Mestrado Decisão Económica e Empresarial
COMPUTAÇÃO

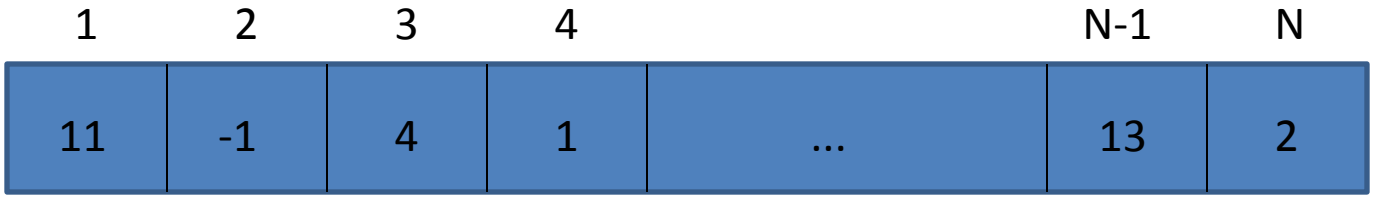
Tipos de dados abstractos.
Representação de Grafos. Determinação de componentes conexas.

Sumário

- Arrays
- Pilhas
- Filas
- Árvores
- Grafos
- Topological Sorting

Arrays

- Estrutura de dados linear
- Estástic

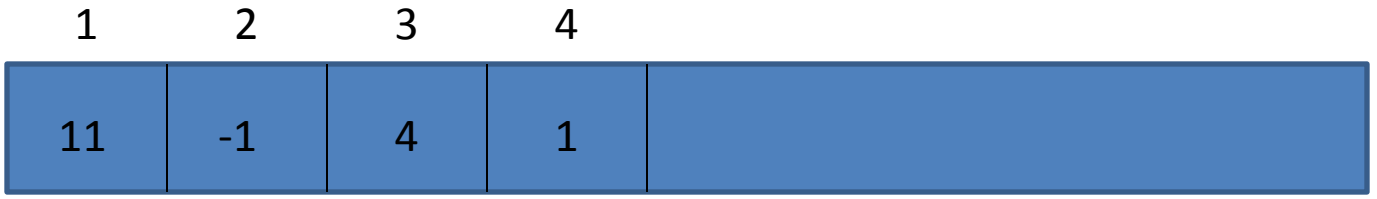


- Operações com arrays: acesso, inserção, remoção, etc.
- Implementação (VBA): **Dim** v(1 **To** N) **as** TipoDado

Pilhas

- Estrutura de dados baseado no princípio **LIFO** *last in first out*
- Operações usuais: inserção (início da pilha), remoção (início da pilha)
- Implementação:

```
Dim pilha() as TipoDado
Dim DimPilha as integer
```



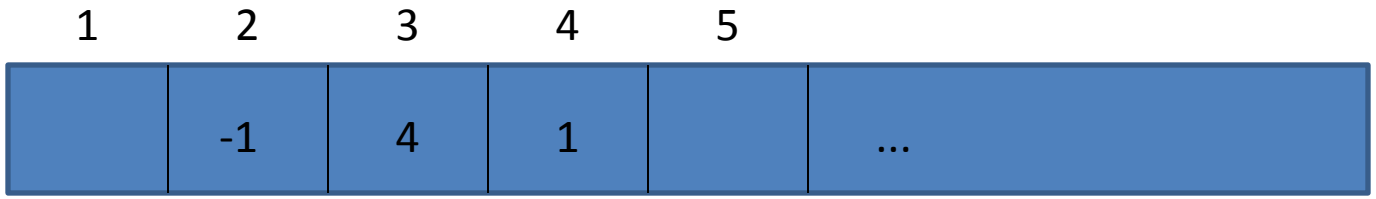
DimPilha=4

Filas

- Estruturas de dados baseadas no princípio **FIFO** *first in first out*.
- Operações usuais: *enqueue* (inserção no fim da fila) e *dequeue* (remoção do início da fila)
- Implementação:

Dim fila() **as** TipoDado

Dim cabeca **as integer**, cauda **as integer**



cabeca=2 e cauda=5

Árvores: Motivação

Diversas aplicações necessitam de estruturas mais complexas do que as listas, pilhas e filas?

Ex.: visualizar o conjunto de directórios.

Exitem algoritmos eficientes para tratamento de árvores.

Árvore

Estabelece um estrutura hierárquica numa colecção de objectos. Exemplos:

- Organização dos directorios num computador;
- Índice de um livro;
- Árvore geneológica;
- Estrutura hierárquica de uma organização;
- ...

Árvore

Uma árvore armazena os elementos de uma forma hierárquica

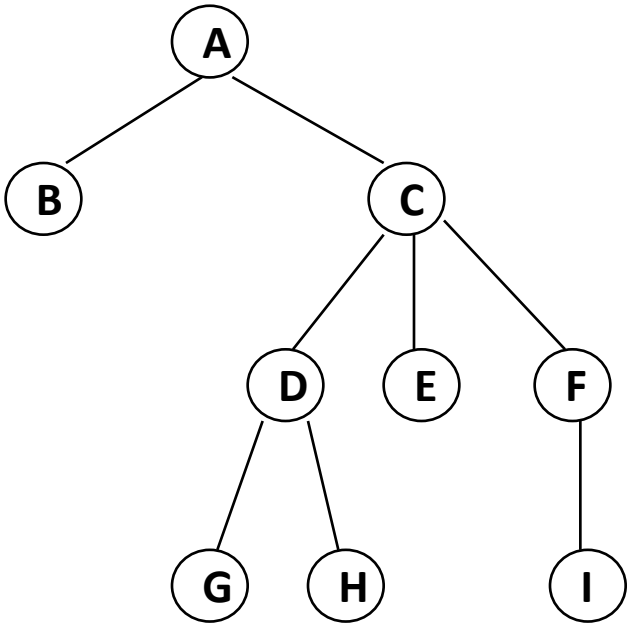
É um conjunto finito de nós ou vértices, com informação e uma relação entre si de parentesco,

Se uma árvore é não vazia

- Existe um nó **r**, chamado raiz de **T**, que não tem pai;
- Todos os restantes nós (excepto a raíz) têm um único pai;

Representação

Hierárquica



Alinhamento dos nós

A

B

C

D

G

H

E

F

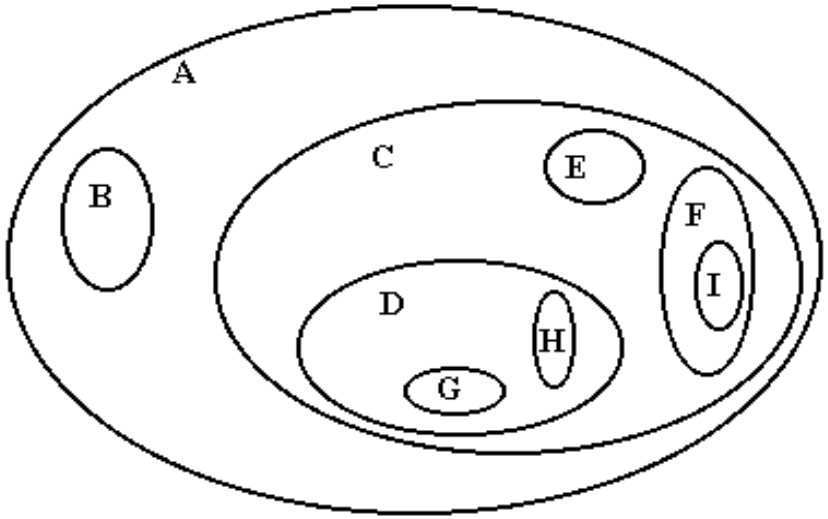
I

Representação

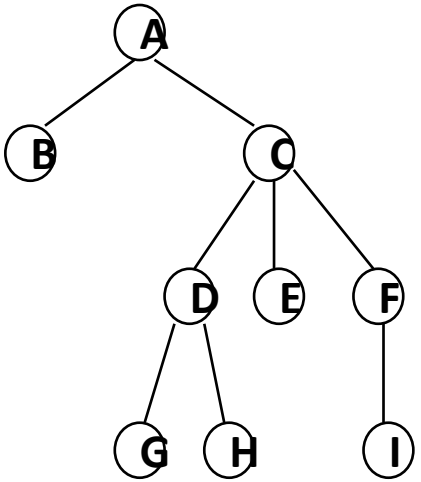
Parênteses

(A (B) (C (D (G) (H)) (E) (F (I))))

