in*(*pt.dir*)



D B A E

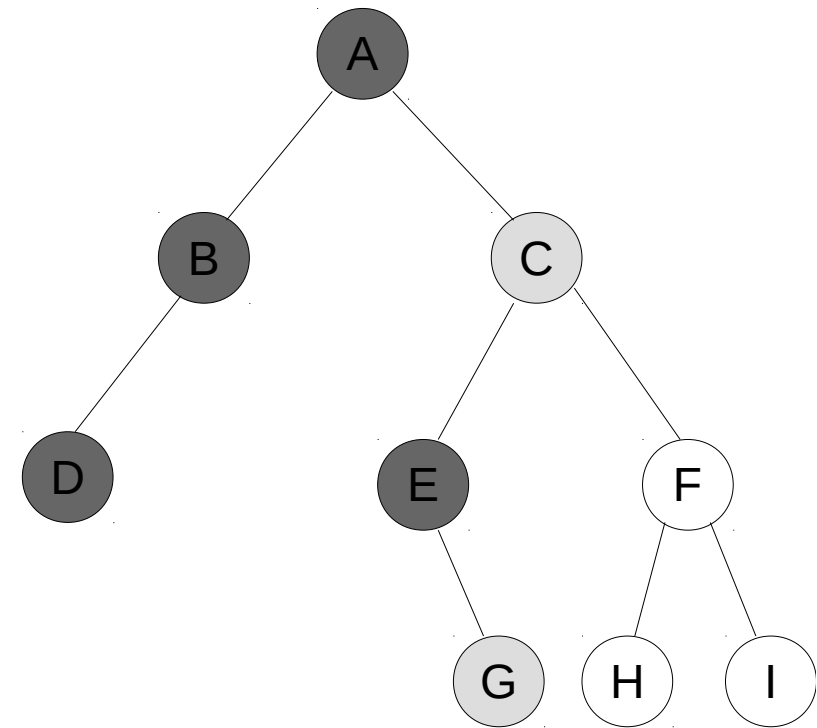
In-ordem

procedimento *in*(*pt*)

se *pt.esq* \neq *vazio* **então** *in*(*pt.esq*)

visita(*pt*)

se *pt.dir* \neq *vazio* **então** *in*(*pt.dir*)



D B A E

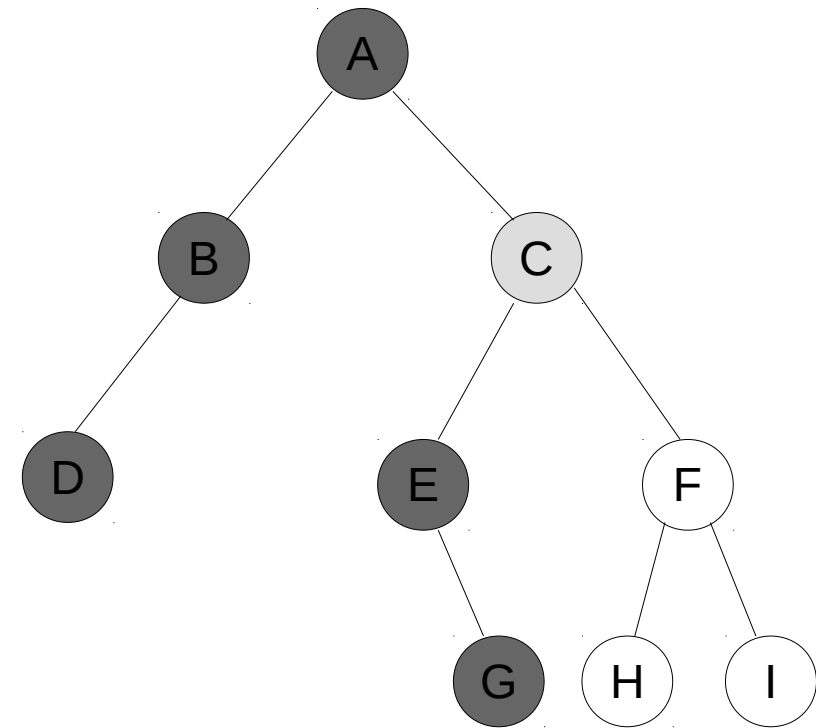
In-ordem

procedimento *in*(*pt*)

se *pt.esq* \neq *vazio* **então** *in*(*pt.esq*)

visita(*pt*)

se *pt.dir* \neq *vazio* **então** *in*(*pt.dir*)



D B A E G

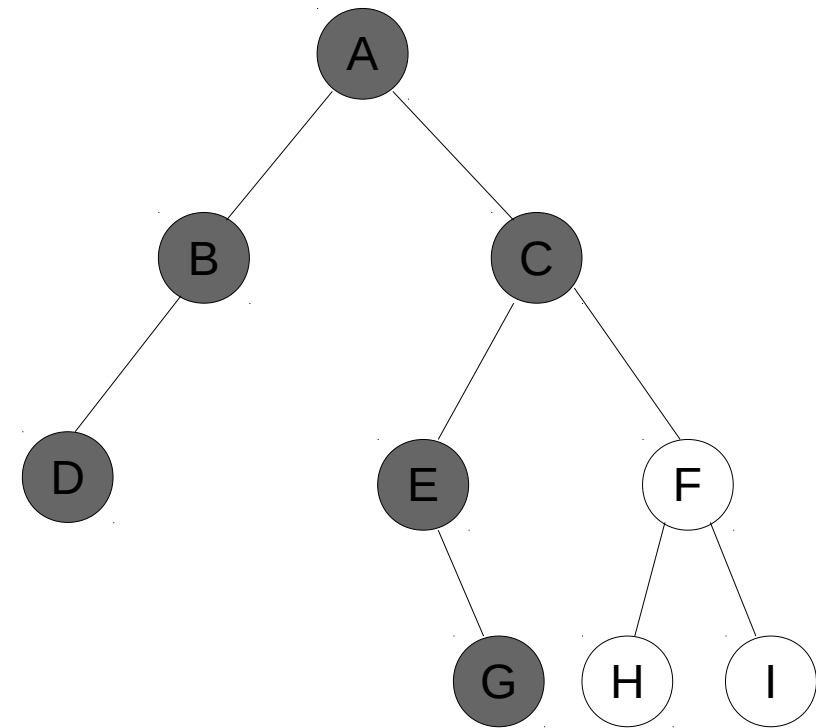
In-ordem

procedimento *in*(*pt*)

se *pt.esq* \neq *vazio* **então** *in*(*pt.esq*)

visita(*pt*)

se *pt.dir* \neq *vazio* **então** *in*(*pt.dir*)



D B A E G C

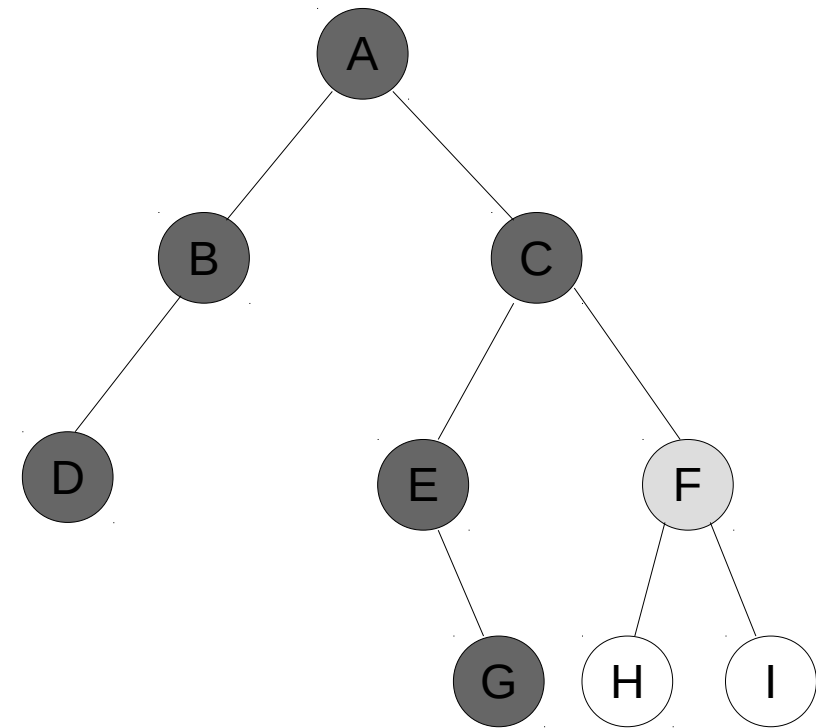
In-ordem

procedimento *in(pt)*

se *pt.esq* \neq *vazio* **então** *in(pt.esq)*

visita(pt)

