Árvores

SCC-202 – Algoritmos e Estruturas de Dados I

Lucas Antiqueira

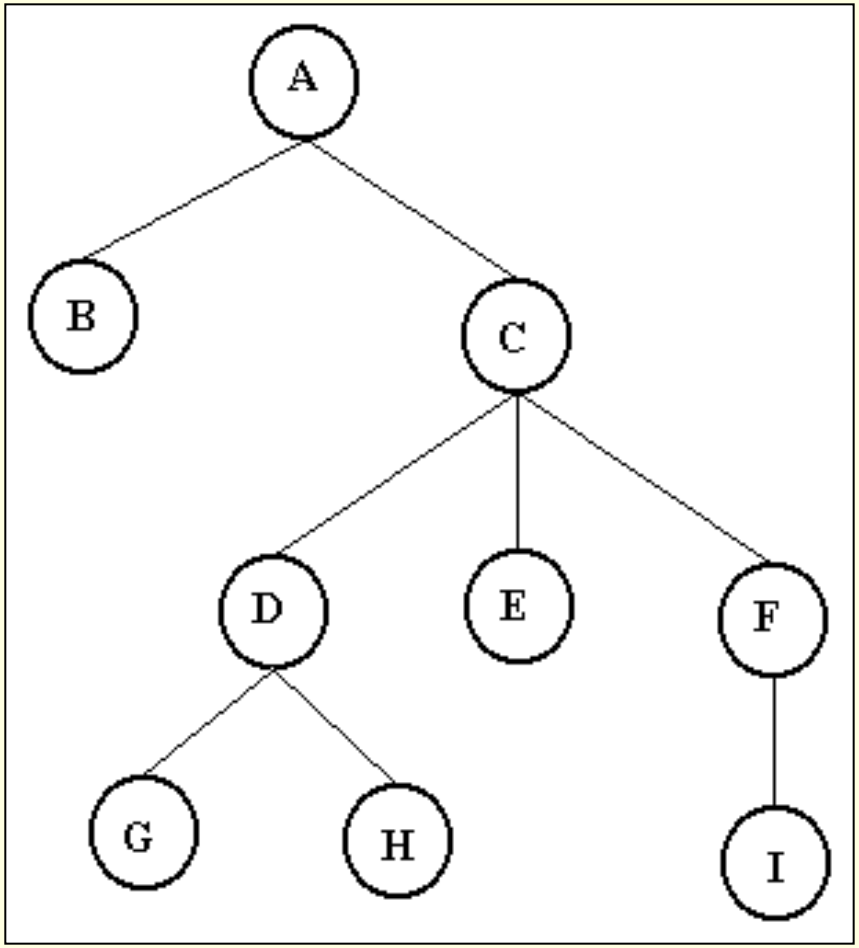
Listas e árvores

- Listas lineares
  - Um nó após o outro, adjacentes
  - Nó sucessor e antecessor

- Diversas aplicações necessitam de estruturas mais complexas do que as listas estudadas até agora
  - Listas não lineares: árvores, grafos, etc.

Árvores

- Exemplo



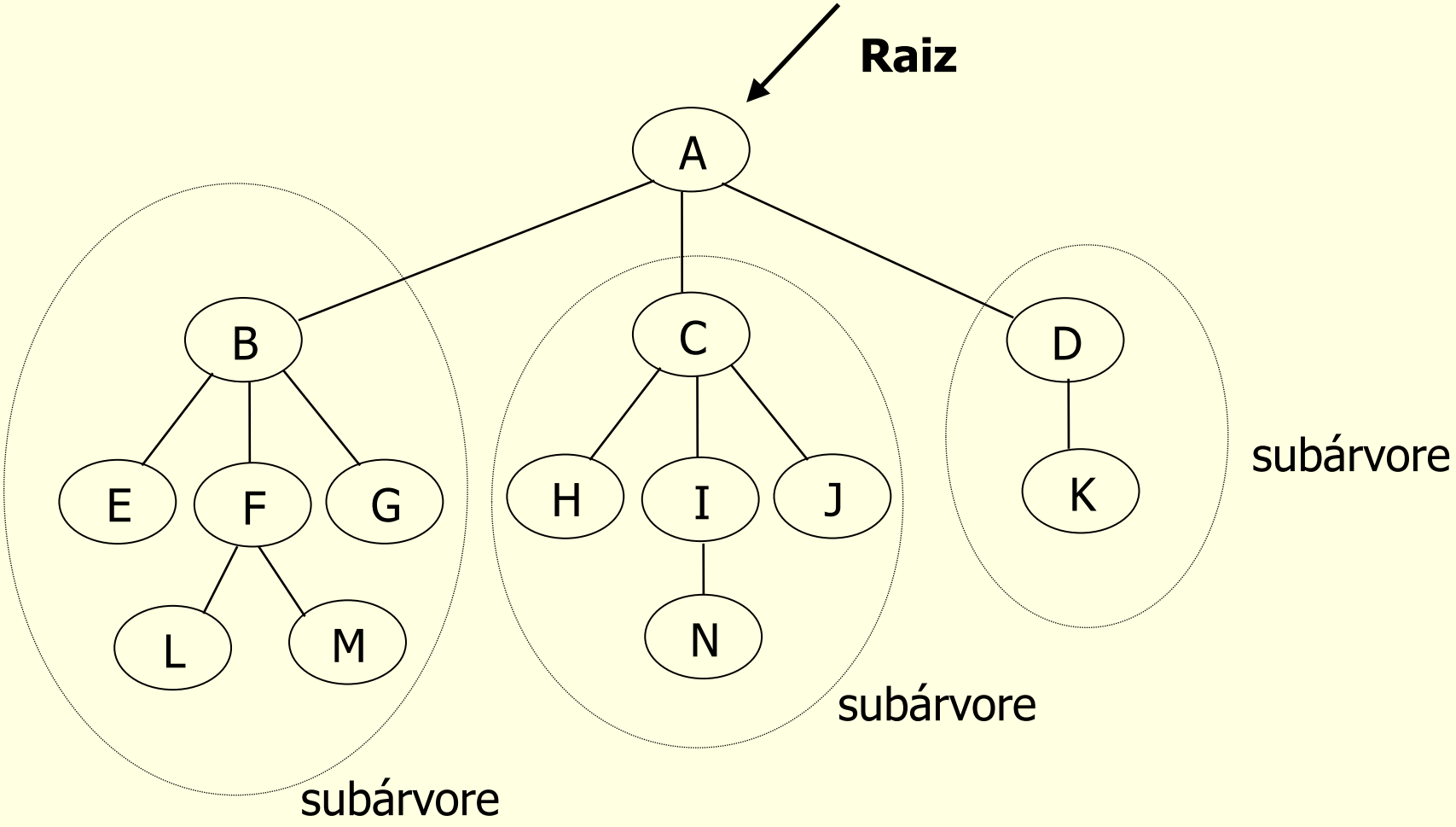
Árvores

- Motivações para usá-las
  - Inúmeros problemas podem ser representados e tratados por árvores
  - Árvores admitem tratamento computacional eficiente quando comparadas a estruturas mais genéricas como os grafos (os quais, por sua vez são mais flexíveis e, portanto, mais complexos)
  - Ótimas para busca