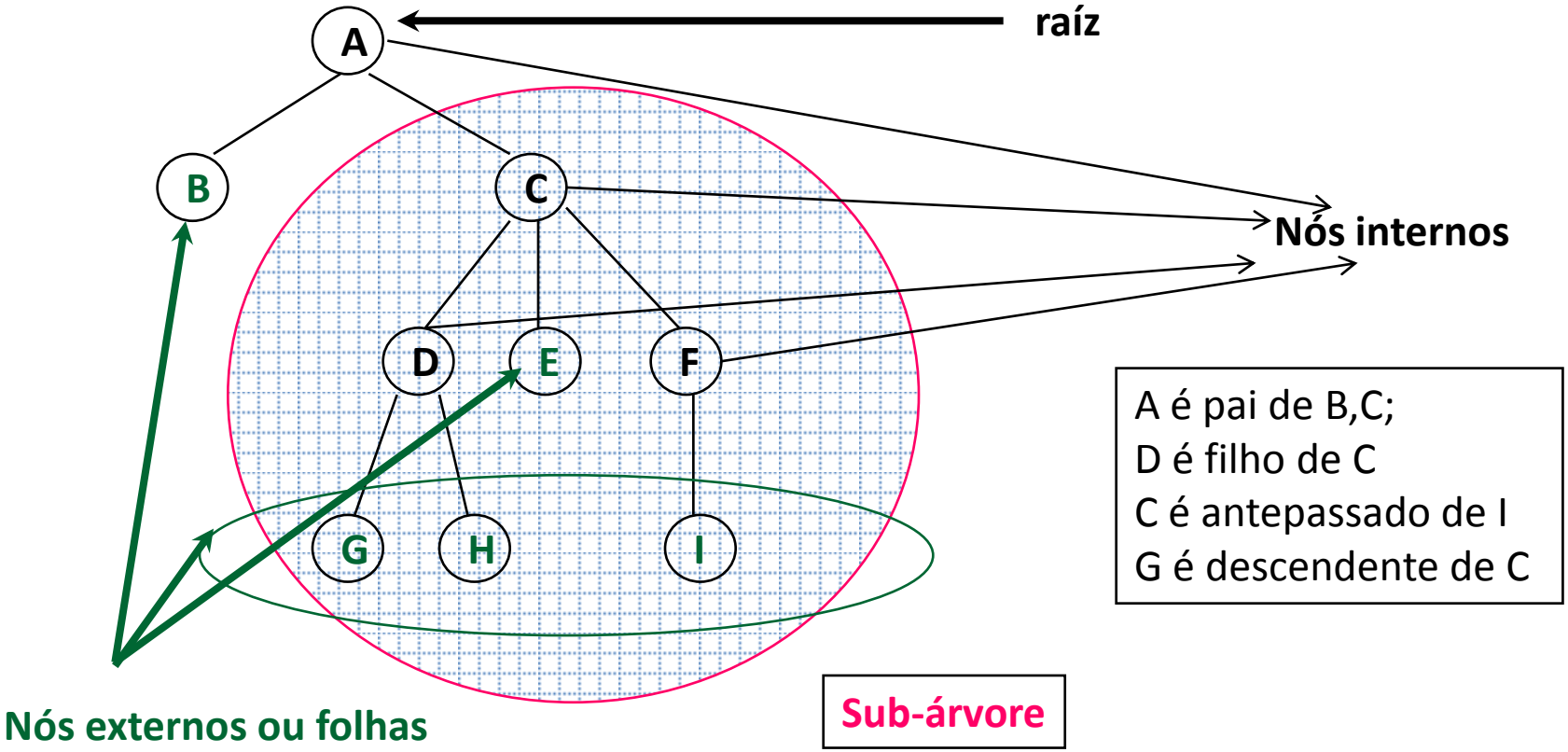
Diagramas de inclusão



Hierárquica



Terminologia



A é pai de B,C;
D é filho de C
C é antepassado de I
G é descendente de C

Terminologia

Qualquer nó da árvore (raíz incluída)

- ou é interno – nó que tem um ou mais filhos
- ou é externo (ou folha) – nó sem filhos.

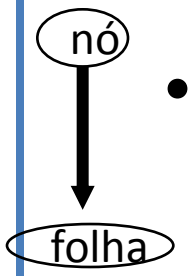
Diz-se que o nó u é **antepassado** de v sse
u=v ou u é antepassado do pai de v;
Diz-se que nó o v é **descendente** de u sse
v=u ou v é descendente de um filho de u;
Sub-árvore de T com raíz u – nó u e todos os seus descendentes mantendo as relações entre si;

Aresta da árvore – (u,v) em que u é pai de v;
Caminho na árvore – sequência de nós em que 2 nós consecutivos são uma aresta da árvore
(na terminologia de computação)

Altura e profundidade

Altura

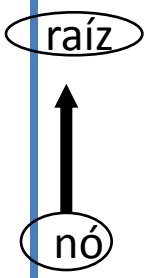
- Do nó – número de nós do caminho mais longo até uma folha;
- Da árvore – altura da raíz



convenções: altura da folha 1, altura da árvore vazia 0

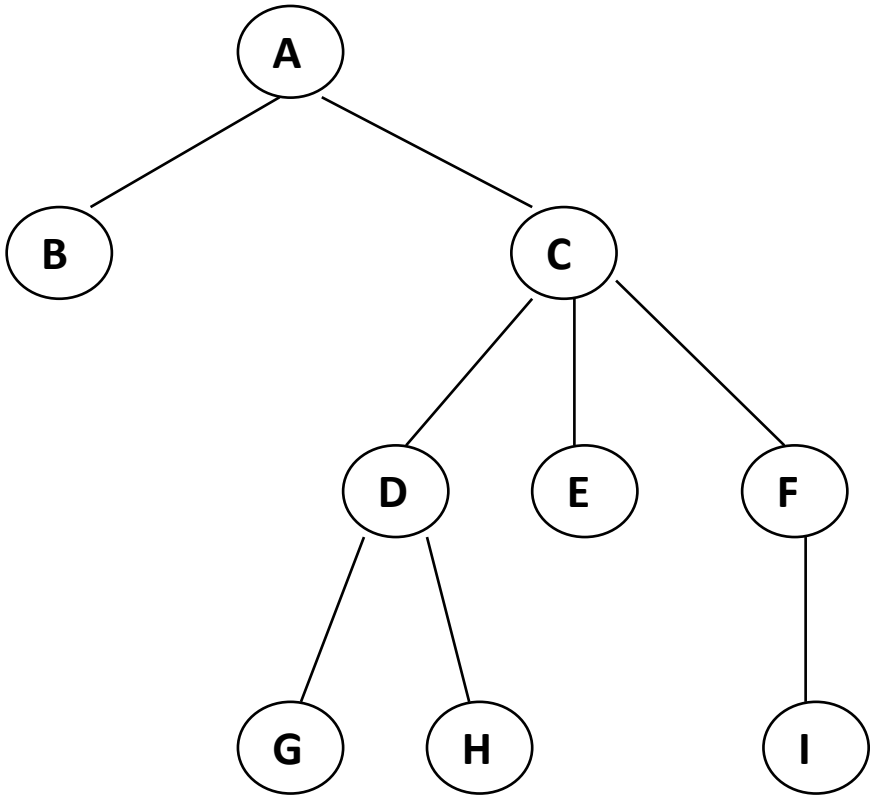
Profundidade

- Do nó – número de nós do único caminho da raíz ao nó;
- Da árvore – profundidade máxima (das suas folhas);



convenções: profundidade da raíz 1, **profundidade** da árvore vazia 0

Altura e profundidade exemplo



Altura da árvore 4

Profundidade da árvore 4

Nó	altura	prof
A	4	1
B	1	2
C	3	2
D	2	3
E	1	3
F	2	3
G	1	4
H	1	4
I	1	4

Operações na árvore

- Determinar o número de nós da árvore
- Responder se a árvore está ou não vazia
- Verificar se um nó
  - É interno
  - É folha
  - É a raíz
- Devolver
  - O pai de um nó
  - Os filhos de um nó
  - A informação contida num nó
- Operações para
  - Criar uma árvore
  - Colocar informação num nó
  - Adicionar uma sub-árvore a uma árvore

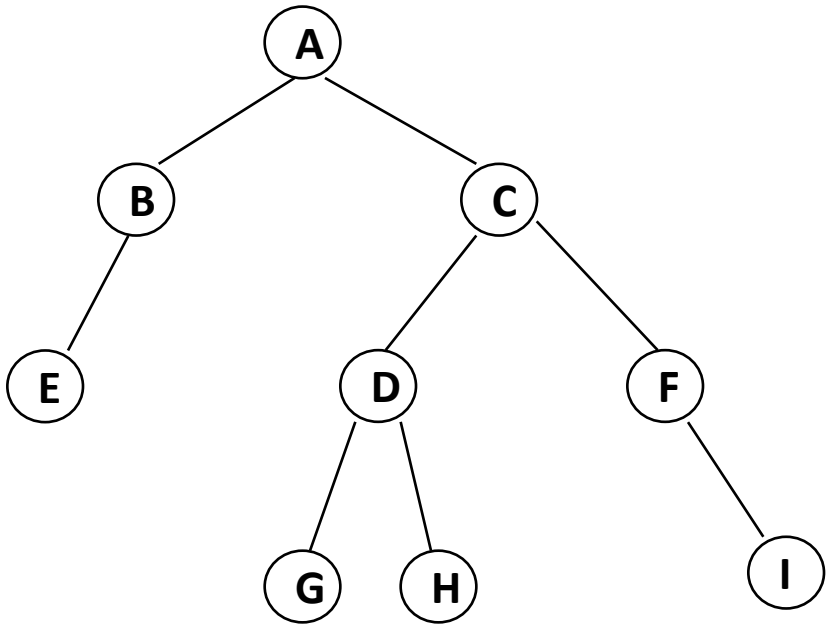
Árvores binárias

Uma árvore binária é uma árvore em que cada nó tem no máximo 2 filhos.