se *pt.dir* \neq *vazio* **então** *in(pt.dir)*



D B A E G C

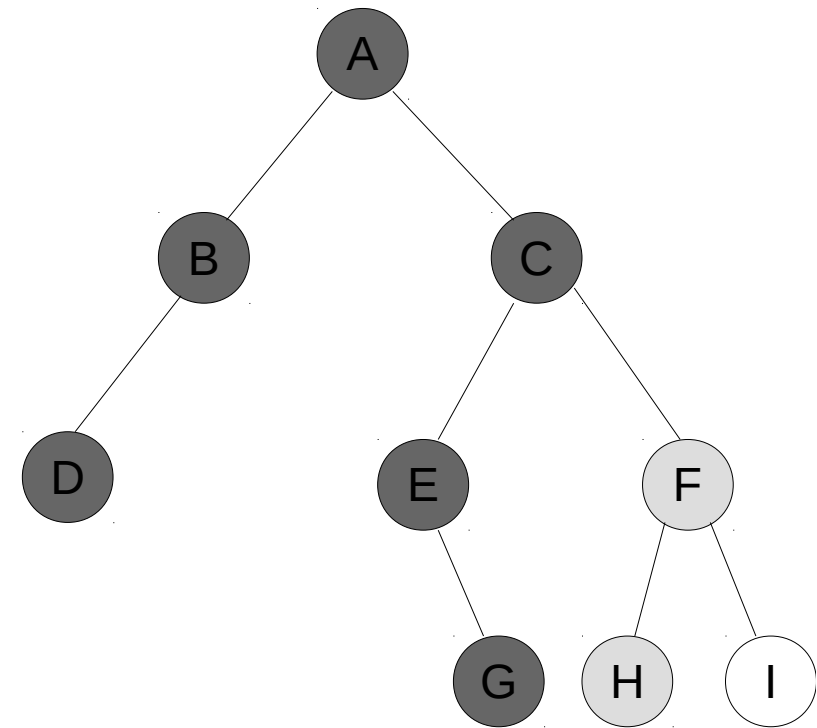
In-ordem

procedimento *in*(*pt*)

se *pt.esq* \neq *vazio* **então** *in*(*pt.esq*)

visita(*pt*)

se *pt.dir* \neq *vazio* **então** *in*(*pt.dir*)



D B A E G C

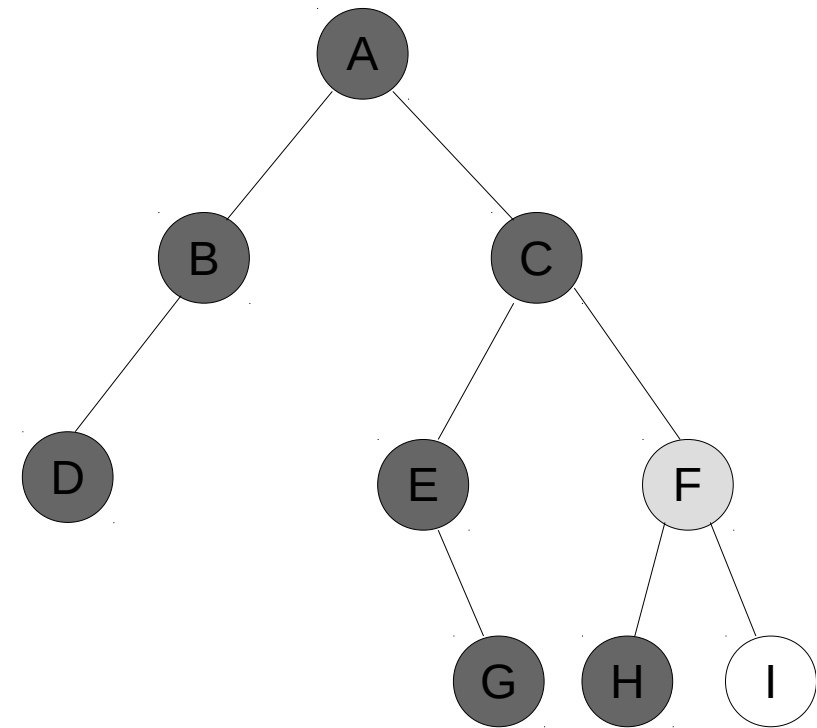
In-ordem

procedimento *in*(*pt*)

se *pt.esq* \neq *vazio* **então** *in*(*pt.esq*)

visita(*pt*)

se *pt.dir* \neq *vazio* **então** *in*(*pt.dir*)



D B A E G C H

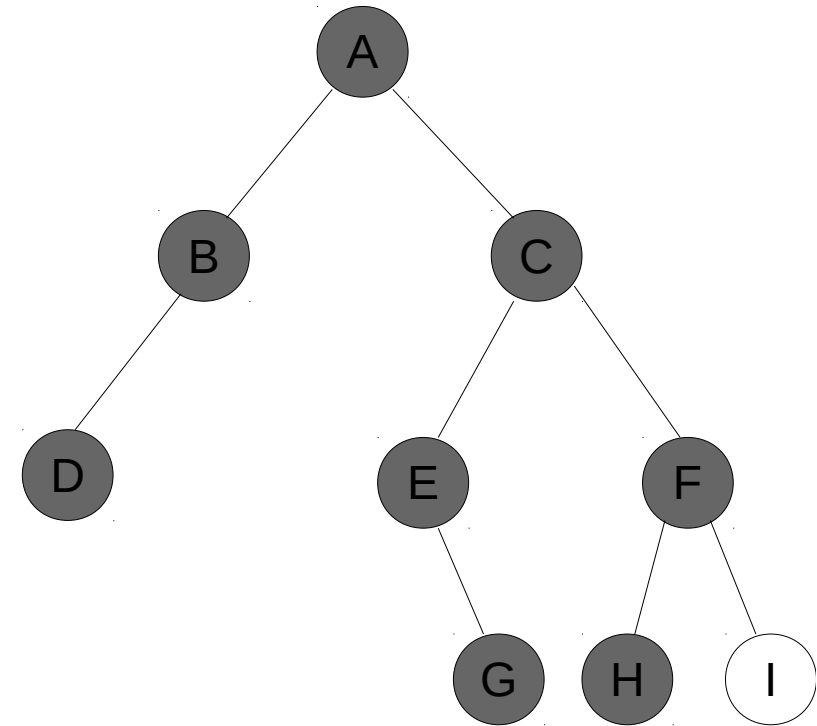
In-ordem

procedimento *in(pt)*

se *pt.esq* \neq *vazio* **então** *in(pt.esq)*

visita(pt)