Árvores

- **Definição**
  - Uma árvore *T*, ou simplesmente uma árvore, é um conjunto finito de elementos denominados nós ou vértices tal que
    - *T* é a árvore dita vazia, ou
    - Existe um nó especial *R*, chamado raiz de *T*; os nós restantes constituem um único conjunto vazio ou são divididos em *m* conjuntos não vazios que são as subárvores de *R*, sendo que cada subárvore é, por sua vez, uma árvore

Árvores



Árvores

- Nós filhos, pais, tios, irmãos e avô
  - Seja V o nó raiz de uma árvore T
    - Os nós raízes w1, w2, ... wj das subárvores de V são chamados filhos de V
    - V é chamado pai de w1, w2, ... wj
    - Os nós w1, w2, ...wj são irmãos
    - Se Z é filho de w1, então w2 é tio de Z e V é avô de Z

Árvores

- Grau de saída, descendente e ancestral
  - O número de filhos de um nó é chamado grau (de saída) desse nó
  - Se X pertence à subárvore V de T, então X é descendente de V e V é ancestral, ou antecessor, de X

Árvores

- Nó folha e nó interior
  - Um nó que não possui descendentes é chamado de nó folha, ou seja, um nó folha é aquele com grau de saída nulo ou zero
  - Um nó que não é folha (isto é, possui grau de saída diferente de zero) é chamado nó interior, nó interno ou, ainda, nó intermediário

Árvores

- Grau de uma árvore
  - O grau de uma árvore é o máximo entre os graus de saída de seus nós
  - É o máior número de filhos

Árvores

- Floresta
  - Uma floresta é um conjunto de zero ou mais árvores

Árvores

- Caminho, comprimento do caminho

  - Uma seqüência de nós distintos v1, v2, ..., vk, tal que existe sempre entre nós consecutivos (isto é, entre v1 e v2, entre v2 e v3, ... , v(k-1) e vk) a relação "é filho de" ou "é pai de" é denominada um caminho na árvore; diz-se que v1 alcança vk e que vk é alcançado por v1

  - Um caminho de vk vértices é obtido pela seqüência de k-1 pares; o valor k-1 é o comprimento do caminho

Árvores

- Nível (ou profundidade) e altura de um nó
  - O nível de um nó é o número de nós do caminho da raiz até o nó
  - O nível da raiz é portanto 1
  - A altura de um nó V é o número de nós no maior caminho de V até qualquer um de seus descendentes
  - As folhas têm altura 1

Árvores

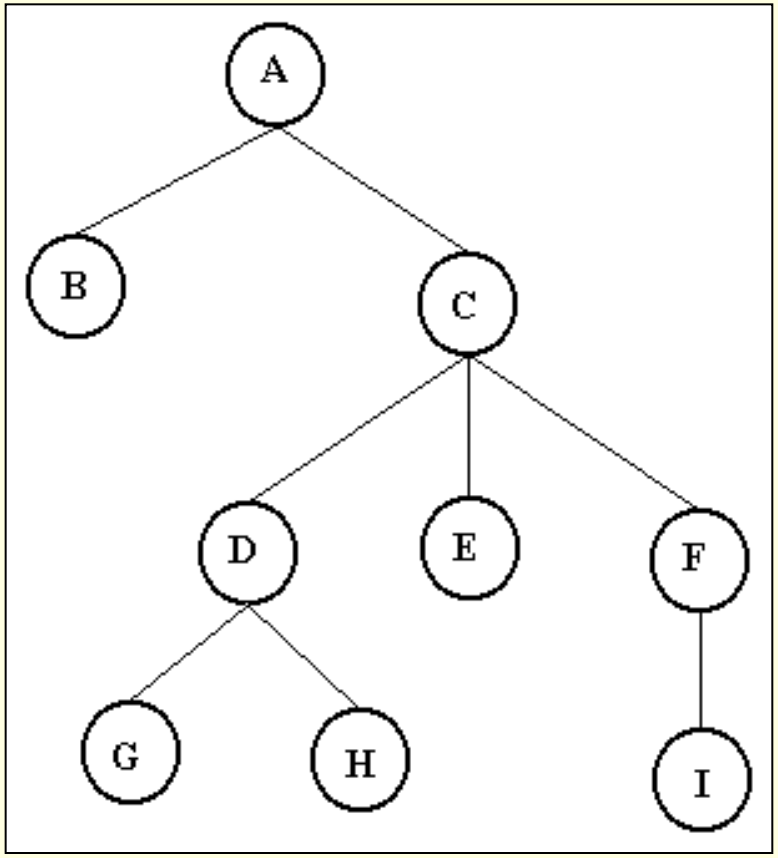
- Altura de uma árvore
  - A altura de uma árvore T é igual ao máximo nível de seus nós
  - Em geral, representa-se a altura de T por h(T) e a altura da subárvore de raiz V por h(V)

Árvores

- Árvore ordenada
  - Uma árvore ordenada é aquela na qual os filhos de cada nó estão ordenados
  - Assume-se ordenação da esquerda para a direita