Os filhos são habitualmente designados por

filho esquerdo e

filho direito.



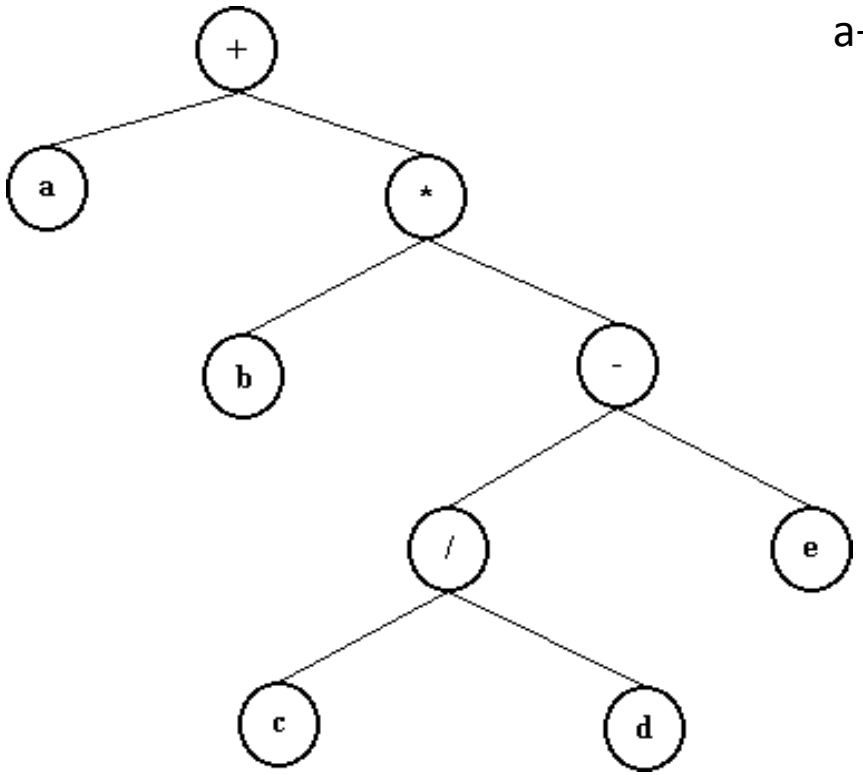
Exemplo de árvore binária

Exemplos de Aplicações

- Árvores de decisão
  - Questões SIM / NÃO
- Pesquisa de elementos
- Árvores com expressões aritméticas
  - Operandos nos nós internos
  - Operadores nas folhas

Expressão aritmética:

(a + (b *((c / d) - e)))



a+(b*(c/d-e))

b*(c/d-e)

c/d-e

c/d

Percursos em árvores binárias

Depth-first

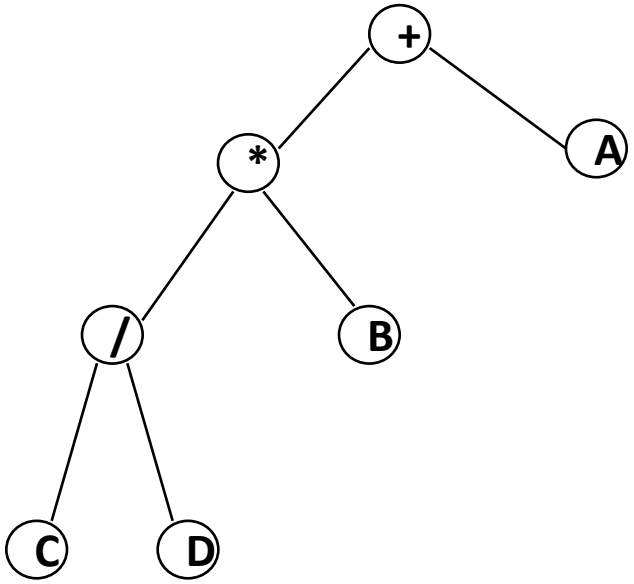
- **Pre-order:**
  - Visitar nó
  - Visitar sub-árvore da esquerda
  - Visitar sub-árvore da direita
- **In-order:**
  - Visitar sub-árvore da esquerda
  - Visitar nó
  - Visitar sub-árvore da direita
- **Post-order:**
  - Visitar sub-árvore da esquerda
  - Visitar sub-árvore da direita
  - Visitar nó

Breadth-first

- **Level-order:**

  Os nós de um nível são visitados antes dos nós do nível seguinte

Percursos em árvores binárias exemplo



Percurso	expressão
Pre-order	+*/CDBA
In-order	C/D*B+A
Post-order	CD/B*A+
level-order	+*A/BCD

Árvore binária de pesquisa

Numa árvore binária ordenada, para o todo o nó *u* verifica-se:

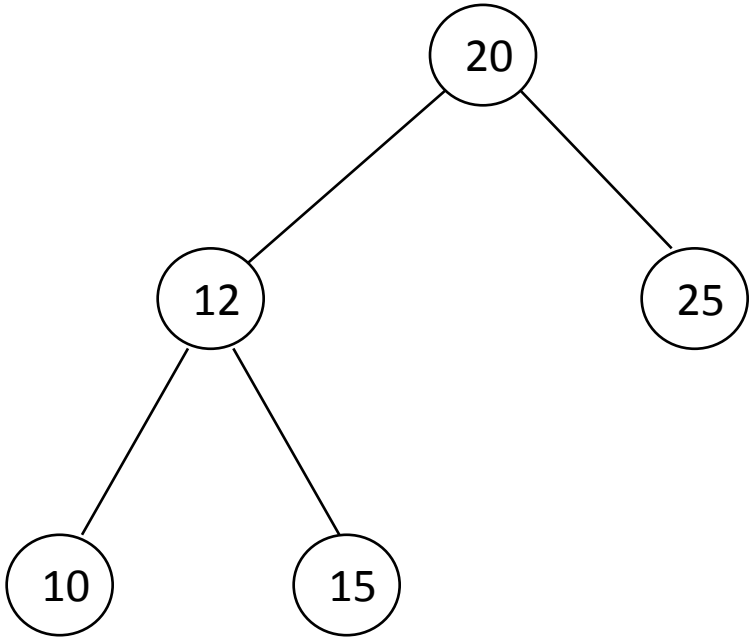
- Qualquer nó pertencente à sub-árvore esquerda com raíz em *u* tem valor menor do que o valor de *u*

- Qualquer nó pertencente à sub-árvore direita com raíz em *u* tem valor maior do que o valor de *u*

Uma árvore binária de pesquisa está ordenada:

- Se não há repetições de elementos
- Se as sub-árvores esquerda e direita também são árvores binárias de pesquisa