se *pt.dir* \neq *vazio* **então** *in(pt.dir)*



D B A E G C H F

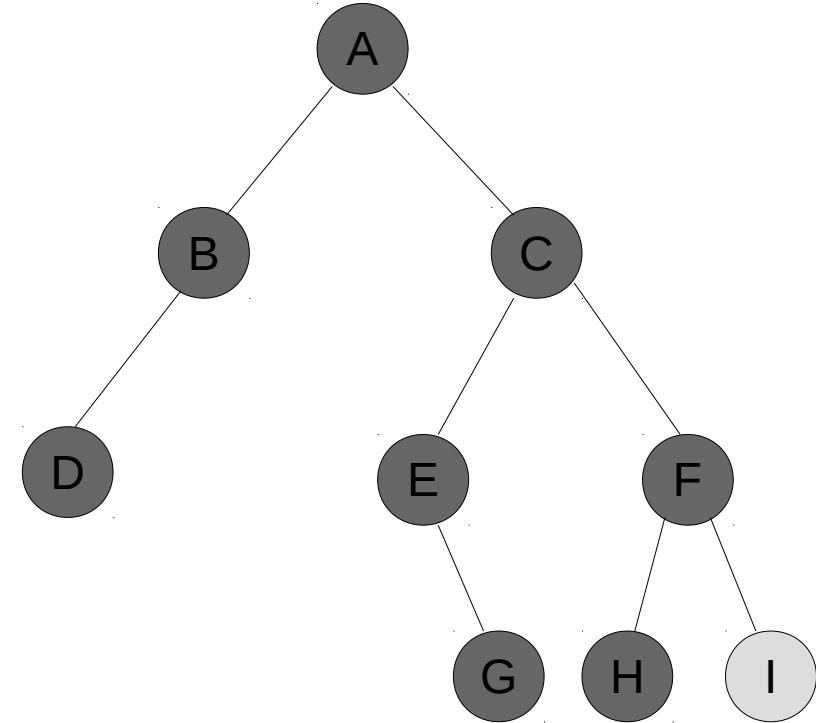
In-ordem

procedimento *in(pt)*

se *pt.esq* \neq *vazio* **então** *in(pt.esq)*

visita(pt)

se *pt.dir* \neq *vazio* **então** *in(pt.dir)*



D B A E G C H F

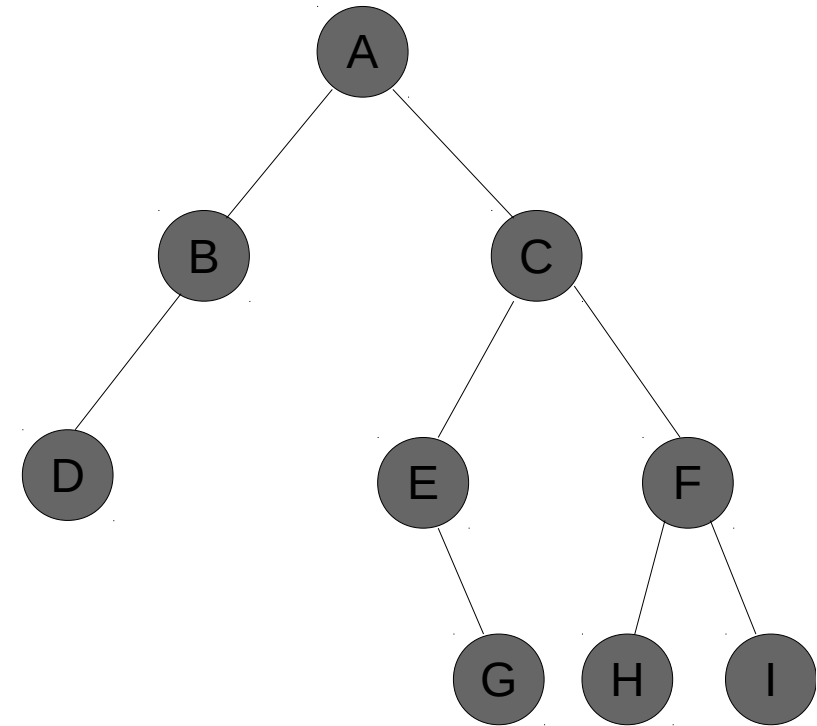
In-ordem

procedimento *in(pt)*

se *pt.esq* \neq *vazio* **então** *in(pt.esq)*


visita(pt)

se *pt.dir* \neq *vazio* **então** *in(pt.dir)*



D B A E G C H F I

In-ordem

O percurso em ordem simétrica ou in-ordem é muito utilizado para árvores binárias de busca, um dos próximos assuntos que iremos abordar! 

Exercício: Apresente a versão iterativa do método de busca in-ordem

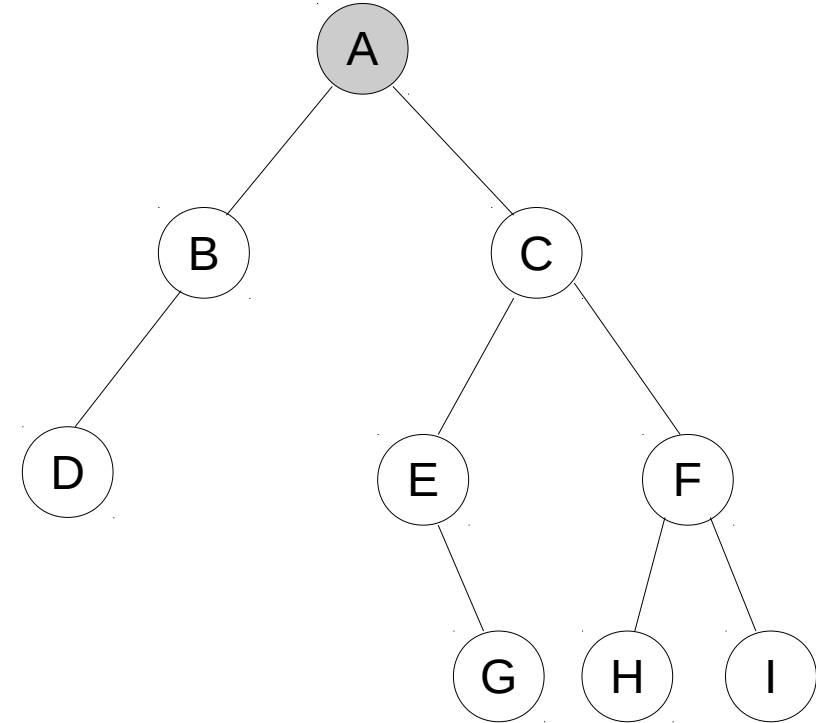
Pós-ordem

procedimento *pos(pt)*

se *pt.esq* \neq *vazio* **então** *pos(pt.esq)*

se *pt.dir* \neq *vazio* **então** *pos(pt.dir)*

visita(pt)



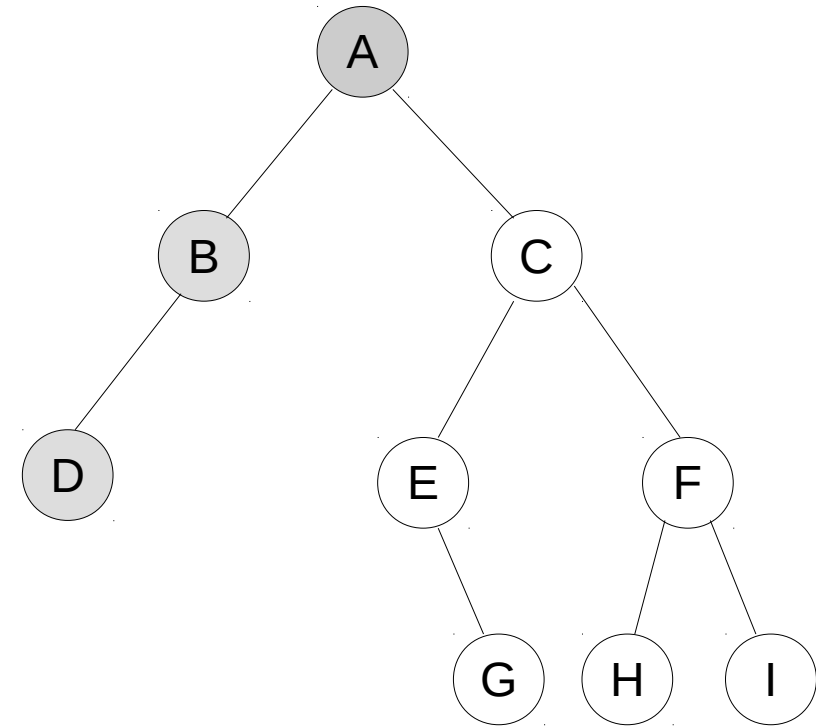
Pós-ordem

procedimento *pos(pt)*

se *pt.esq* \neq *vazio* ***então*** *pos(pt.esq)*

se *pt.dir* \neq *vazio* ***então*** *pos(pt.dir)*

visita(pt)