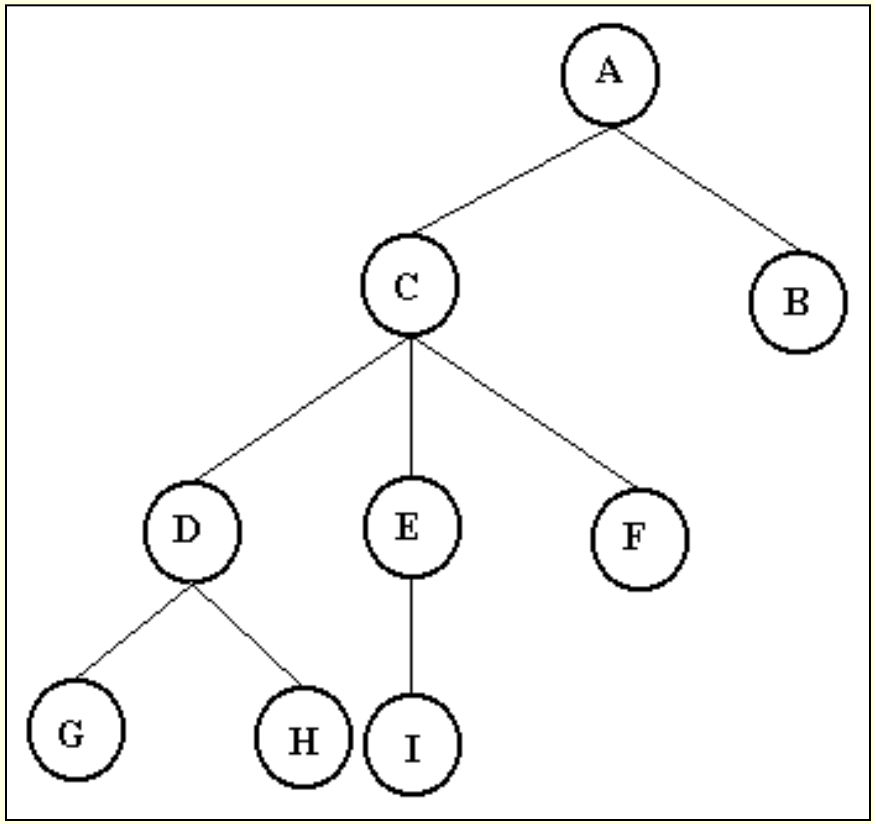
Árvores

- Árvore ordenada



Árvores

- Árvore não ordenada

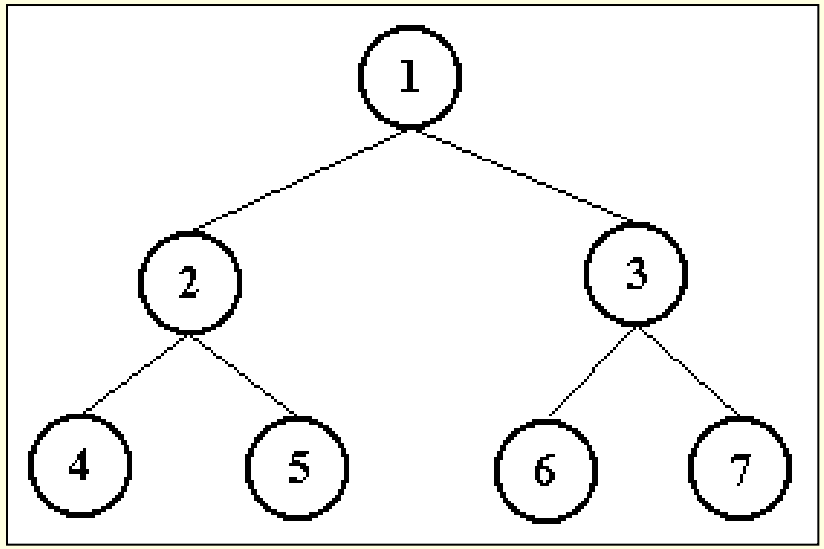


Árvores

- Árvore cheia
  - Uma árvore de grau d é uma árvore cheia se possui o número máximo de nós, isto é, todos os nós tem número máximo de filhos (exceto as folhas, logicamente) e todas as folhas estão na mesma altura