Árvore binária de pesquisa exemplo



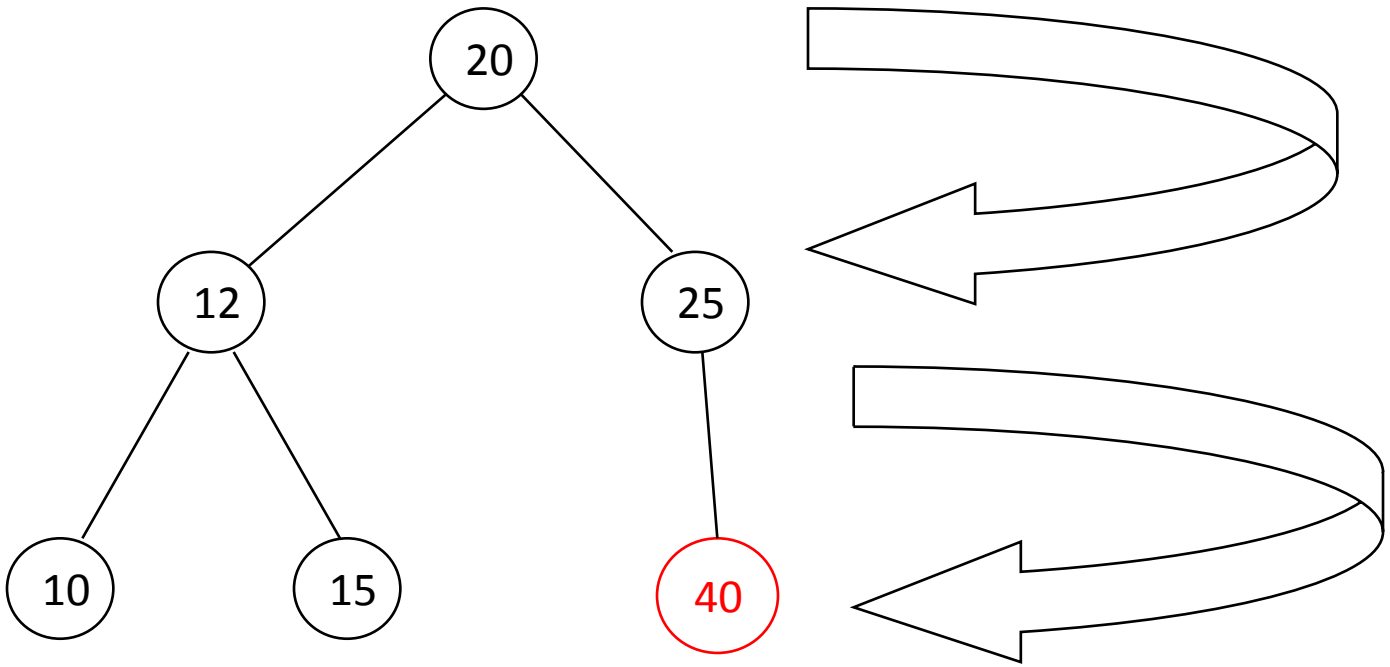
Pesquisa - Complexidade

A pesquisa de um elemento na árvore binária de pesquisa devolve ou o nó em que se encontra o elemento ou a informação de que tal elemento não se encontra na árvore.

No pior caso o número de comparações é da ordem da altura da árvore.

INSERÇÃO Árvore binária de pesquisa

Inserção de 40



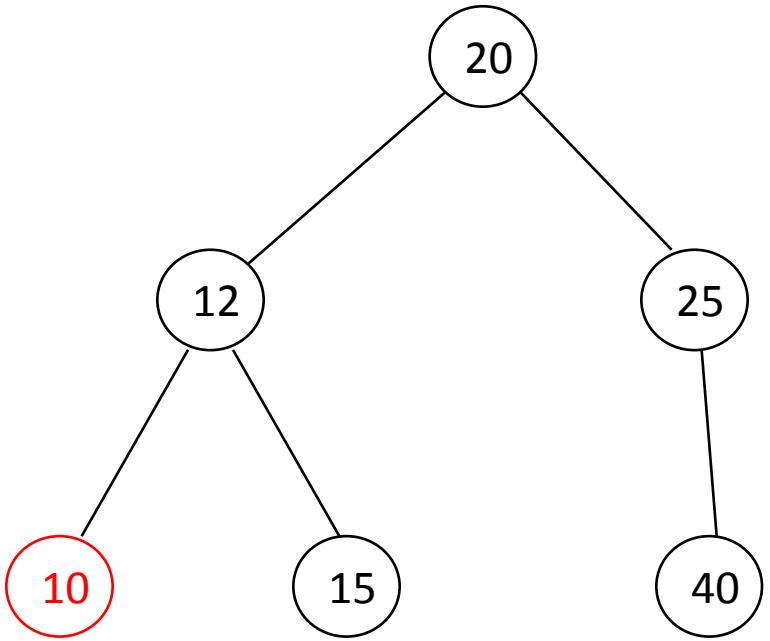
REMOÇÃO Árvore binária de pesquisa

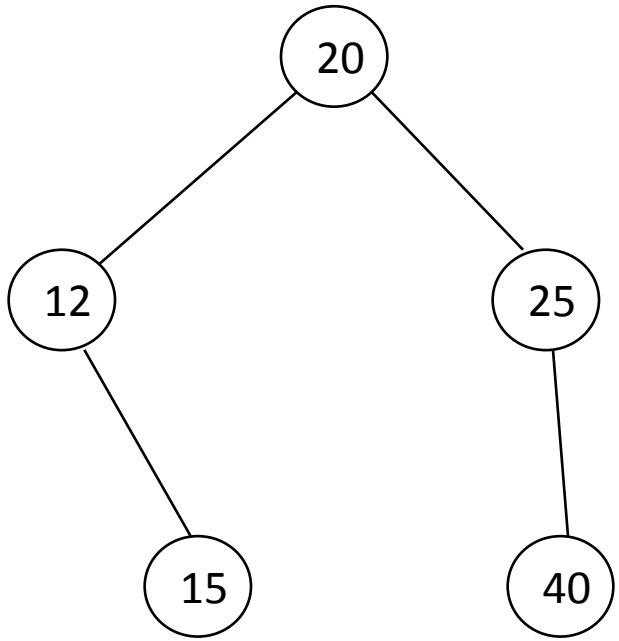
Remoção de um elemento que

- é uma folha – retira-se simplesmente esse nó da árvore;
- é nó com apenas um filho – filho colocado no lugar do nó a retirar;
- É nó com 2 filhos – coloca-se no lugar do nó a remover o menor elemento da sub-árvore direita.