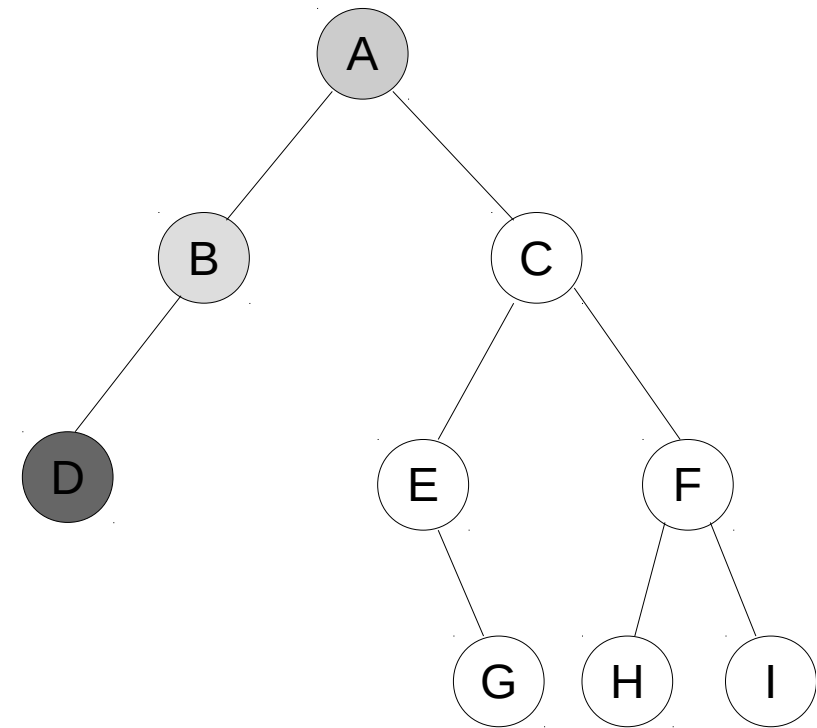
Pós-ordem

procedimento *pos(pt)*

se *pt.esq* \neq *vazio* ***então*** *pos(pt.esq)*

se *pt.dir* \neq *vazio* ***então*** *pos(pt.dir)*

visita(pt)



D

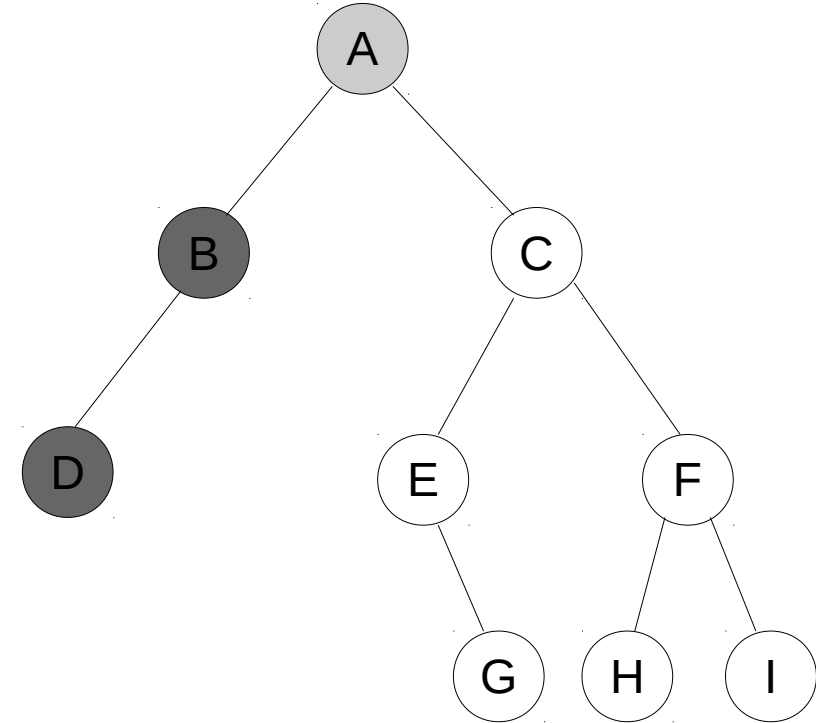
Pós-ordem

procedimento *pos(pt)*

se *pt.esq* \neq *vazio* ***então*** *pos(pt.esq)*

se *pt.dir* \neq *vazio* ***então*** *pos(pt.dir)*

visita(pt)



D B

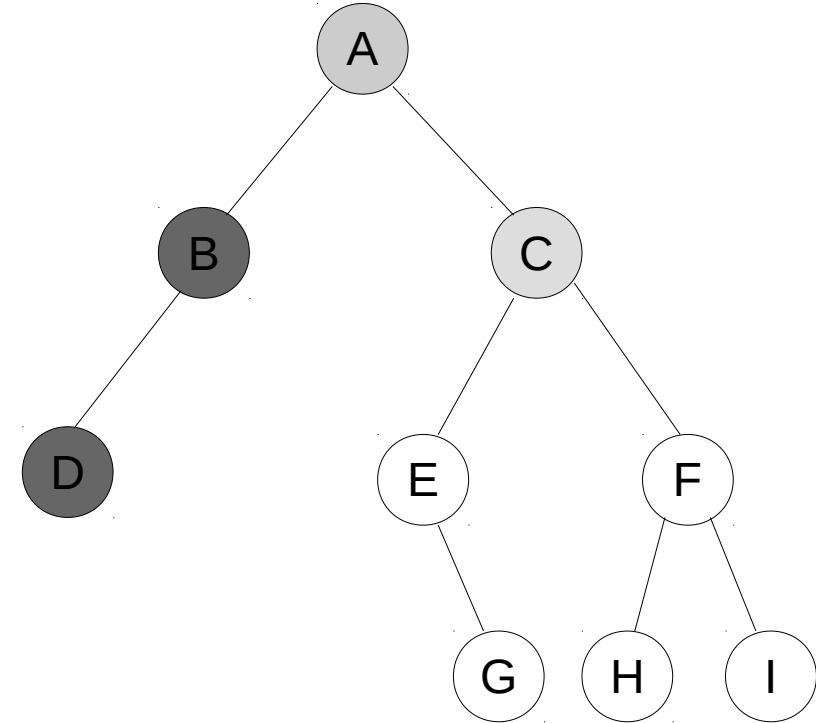
Pós-ordem

procedimento *pos(pt)*

se *pt.esq* \neq *vazio* ***então*** *pos(pt.esq)*

se *pt.dir* \neq *vazio* ***então*** *pos(pt.dir)*

visita(pt)



D B

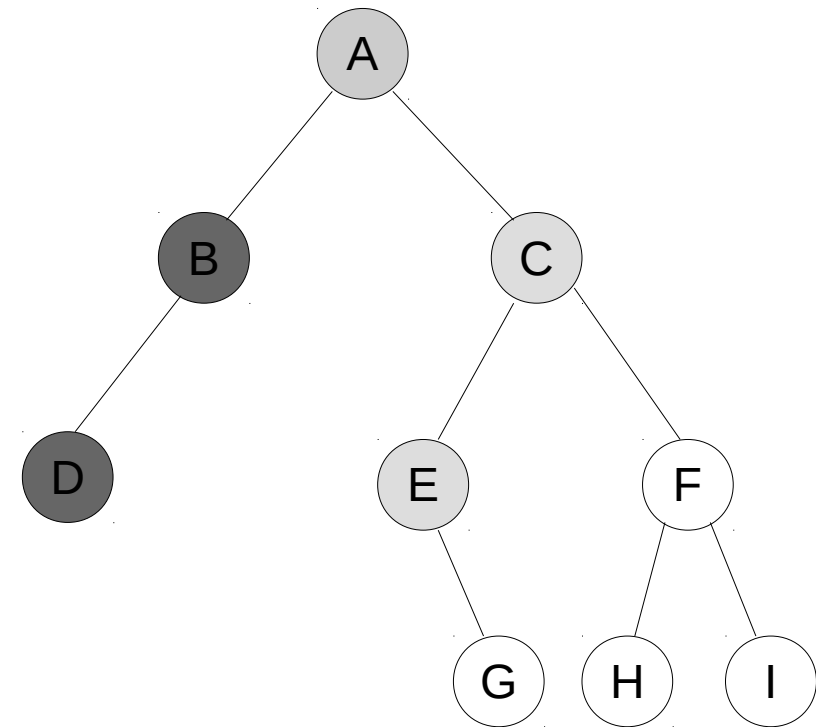
Pós-ordem

procedimento *pos(pt)*

se *pt.esq* \neq *vazio* ***então*** *pos(pt.esq)*

se *pt.dir* \neq *vazio* ***então*** *pos(pt.dir)*

visita(pt)