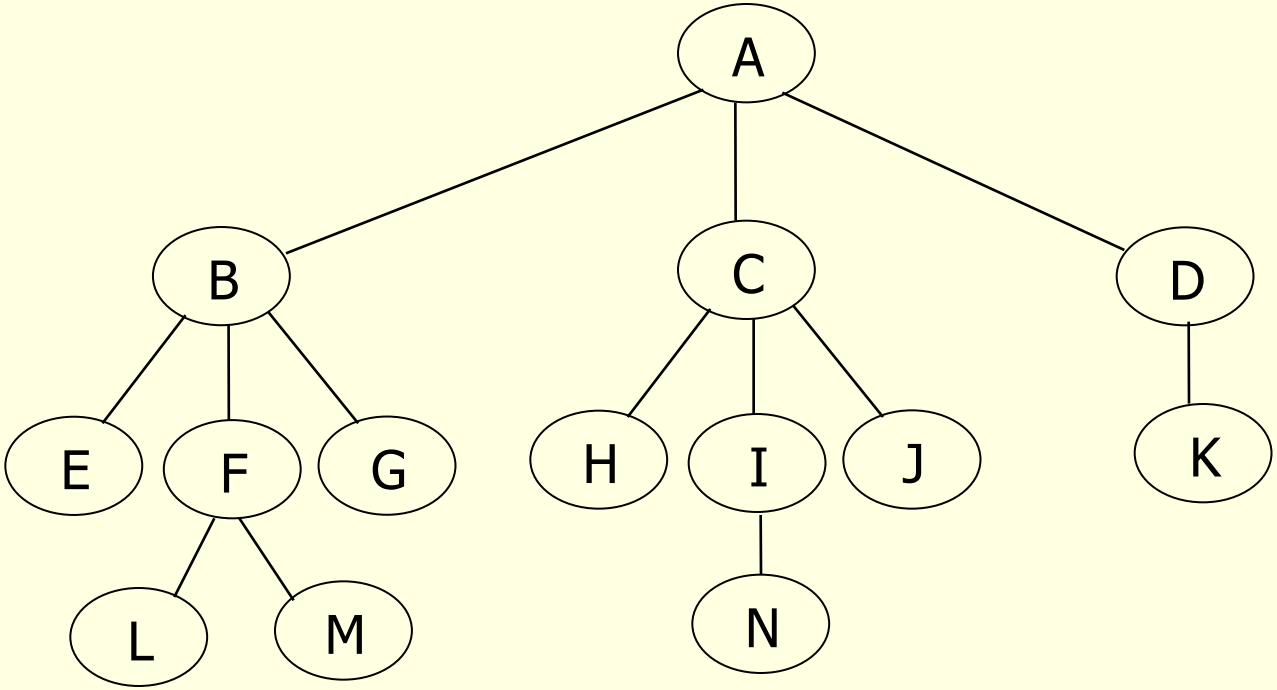
Árvores

- Exemplo de árvore cheia de grau 2



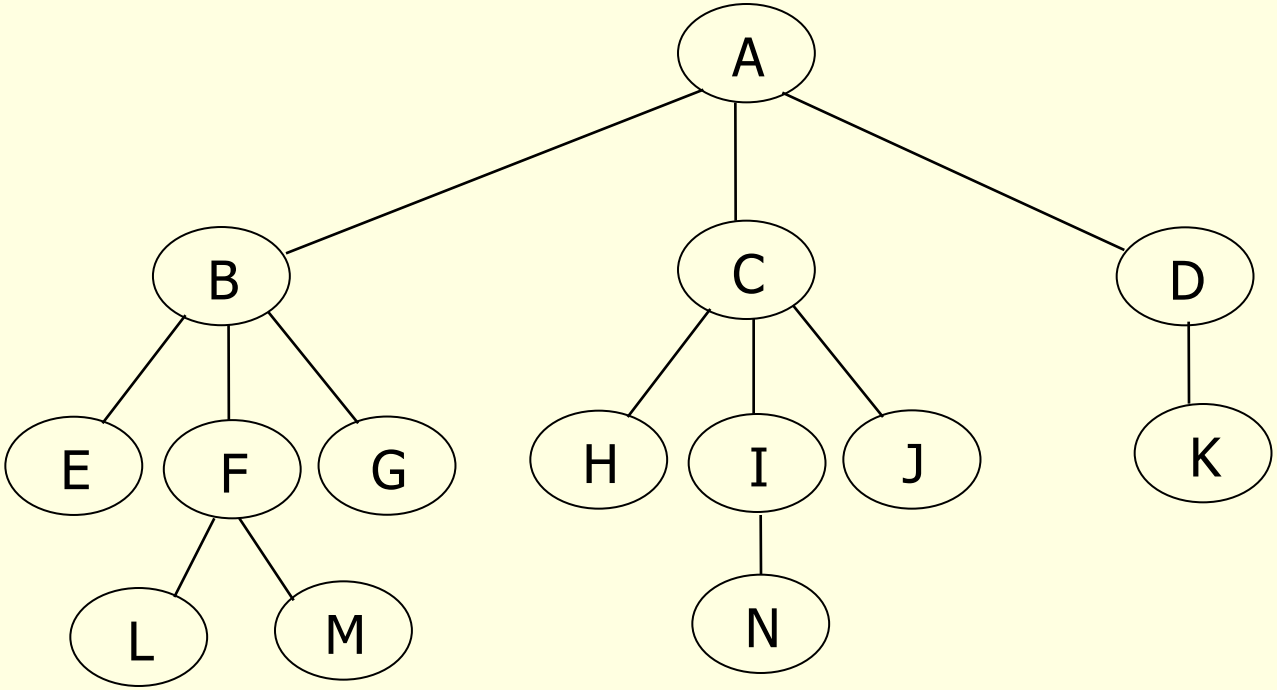
Árvores

- Considere a árvore abaixo
  - Quantas subárvores A tem?



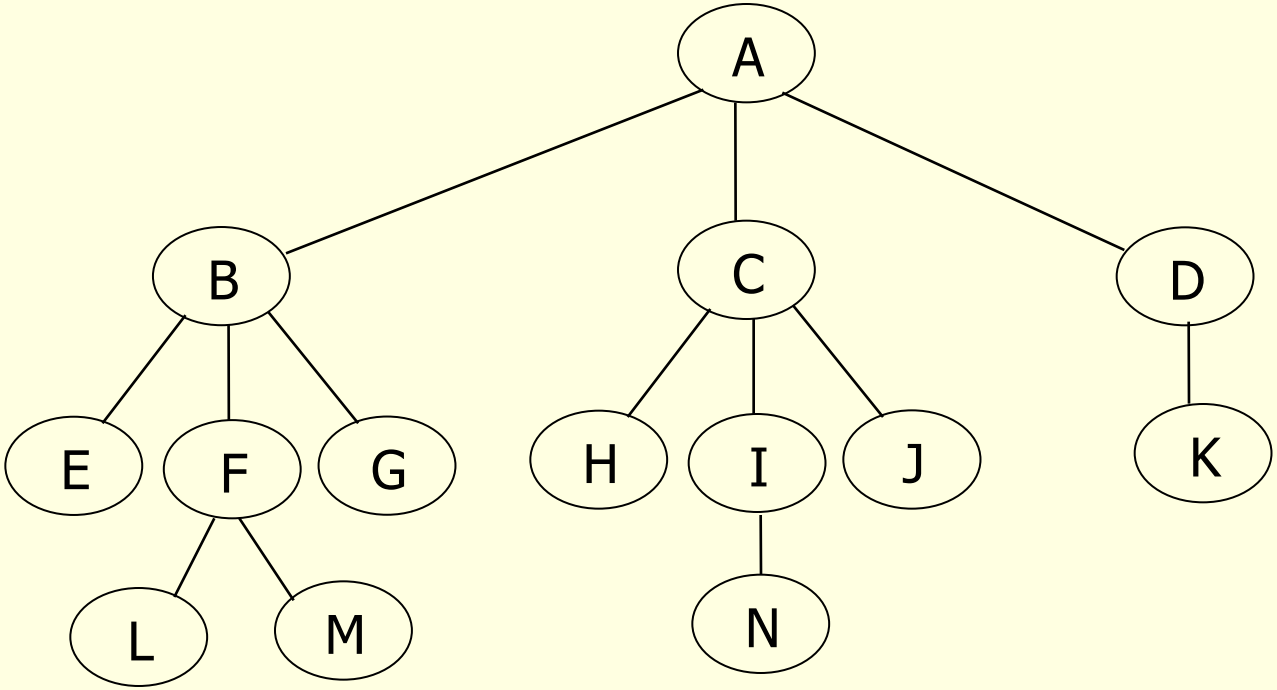
Árvores

- Considere a árvore abaixo
  - Quem são os filhos de A? E os descendentes de A?



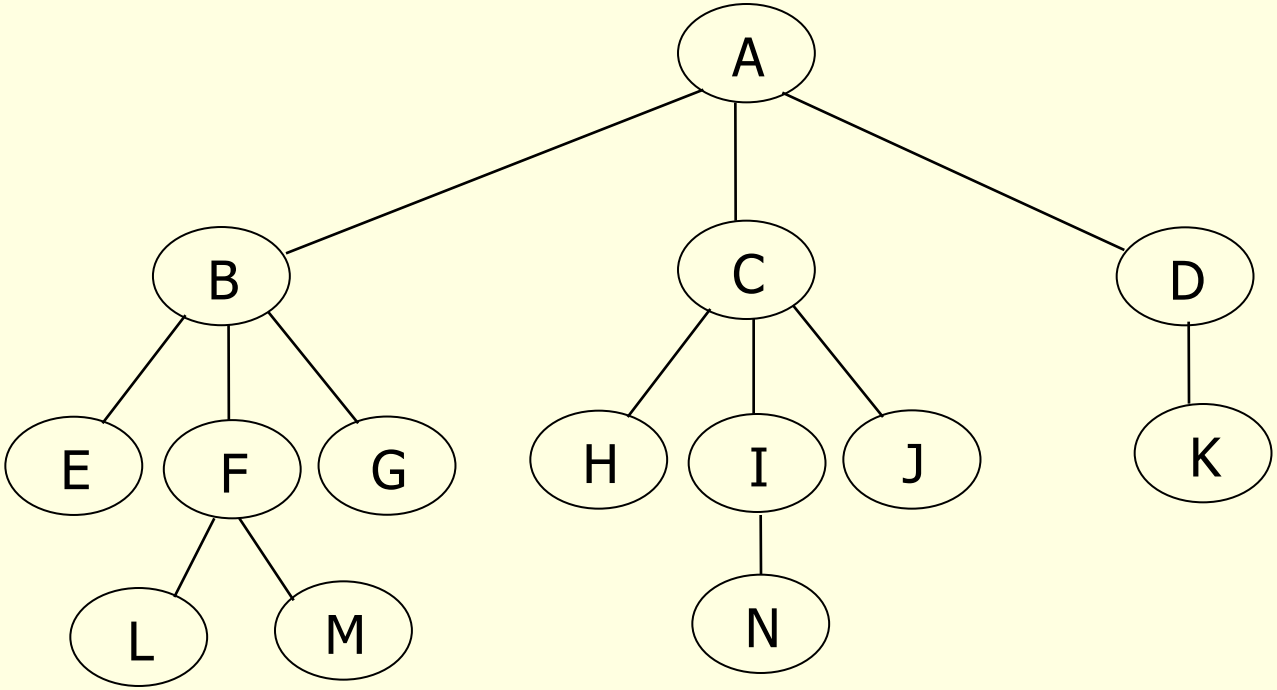
Árvores

- Considere a árvore abaixo
  - Quais são os nós folha dessa árvore?