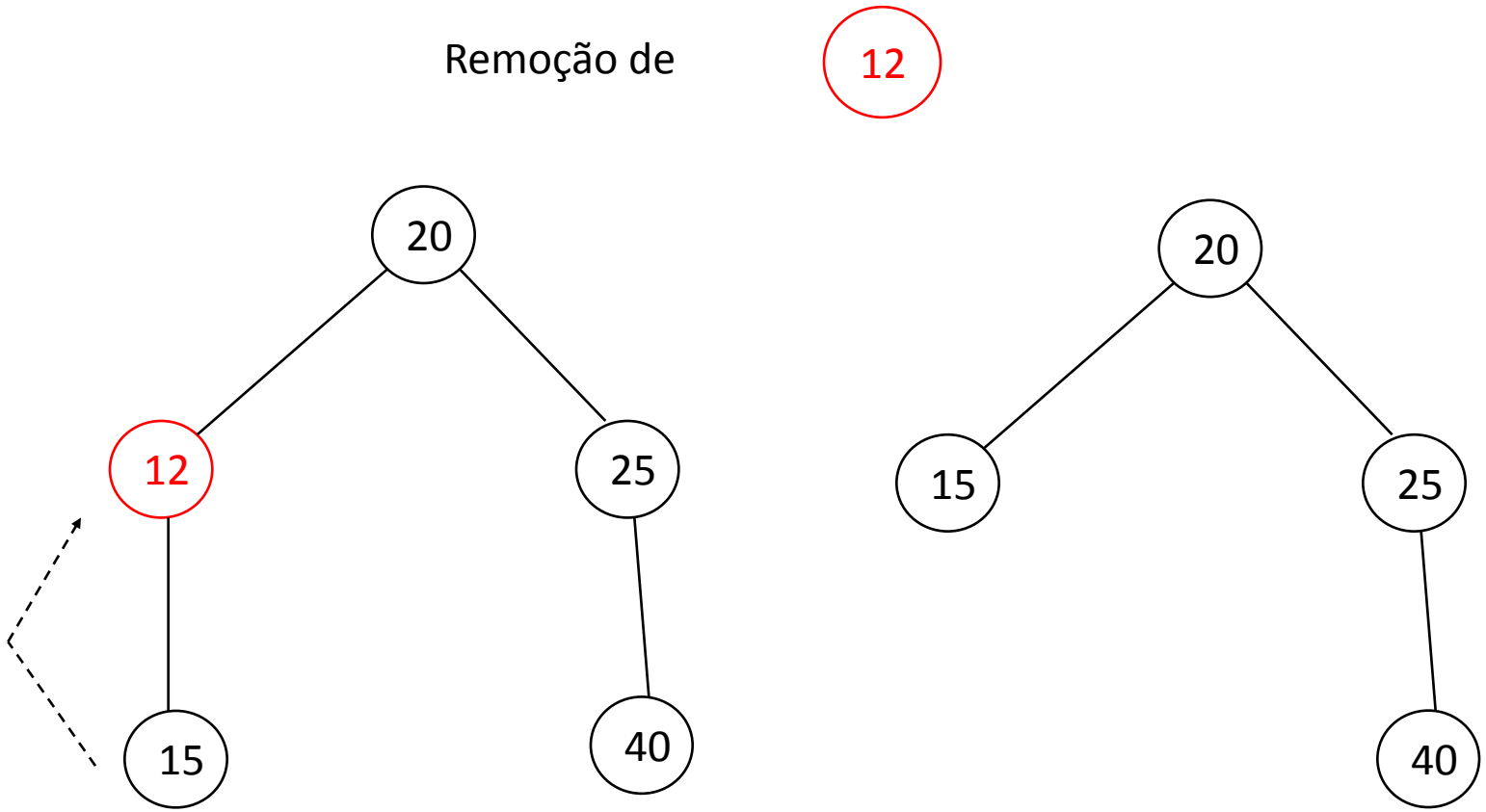
REMOÇÃO de uma folha

Remoção de 10

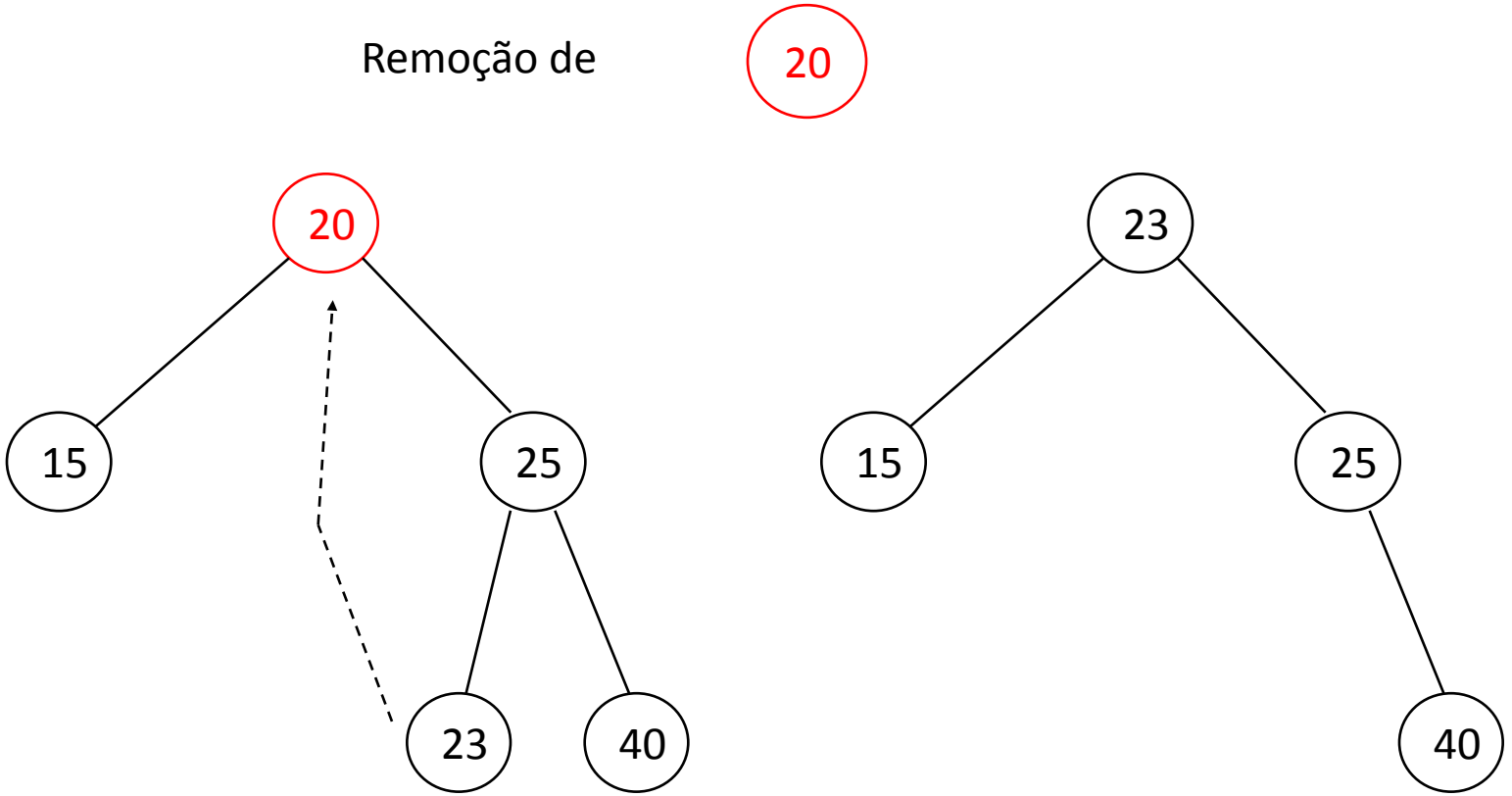




REMOÇÃO elemento com um filho



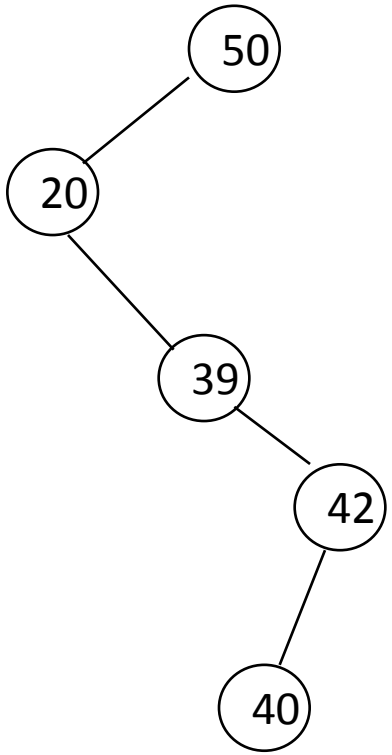
REMOÇÃO elemento com 2 filhos



Árvore – lista

Árvore binária de pesquisa pode degenerar numa lista e assim o tempo de pesquisa passa a ser como numa lista

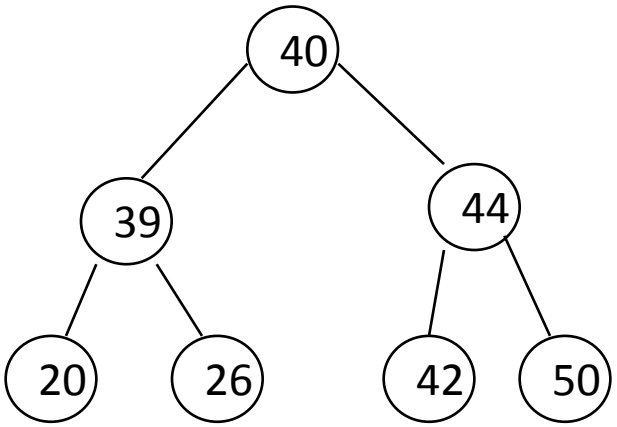
Sequência das inserções na árvore: 50,20,39,42,40

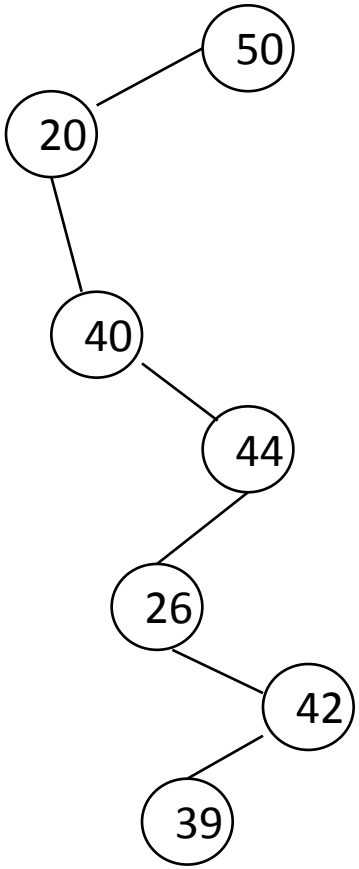


Neste caso, não existe vantagem em ter uma árvore

Árvore Completas

Árvores completas minimizam o número de comparações efectuadas no pior caso para pesquisar um elemento.





Uma árvore diz-se completa se o seu número de nós for igual a $2^h - 1$
Sendo h altura da árvore

Permite a representação
V[0] raíz; filhos de v[i] são v[2*i+1] e v[2*i+2]

Árvores Balanceadas AVL

Uma árvore binária diz-se balanceada AVL sse

para todo o nó da árvore a diferença entre as alturas das suas sub-árvores não excede 1.

Uma árvore que contenha um nó que não satisfaça esta condição é uma árvore desregulada.

A diferença entre as alturas das sub-árvores de um nó é designada por factor de balanceamento do nó.