D B

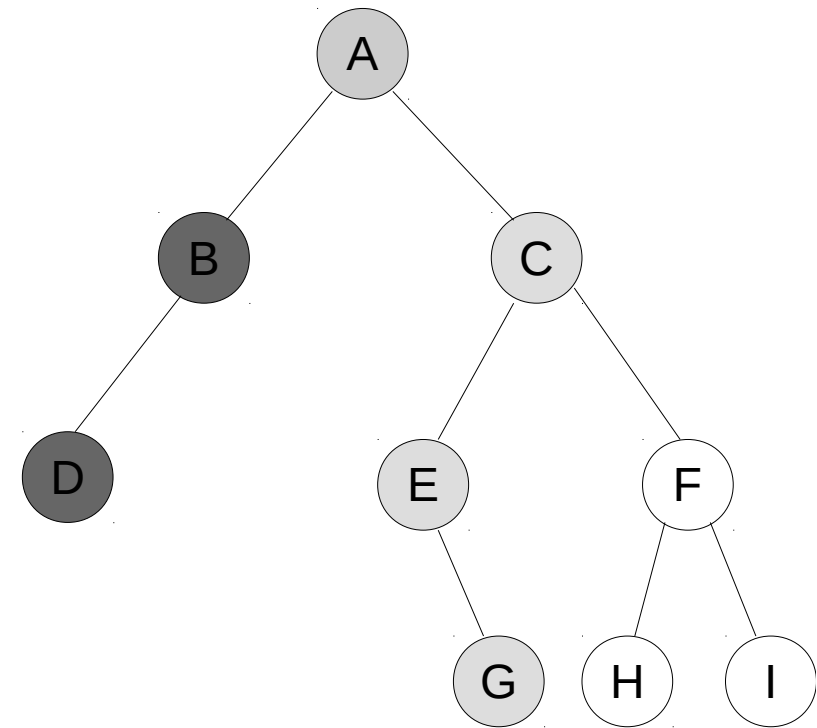
Pós-ordem

procedimento *pos(pt)*

se *pt.esq* \neq *vazio* ***então*** *pos(pt.esq)*

se *pt.dir* \neq *vazio* ***então*** *pos(pt.dir)*

visita(pt)



D B

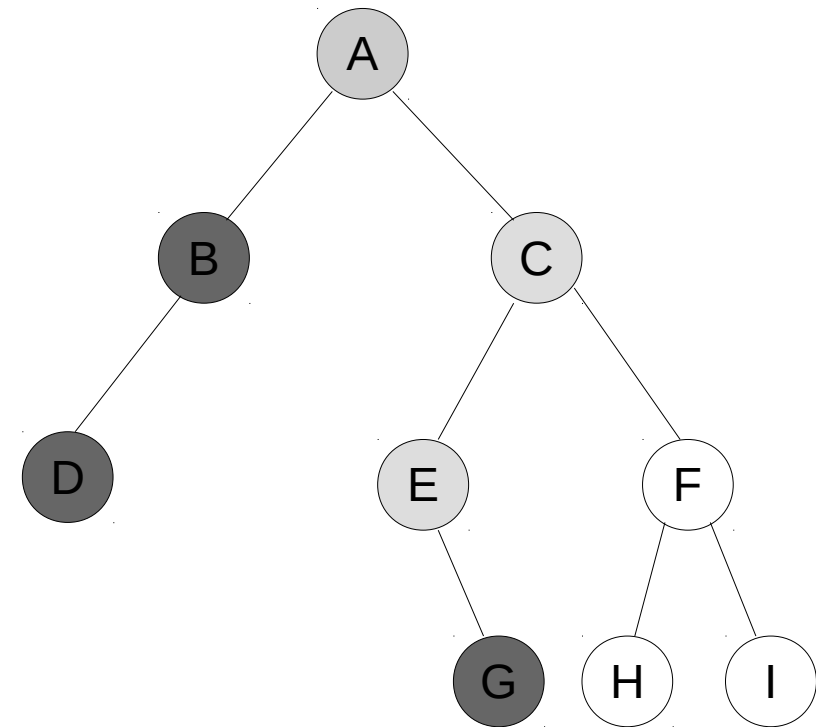
Pós-ordem

procedimento *pos(pt)*

se *pt.esq* \neq *vazio* **então** *pos(pt.esq)*

se *pt.dir* \neq *vazio* **então** *pos(pt.dir)*

visita(pt)



D B G

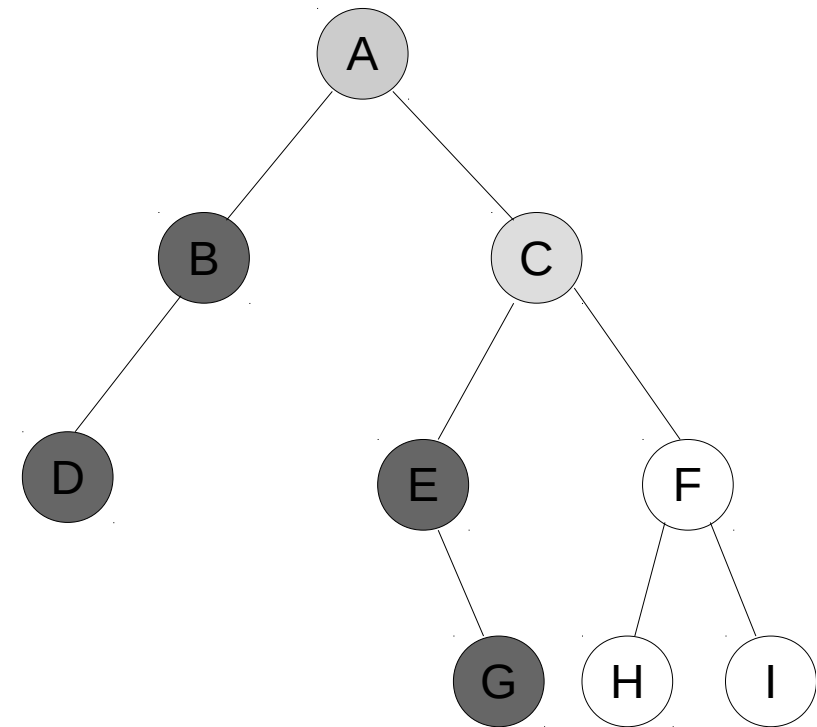
Pós-ordem

procedimento *pos(pt)*

se *pt.esq* \neq *vazio* ***então*** *pos(pt.esq)*

se *pt.dir* \neq *vazio* ***então*** *pos(pt.dir)*

visita(pt)