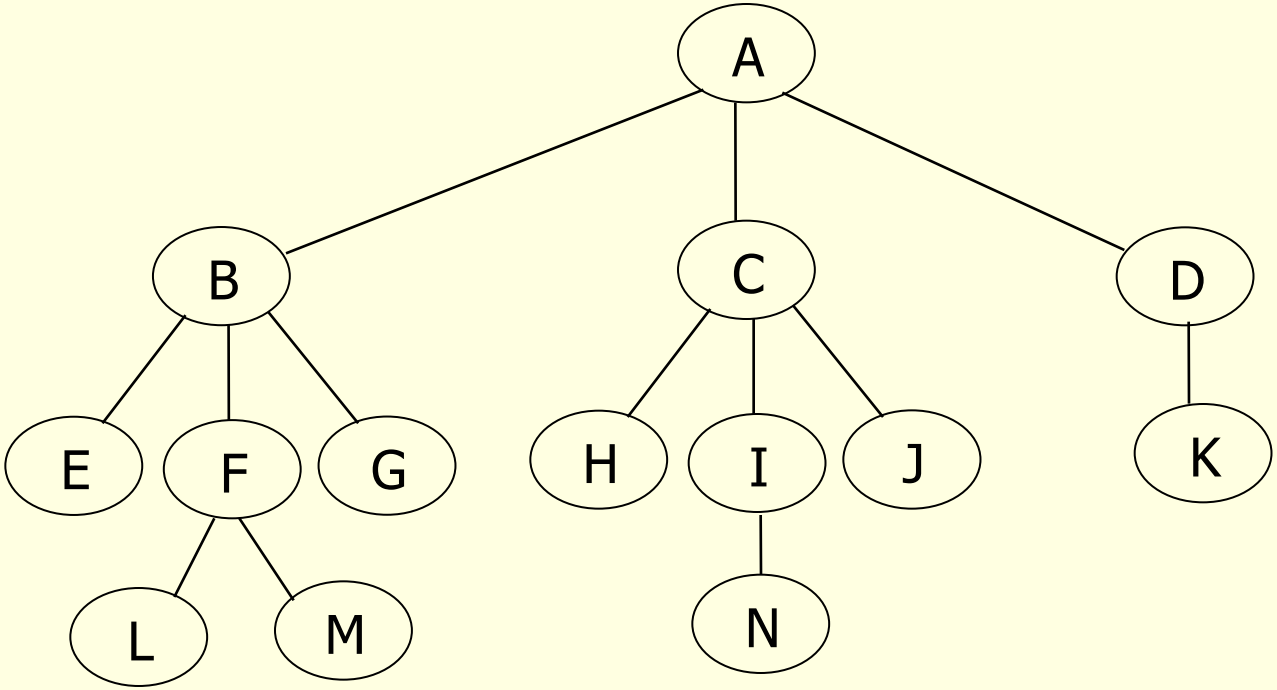
Árvores

- Considere a árvore abaixo
  - Qual o grau dessa árvore?



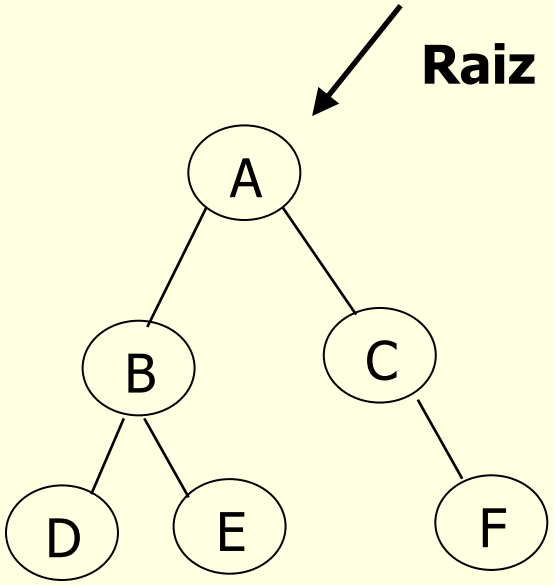
Árvores

- Considere a árvore abaixo
  - Qual a altura dessa árvore?



Árvores binárias

- Árvores com grau 2, ou seja, cada nó pode ter 2 filhos, no máximo



Terminologia:

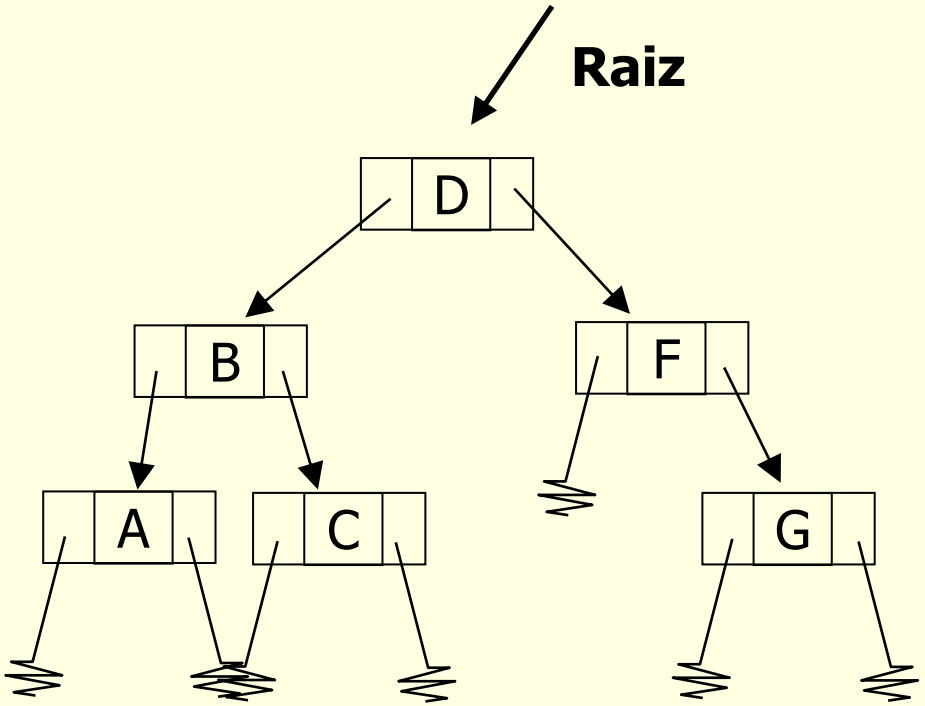
- filho esquerdo
- filho direito
- informação