Adel'son-Vel'skii e Landis (1962)

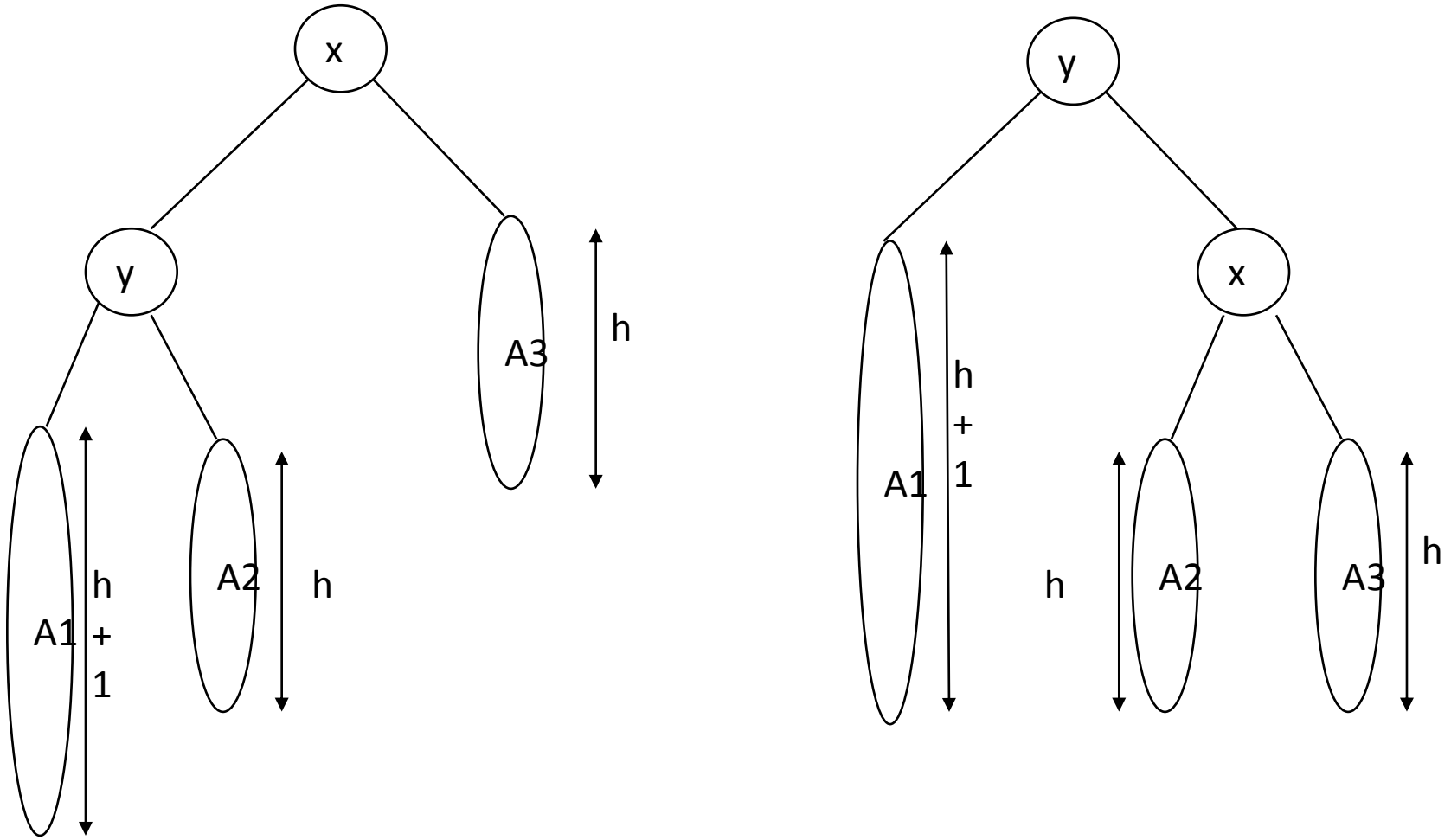
AVL

A manutenção do balanceamento de uma árvore é feita por diversas operações de rotação

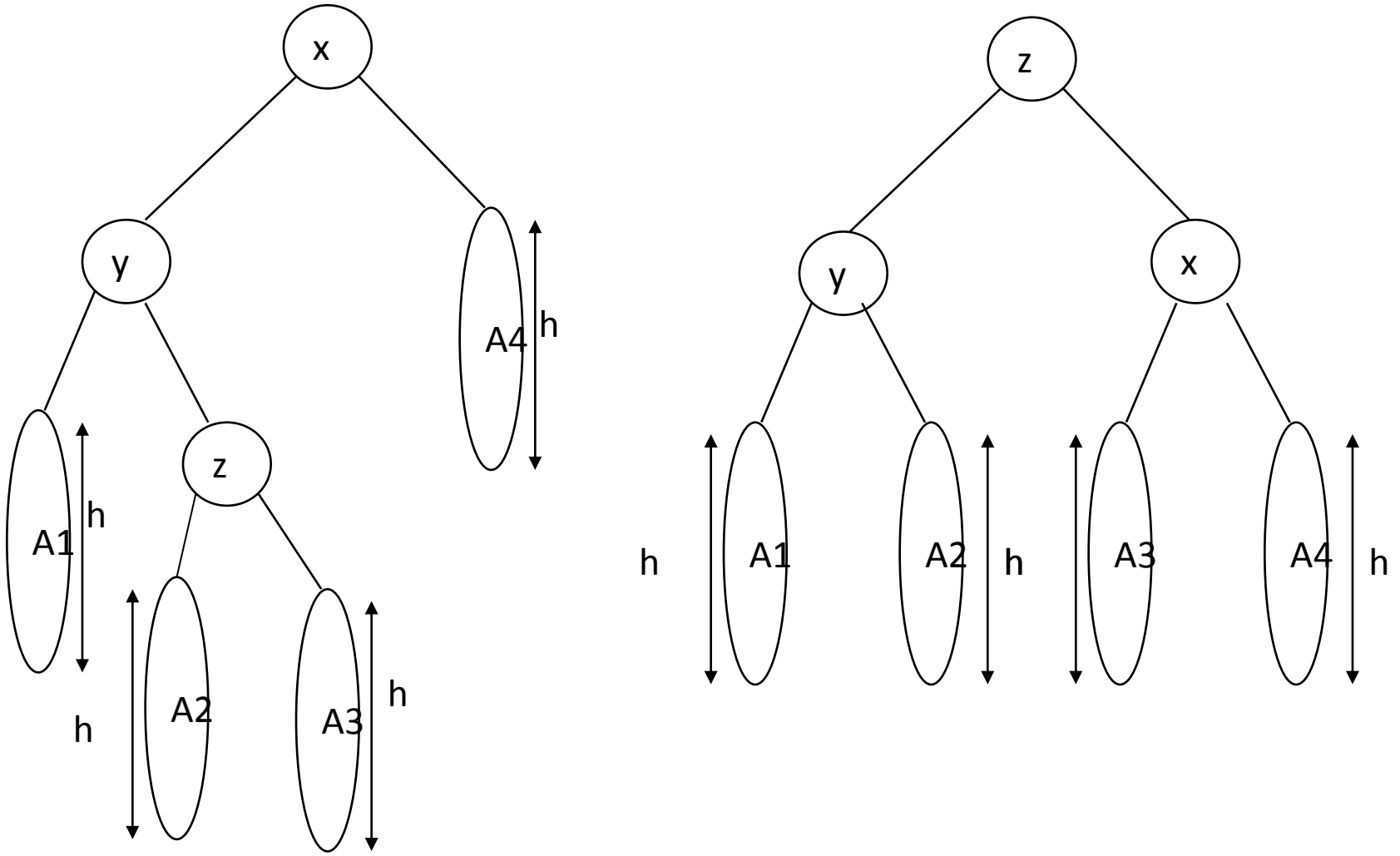
- Simples ou dupla
- à direita e à esquerda

Implicam consumo de tempo, por isso há que ponderar o seu uso.
Uma possibilidade para conseguir menos consumo de tempo é manter um registo em cada nó com seu factor de balanceamento, mas consome mais memória e é mais complexo.

Rotação simples à direita



Rotação dupla à direita



Sites interessantes com animação sobre árvores

http://www.csi.uottawa.ca/~stan/csi2514/applets/avl/BT.html

http://students.ceid.upatras.gr/~perisian/data_structure/HeapSort/heap_applet.html

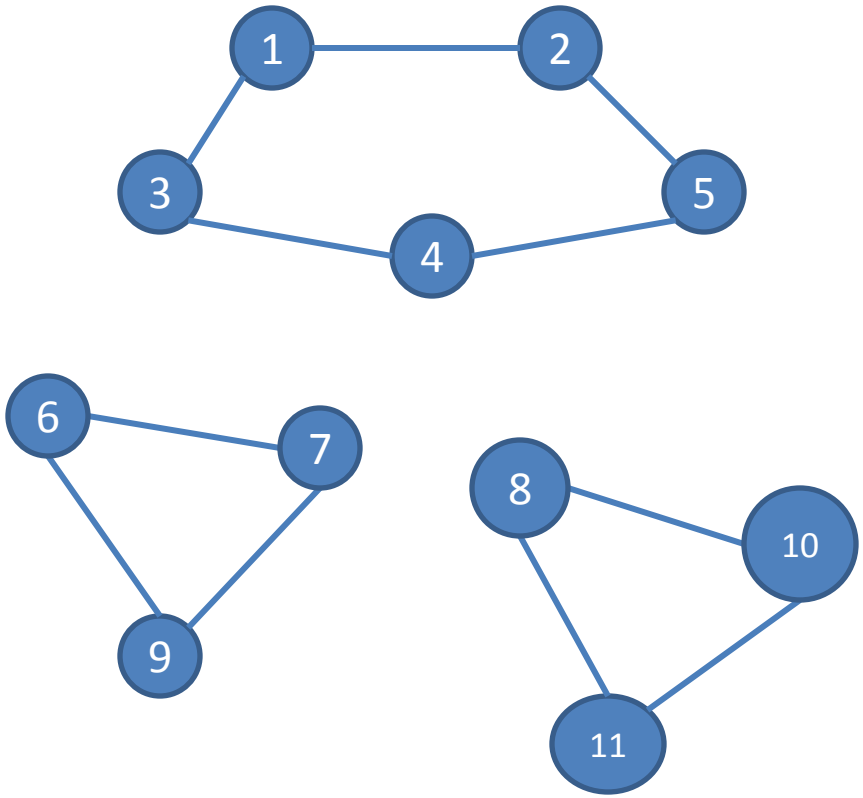
http://www.cse.iitk.ac.in/users/dsrkg/cs210/applets/sortingII/heapSort/heapSort.html

http://rw4.cs.uni-sb.de/projects/ganimal/HEAPSORT/index.html

Grafos

Grafo G=(N,E)