D B G E

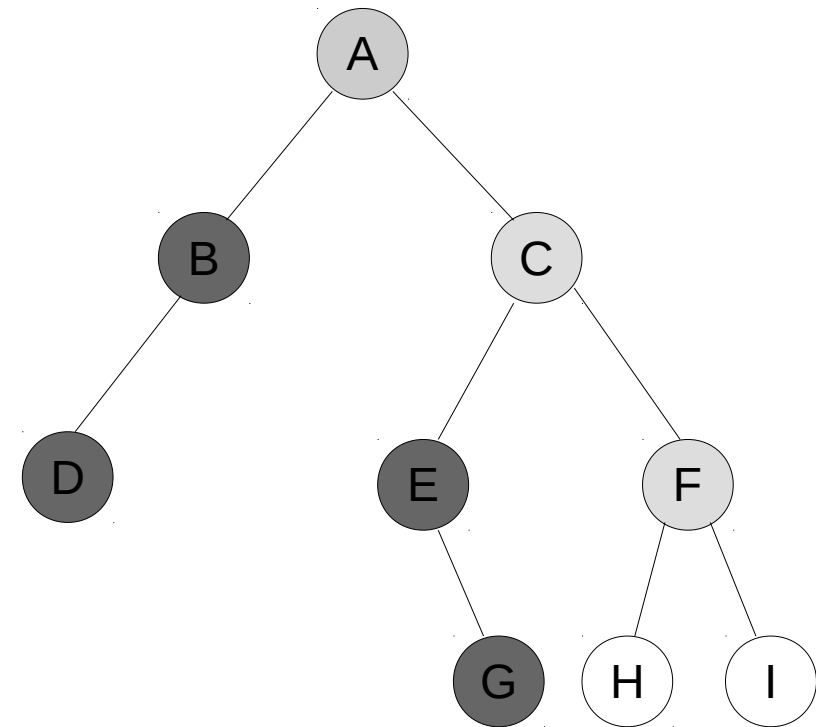
Pós-ordem

procedimento *pos(pt)*

se *pt.esq* \neq *vazio* ***então*** *pos(pt.esq)*

se *pt.dir* \neq *vazio* ***então*** *pos(pt.dir)*

visita(pt)



D B G E

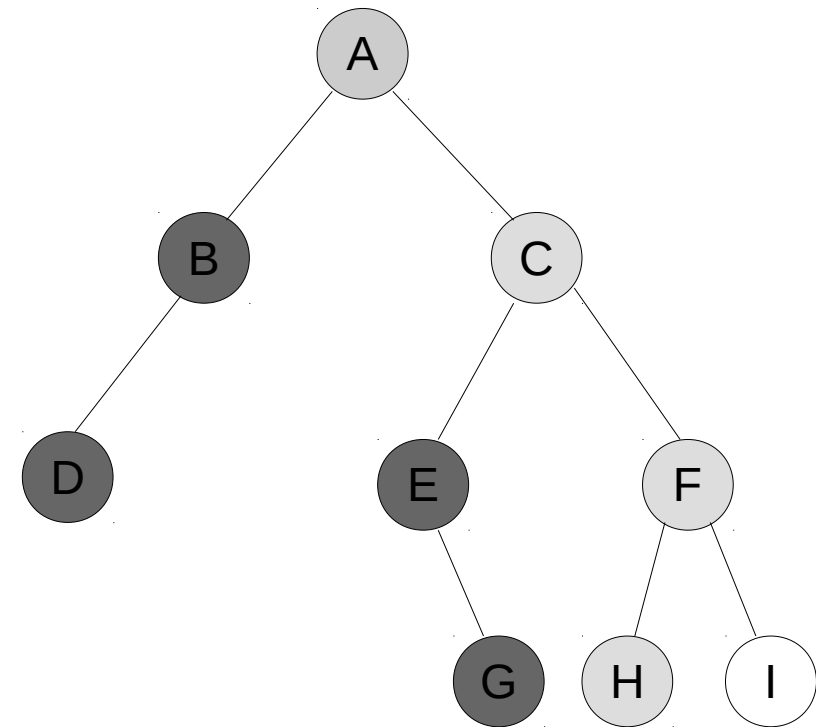
Pós-ordem

procedimento *pos(pt)*

se *pt.esq* \neq *vazio* ***então*** *pos(pt.esq)*

se *pt.dir* \neq *vazio* ***então*** *pos(pt.dir)*

visita(pt)



D B G E

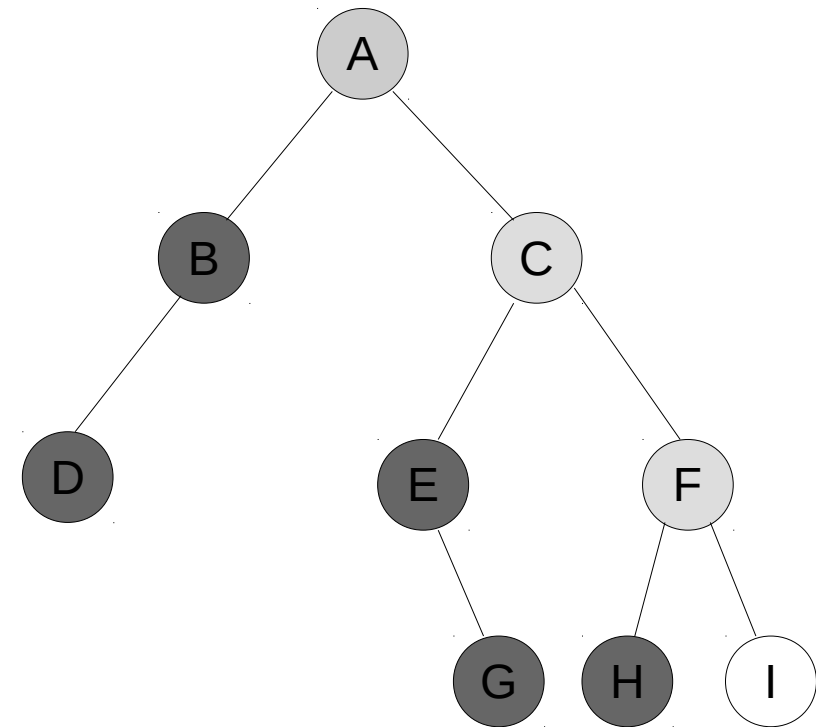
Pós-ordem

procedimento *pos(pt)*

se *pt.esq* \neq *vazio* ***então*** *pos(pt.esq)*

se *pt.dir* \neq *vazio* ***então*** *pos(pt.dir)*

visita(pt)



D B G E H

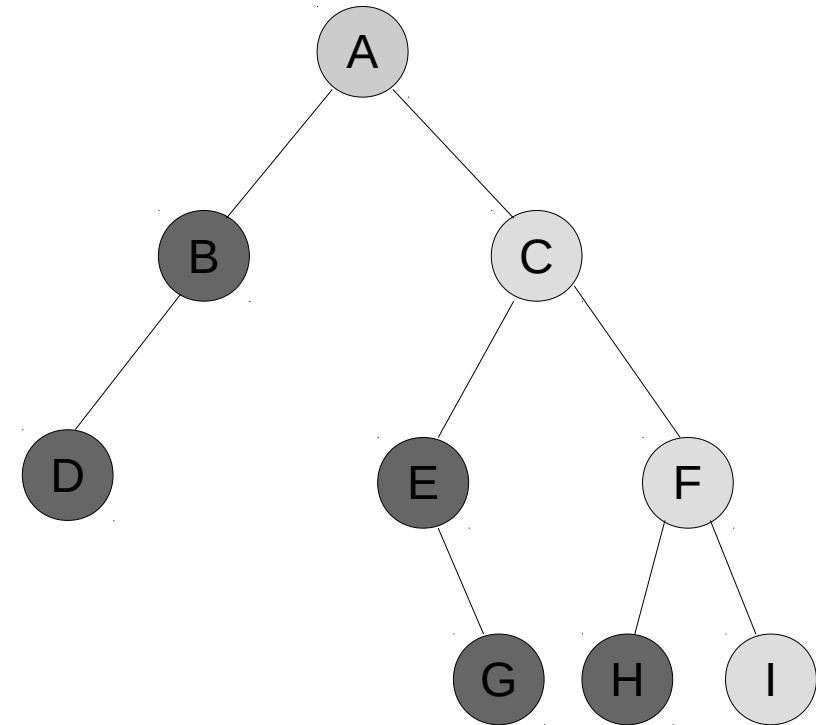
Pós-ordem

procedimento *pos(pt)*

se *pt.esq* \neq *vazio* ***então*** *pos(pt.esq)*

se *pt.dir* \neq *vazio* ***então*** *pos(pt.dir)*

visita(pt)