Árvores binárias

- Exercício
  - Considerando a implementação dinâmica e encadeada, declare a estrutura de cada nó de uma árvore binária

Árvores binárias

- Representação dinâmica e encadeada de uma árvore binária



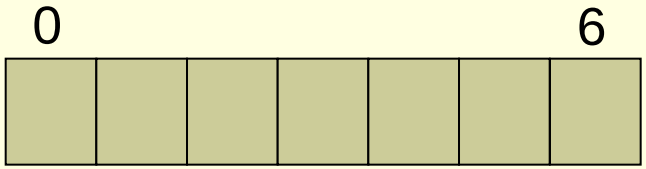
AB vazia

Árvores binárias

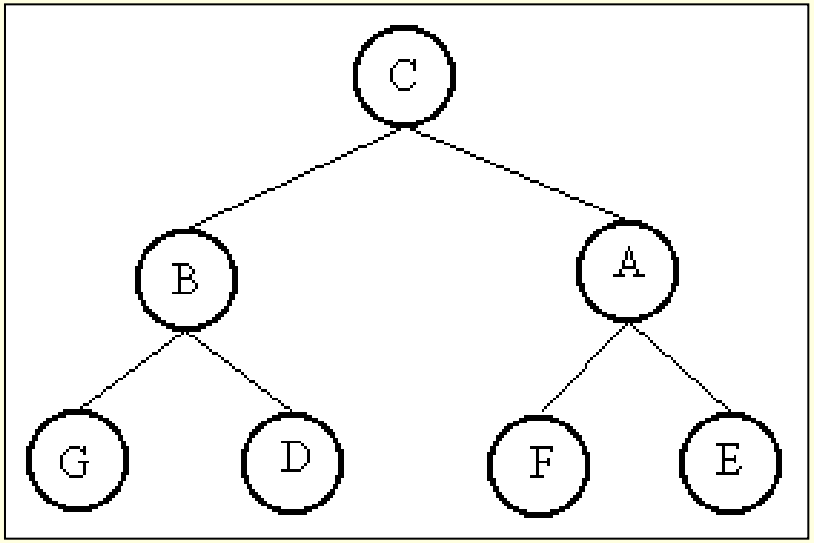
```
typedef int elem;

typedef struct bloco {
   elem info;
   struct bloco *esq, *dir;
} no;

typedef struct {
   no *raiz;
} Arvore;
```

Árvores binárias

- Representação estática e seqüencial de árvores binárias
  - Vetor

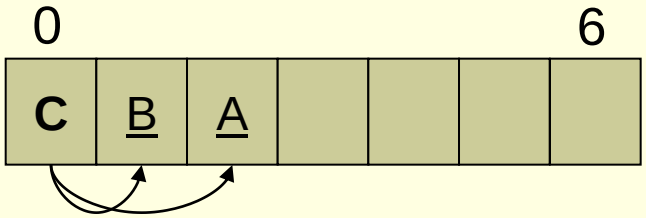


  - Como colocar a árvore abaixo nesse vetor?

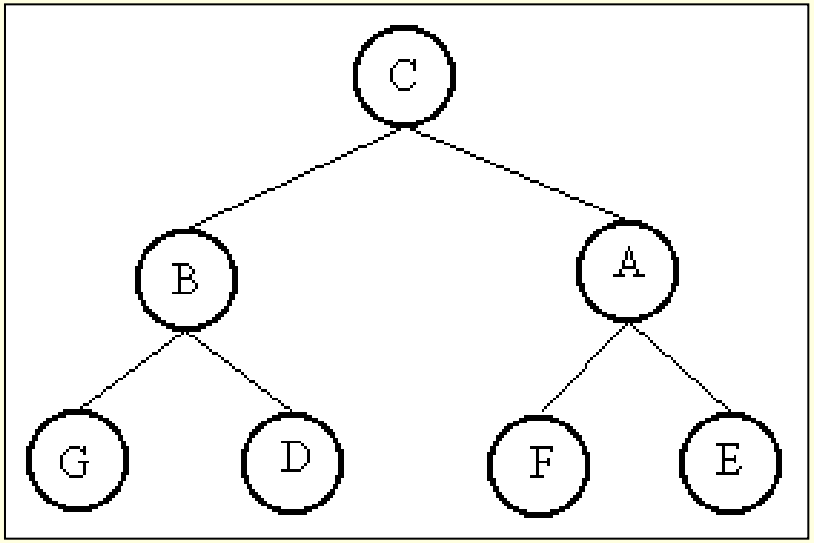


Árvores binárias

- Representação estática e seqüencial de árvores binárias
  - Vetor

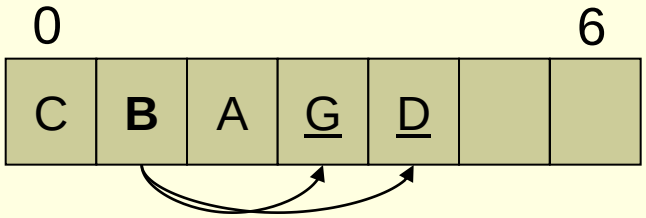


  - Como colocar a árvore abaixo nesse vetor?

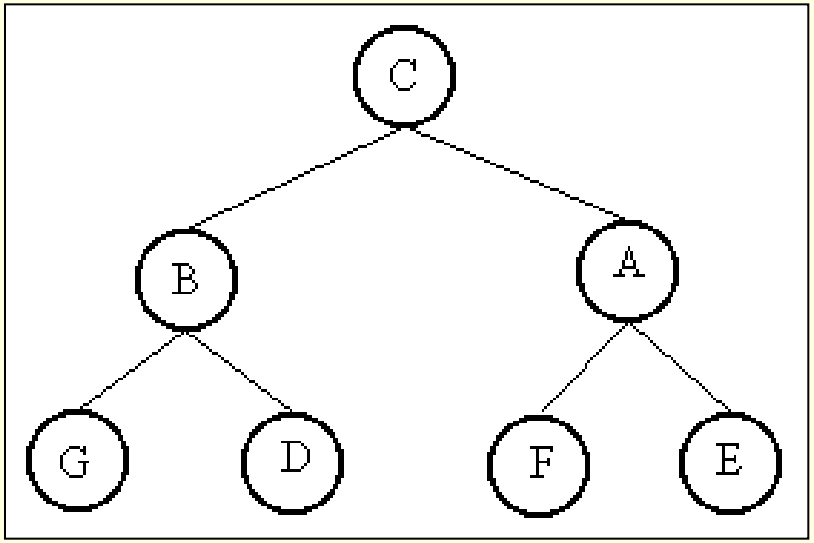


Árvores binárias

- Representação estática e seqüencial de árvores binárias
  - Vetor

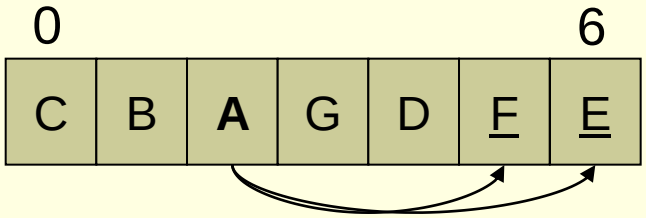


  - Como colocar a árvore abaixo nesse vetor?