N conjunto de vértices
E conjunto de arestas



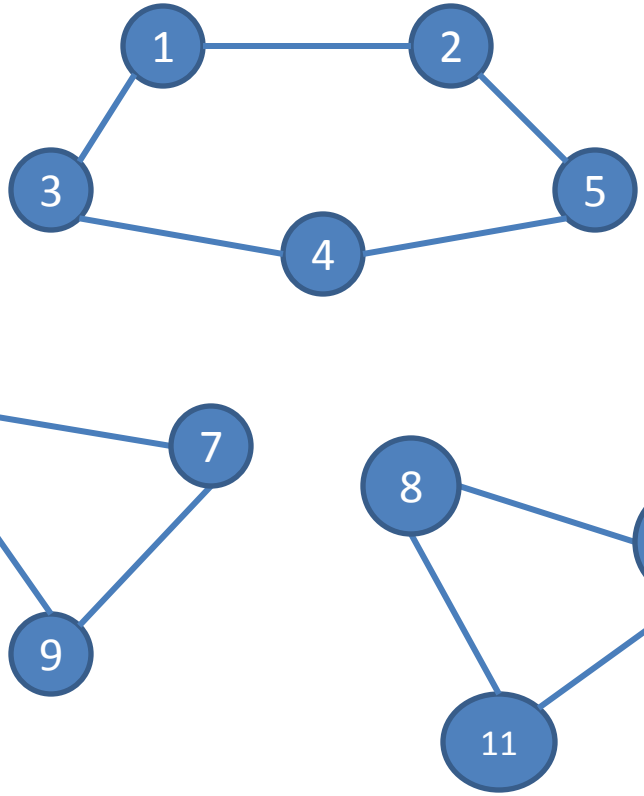
Matriz de adjacência

	1	2	3	4	5	6	7	8	9	10	11
1		1	1								
2	1				1						
3	1			1							
4			1		1						
5		1		1							
6							1		1		
7						1			1		
8										1	1
9						1	1				
10								1			1
11								1		1	

Grafo G=(N,E)

N conjunto de vértices

E conjunto de arestas



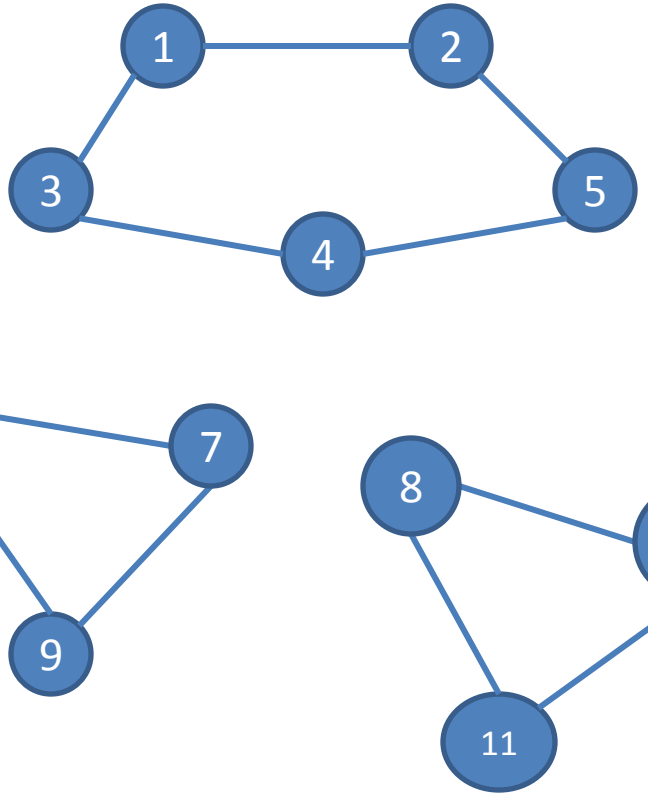
Matriz de incidência

	1	2	3	4	5	6	7	8	9	10	11
1	1	1									
2	1		1								
3		1		1							
4				1	1						
5			1		1						
6						1		1			
7						1	1				
8									1	1	
9							1	1			
10									1		1
11										1	1

Grafo G=(N,**E**)

N conjunto de vértices

E conjunto de arestas



Lista de adjacência

AdjList[1] → 2 → 3

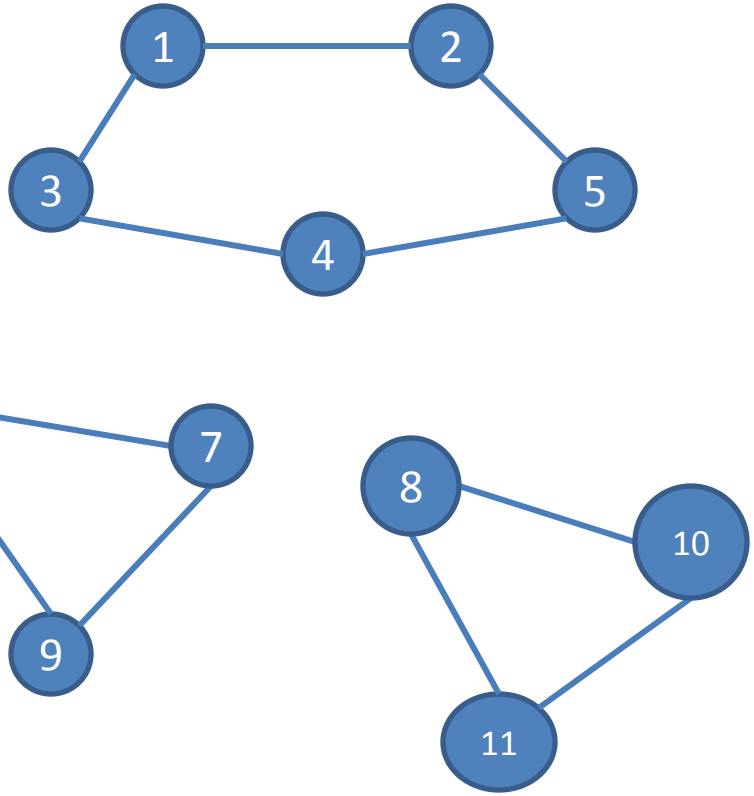
AdjList[2] → 1 → 5

AdjList[11] → 8 → 10

Grafo G=(N,E)

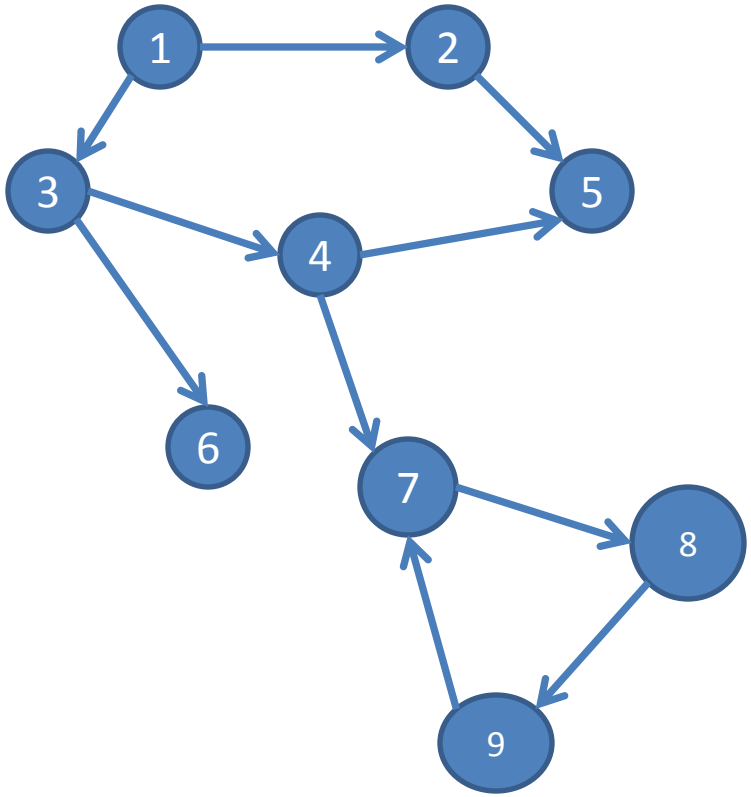
N conjunto de vértices

E conjunto de arestas



Grafos orientados

Grafo G=(N,E)