D B G E H

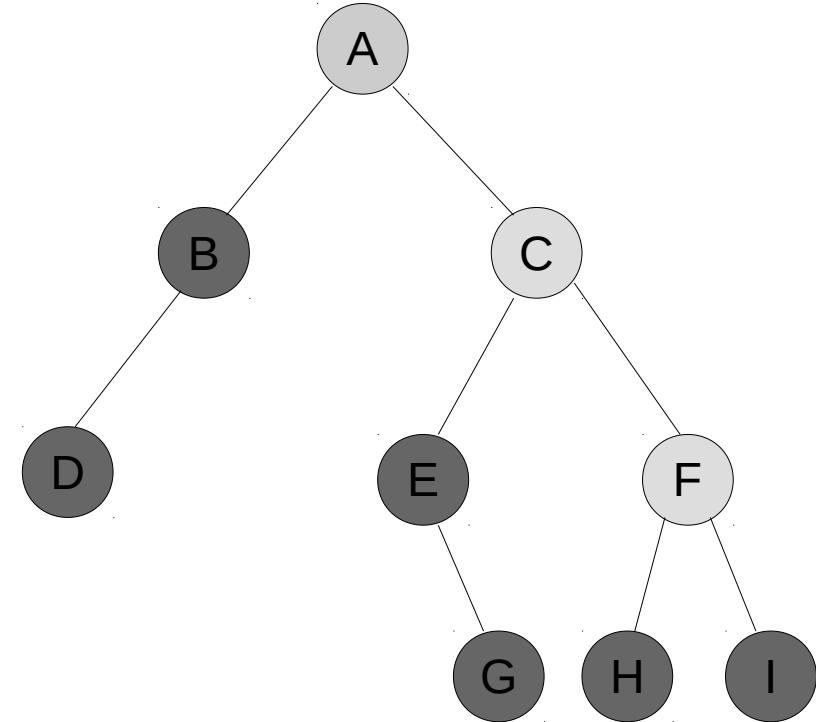
Pós-ordem

procedimento *pos(pt)*

se *pt.esq* \neq *vazio* **então** *pos(pt.esq)*

se *pt.dir* \neq *vazio* **então** *pos(pt.dir)*

visita(pt)



D B G E H I

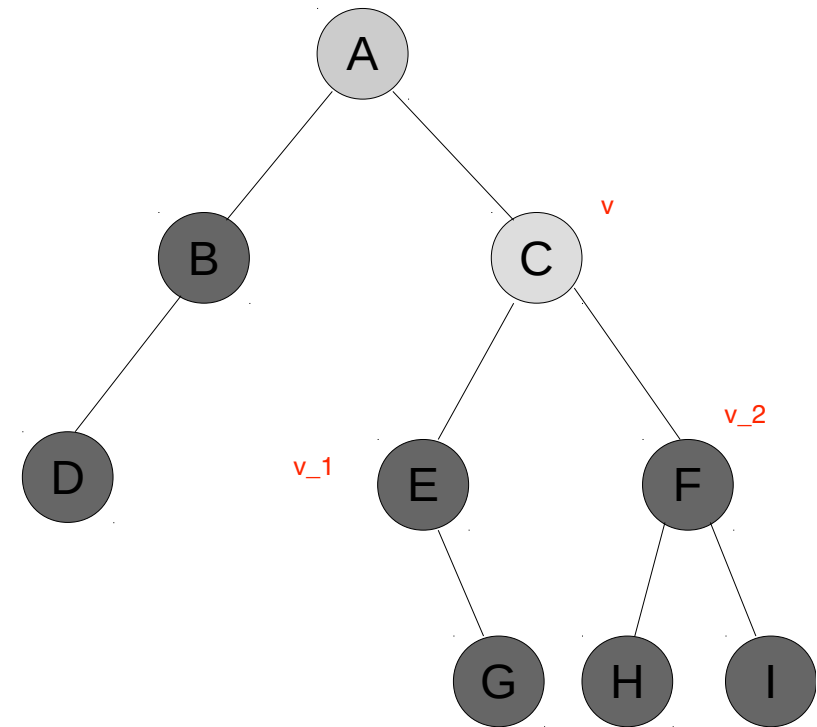
Pós-ordem

procedimento *pos(pt)*

se *pt.esq* \neq *vazio* **então** *pos(pt.esq)*

se *pt.dir* \neq *vazio* **então** *pos(pt.dir)*

visita(pt)



DBGEHIF

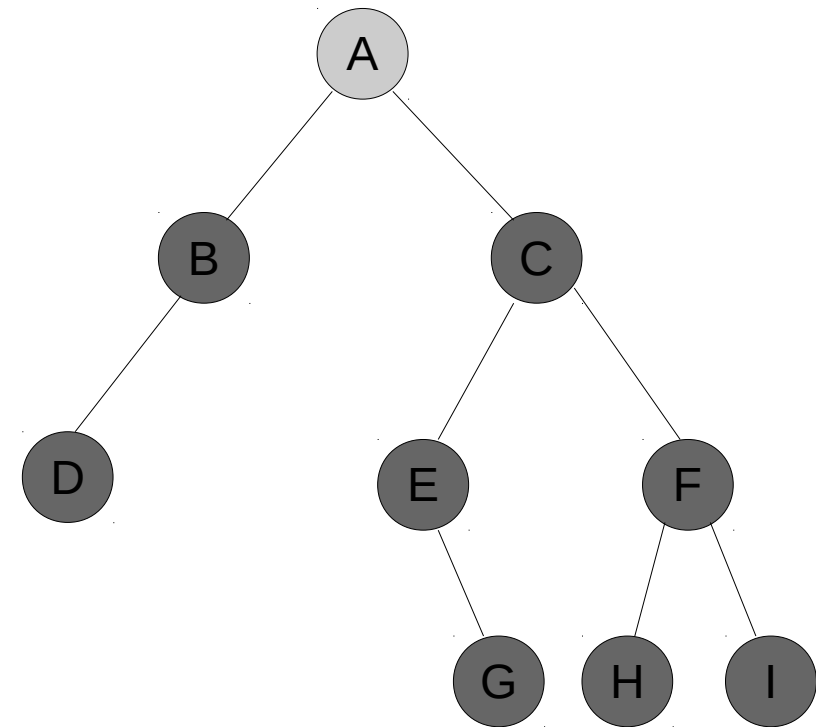
Pós-ordem

procedimento *pos(pt)*

se *pt.esq* \neq *vazio* ***então*** *pos(pt.esq)*

se *pt.dir* \neq *vazio* ***então*** *pos(pt.dir)*

visita(pt)