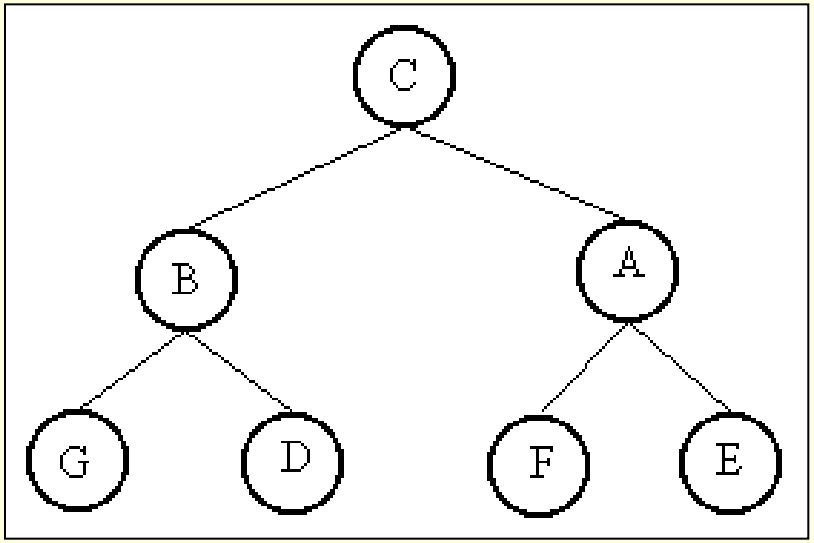


Árvores binárias

- Representação estática e seqüencial de árvores binárias
  - Vetor



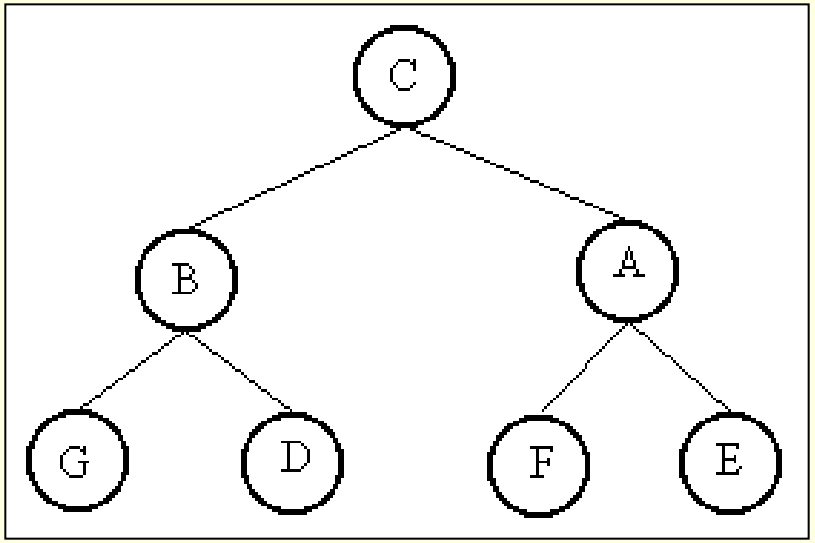
  - Como colocar a árvore abaixo nesse vetor?



Árvores binárias

- Representação estática e seqüencial de árvores binárias

0						6
C	B	A	G	D	F	E

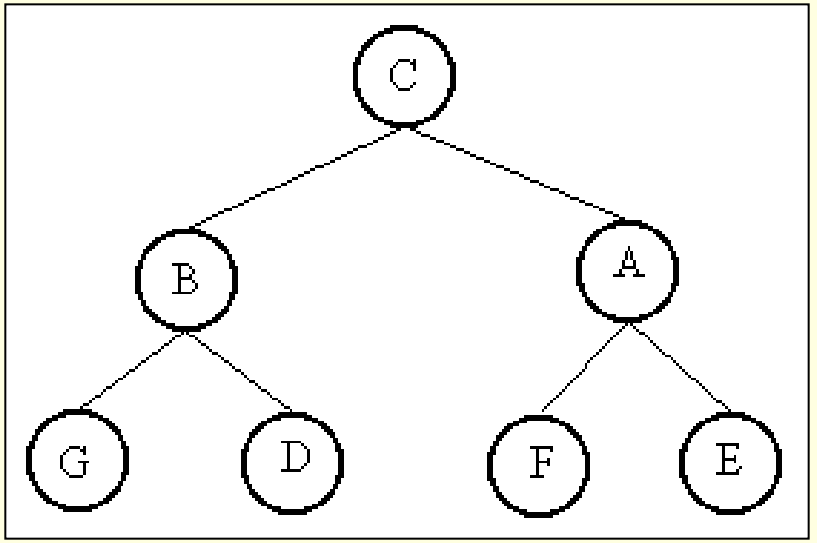


  - Como saber quem é filho de quem?