Matriz de adjacência

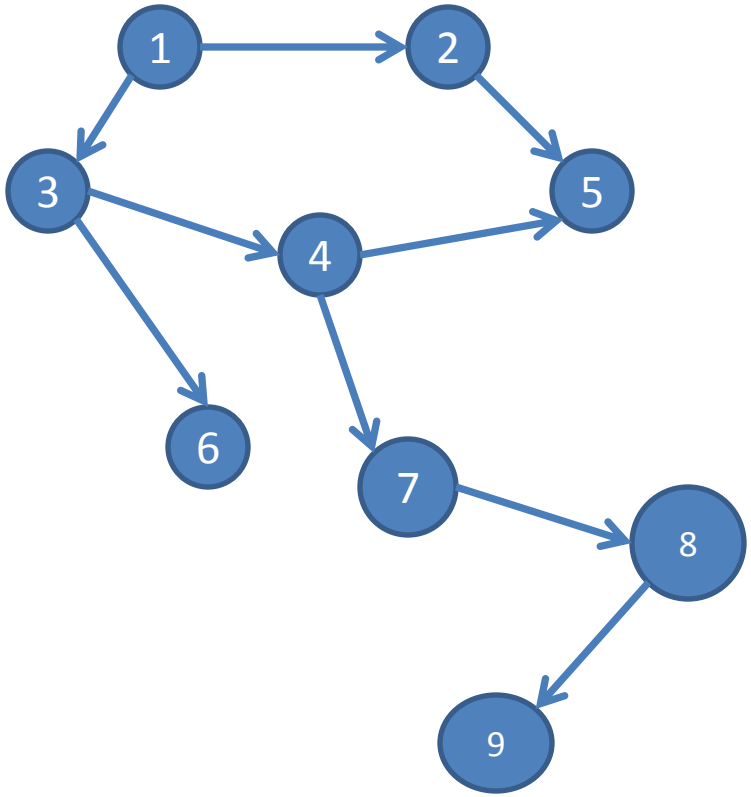
	1	2	3	4	5	6	7	8	9
1		1	1						
2					1				
3				1		1			
4							1		
5									
6									
7								1	
8									1
9							1		

Um grafo é **acíclico** se não existirem ciclos. Exemplo de ciclo: 7 -> 8 -> 9 -> 7

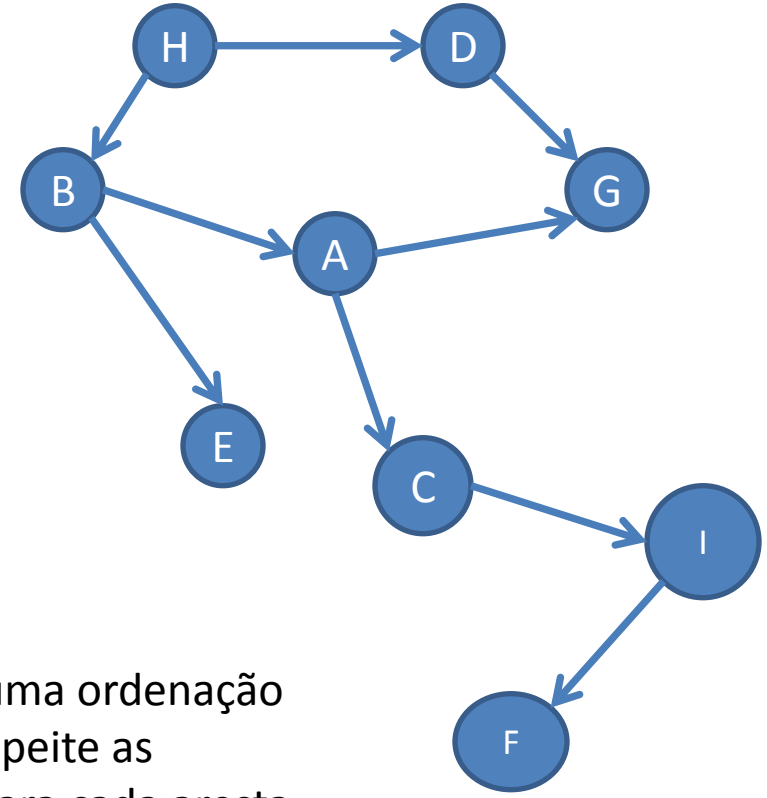
Grafos orientados acíclicos (GAO)

Matriz de adjacência

	1	2	3	4	5	6	7	8	9
1		1	1						
2					1				
3				1		1			
4							1		
5									
6									
7								1	
8									1
9									



Topological sorting de GAO



Problema: Encontrar uma ordenação linear do grafo que respeite as precedências. Isto é, para cada aresta orientada v->w, o vértice v aparece primeiro que w na ordenação.

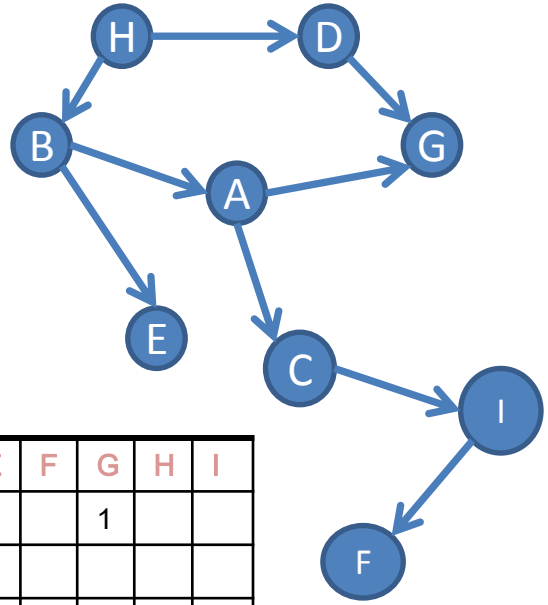
Solução:

	1	2	3	4	5	6	7	8	9
sort:	H	B	D	E	A	G	C	I	F

Topological sorting de GAO

1º Passo: Identificar os vértices que não tem precedentes.

	A	B	C	D	E	F	G	H	I
grau:	1	1	1	1	1	1	2	0	1



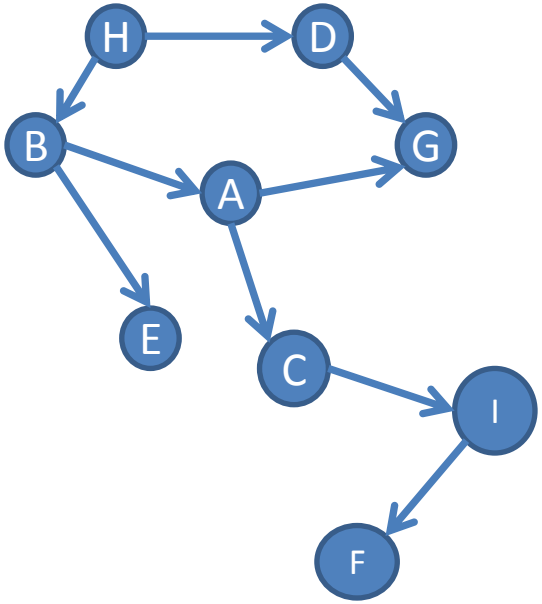
Nota: O vector grau pode ser calculado usando a matriz de adj.

	A	B	C	D	E	F	G	H	I
A			1				1		
B	1				1				
C									1
D							1		
E									
F									
G									
H		1		1					
I						1			

Topological sorting de GAO

2º Passo: criar uma fila com os vértices que tem grau zero.

Fila:	H							



3º Passo: retirar o 1º elemento da fila e adiciona-lo ao vector **sort.**

Sort:	H								

4º Passo: actualizar o vector **grau**

	A	B	C	D	E	F	G	H	I
Grau:	1	0	1	0	1	1	2	0	1

5º Passo: adicionar os novos vértices de grau zero à fila

Fila:	B	D						

6º Passo: voltar ao 3º passo até não restarem vértices na fila

Topological sorting de GAO

Algoritmo:

1. Criar um vector **grau**
2. Inicializar uma **fila** com todos os vértices de grau zero
3. Enquanto existirem vértices na fila
    1. Dequeue um vértice e adicionar ao vector **sort**
    2. Actualizar o vector **grau**
    3. Enqueue os novos vértices de grau zero