D B G E H I F C

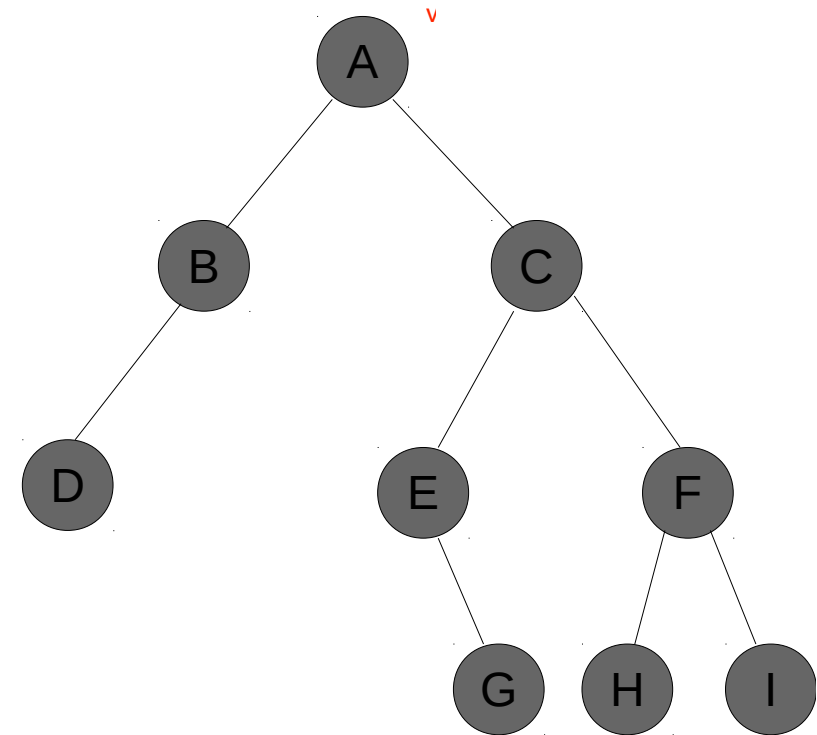
Pós-ordem

procedimento *pos(pt)*

se *pt.esq* \neq *vazio* **então** *pos(pt.esq)*

se *pt.dir* \neq *vazio* **então** *pos(pt.dir)*

visita(pt)



D B G E H I F C A

Pós-ordem

- O cálculo da altura de todos os nós de uma árvore binária é uma aplicação do percurso em pós ordem
- A altura das folhas, pela própria definição, é um
- Para os outros nós, por exemplo v , é necessário conhecer o comprimento do maior caminho de v até um dos seus descendentes

Árvores Binárias de Busca

Problema: Seja $S = \{s_1, \dots, s_n\}$, $s_1 < \dots < s_n$.
Dado um valor x o objetivo é verificar se x pertence a S ou não.

Como resolver “da melhor forma” ?

Árvores Binárias de Busca

Problema: Seja $S = \{s_1, \dots, s_n\}$, $s_1 < \dots < s_n$.
Dado um valor x o objetivo é verificar se x pertence a S ou não.

Como resolver?

Árvores Binárias de Busca

Problema: Seja $S = \{s_1, \dots, s_n\}$, $s_1 < \dots < s_n$.
Dado um valor x o objetivo é verificar se x pertence a S ou não.

→ Podemos resolver esse problema utilizando uma árvore binária T 😊

COMO?

Árvores Binárias de Busca

- Nossa árvore binária T terá as seguintes características:
 - T possui n nós. Cada nó corresponde a uma chave distinta s_j pertencente a S e possui como rótulo o valor **$rt(v) = s_j$**

Árvores Binárias de Busca

- Nossa árvore binária T terá as seguintes características:
 - T possui n nós. Cada nó corresponde a uma chave distinta s_j pertencente a S e possui como rótulo o valor **$rt(v) = s_j$**
 - Seja um nó v de T . Seja também v_1 , pertencente à subárvore esquerda de v . Então:

$$rt(v_1) < rt(v)$$

Árvores Binárias de Busca

- Nossa árvore binária T terá as seguintes características:
 - T possui n nós. Cada nó corresponde a uma chave distinta s_j pertencente a S e possui como rótulo o valor $rt(v) = s_j$
 - Seja um nó v de T . Seja também v_1 , pertencente à subárvore esquerda de v . Então:

$$rt(v_1) < rt(v)$$

- Analogamente, se v_2 pertence à subárvore direita de v ,

$$rt(v_2) > rt(v)$$

Árvores Binárias de Busca

- A árvore binária T que construímos denomina-se árvore binária de busca S .

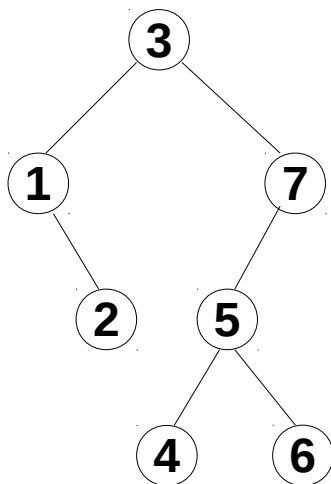
Construa uma árvore binária de busca para o conjunto $S = \{1,2,3,4,5,6,7\}$

Árvores Binárias de Busca

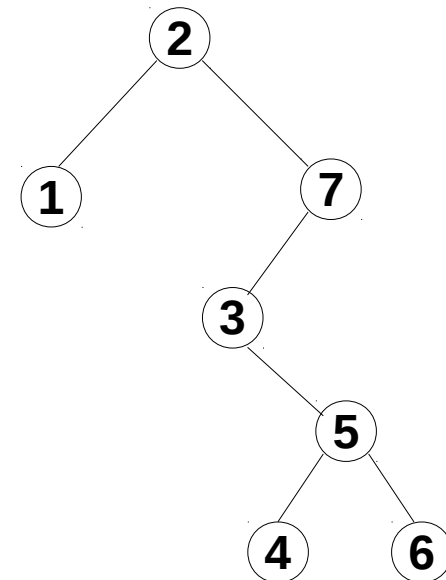
- A árvore binária T que construímos denomina-se árvore binária de busca S .

Construa uma árvore binária de busca para o conjunto $S = \{1,2,3,4,5,6,7\}$

→ Se $|S| > 1$, existem várias árvores de busca para S



busca-arvore(x, pt, f)



Algoritmo – Busca em Árvore Binária de Busca

Problema: Determinar se dado uma chave x pertence a Árvore de Busca S .

Algoritmo – Busca em Árvore Binária de Busca

Problema: Determinar se dado uma chave x pertence a Árvore de Busca S .

SAÍDA: $f = 0$, se a árvore é vazia

$f = 1$, se x pertence a S . Neste caso, um ponteiro pt aponta para o nó procurado

$f > 1$, se x não pertence a S

Algoritmo – Busca em Árvore Binária de Busca

procedimento *busca-arvore*(x, pt, f)

se pt = *vazio* **então** f = 0

senão se x = pt.chave **então** f = 1

senão se x < pt.chave **então**

se pt.esq = *vazio* **então** f = 2

senão pt = pt.esq

busca-arvore(x, pt, f)

senão se pt.dir = *vazio* **então** f = 3

senão pt = pt.dir

busca-arvore(x, pt, f)

Algoritmo – Inserção em Árvore Binária de Busca

Problema: Dado um valor x deve se inserir o valor na árvore binária de busca mantendo suas propriedades. Caso já exista a chave a inserção não deve ser realizada.

Algoritmo – Inserção em Árvore Binária de Busca

procedimento *insercao-arvore* (x)

pt = ptraiiz

busca-arvore(x, pt, f)

se f = 1 **então** “*inserção inválida*”

senão *ocupar*(pt1)

pt1.chave = x

pt.esq = *vazio*

pt.dir = *vazio*

se f = 0 **então** ptraiiz = pt1

senão se f = 2 **então**

pt.esq = pt

senão pt.dir = pt1

Próxima aula...

- Alguns conceitos sobre árvore binária de busca
- Árvore de Partilha

Construção de uma Árvore Binária de Busca

Com vimos na última aula, para construir uma árvore de busca, pode-se utilizar o algoritmo de inserção em árvore binária de busca.

Algoritmo – Inserção em Árvore Binária de Busca

procedimento *insercao-arvore* (x)

pt = ptraiiz

busca-arvore(x, pt, f)

se f = 1 **então** “*inserção inválida*”

senão *ocupar*(pt1)

pt1.chave = x

pt.esq = *vazio*

pt.dir = *vazio*

se f = 0 **então** ptraiiz = pt1

senão se f = 2 **então**

pt.esq = pt

senão pt.dir = pt1

procedimento *busca-arvore*(x, pt, f)

se pt = *vazio* **então** f = 0

senão se x = pt.chave **então** f = 1

senão se x < pt.chave **então**

se pt.esq = *vazio* **então** f = 2

senão pt = pt.esq

busca-arvore(x, pt, f)

senão se pt.dir = *vazio* **então** f = 3

senão pt = pt.dir

busca-arvore(x, pt, f)