Árvores binárias

- Representação estática e seqüencial de árvores binárias

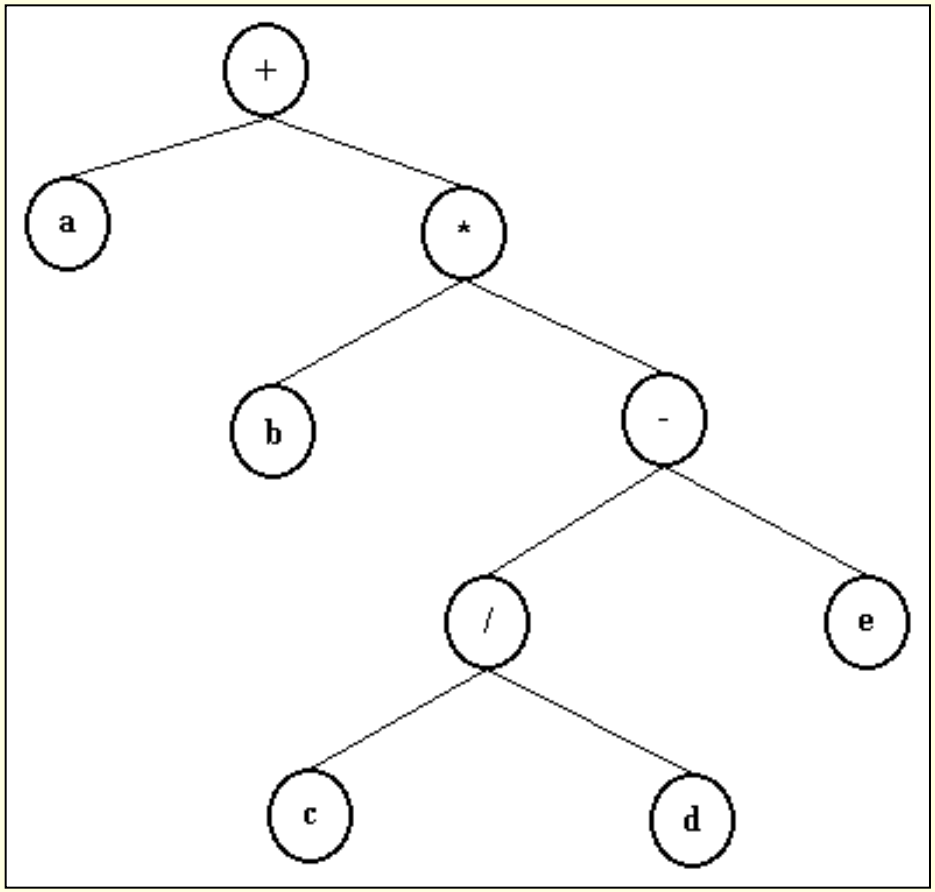
0						6
C	B	A	G	D	F	E



  - Como saber quem é filho de quem?
    - Filhos de i são 2i+1 e 2i+2

Árvores binárias

- Exercício: represente a árvore abaixo em um vetor
  - O que essa árvore representa?



Árvores binárias

- Representação estática e seqüencial de árvores binárias
  - Como fazer a inserção e remoção de elementos nessa representação?
  - É mais fácil ou difícil do que na implementação encadeada e dinâmica? É mais eficiente?
  - E em termos de uso da memória?

Operações em árvores binárias

- Algumas operações do TAD, considerando árvore dinâmica e encadeada
  - Criar árvore
  - Verificar se a árvore está vazia
  - Finalizar árvore
  - Imprimir árvore
  - Determinar altura da árvore
  - Buscar um elemento
  - Buscar pai de um elemento
  - Inserir elemento à esquerda de outro elemento
  - Inserir elemento à direita de outro elemento
  - Remover elemento

Árvores binárias

- Exercício: Implementar o TAD árvore binária

Percurso em árvores binárias

- Percorrer uma árvore visitando cada nó uma única vez gera uma seqüência linear de nós
  - Listagem de todos os elementos
  - Busca por um elemento
- Passa a ter sentido falar em sucessor e predecessor de um nó segundo um determinado percurso

Percurso em árvores binárias

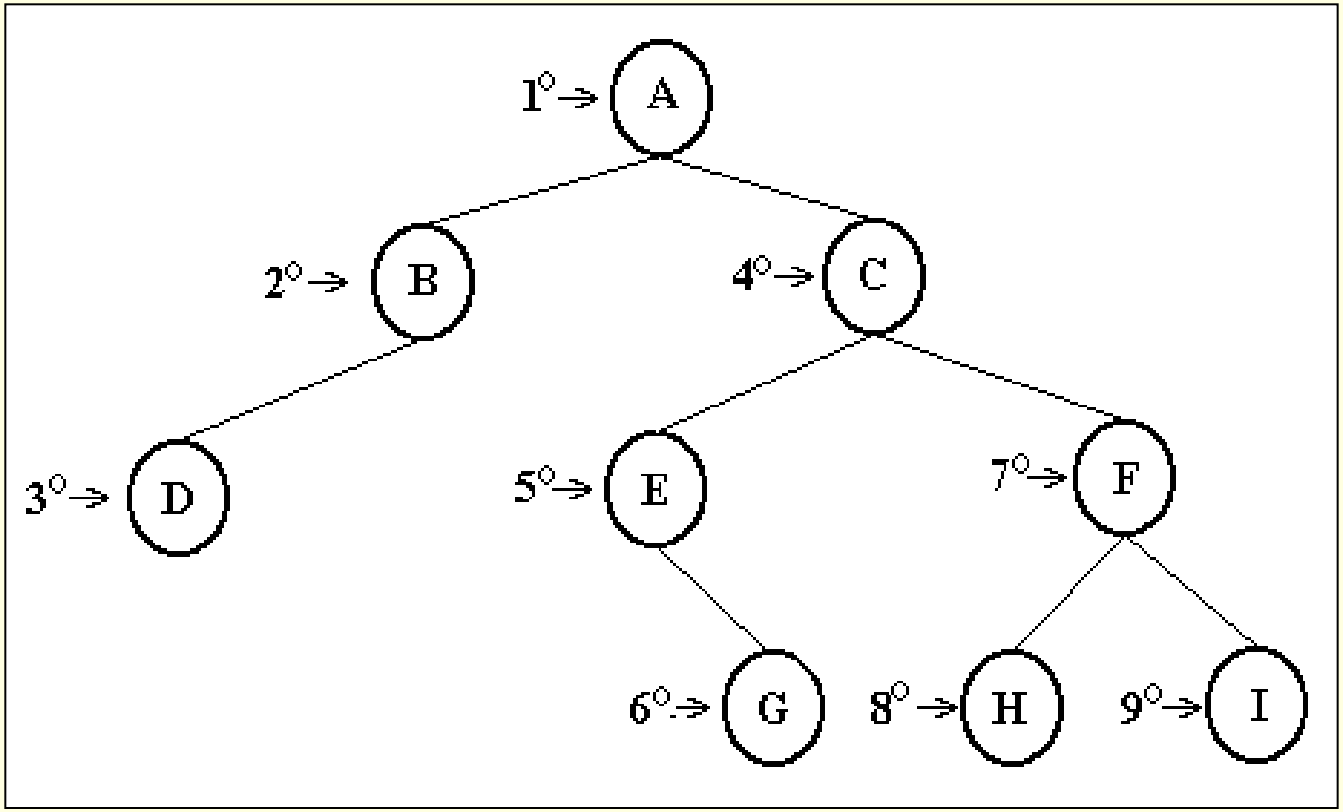
- Há três maneiras de se percorrer árvores binárias
  - Em função da ordem de visitas aos nós
    - Pré-ordem: visita-se nó raiz primeiro e depois as subárvores esquerda e direita, nessa ordem
    - Em-ordem: visita-se subárvore esquerda, nó raiz e subárvore direita, nessa ordem
    - Pós-ordem: visita-se subárvore esquerda, subárvore direita, e, depois, o nó raiz, nessa ordem

Percurso em árvores binárias

- Pré-ordem
  1. se árvore vazia, então fim
  2. visitar o nó raiz
  3. percorrer em pré-ordem a subárvore esquerda
  4. percorrer em pré-ordem a subárvore direita

Percurso em árvores binárias

- Pré-ordem

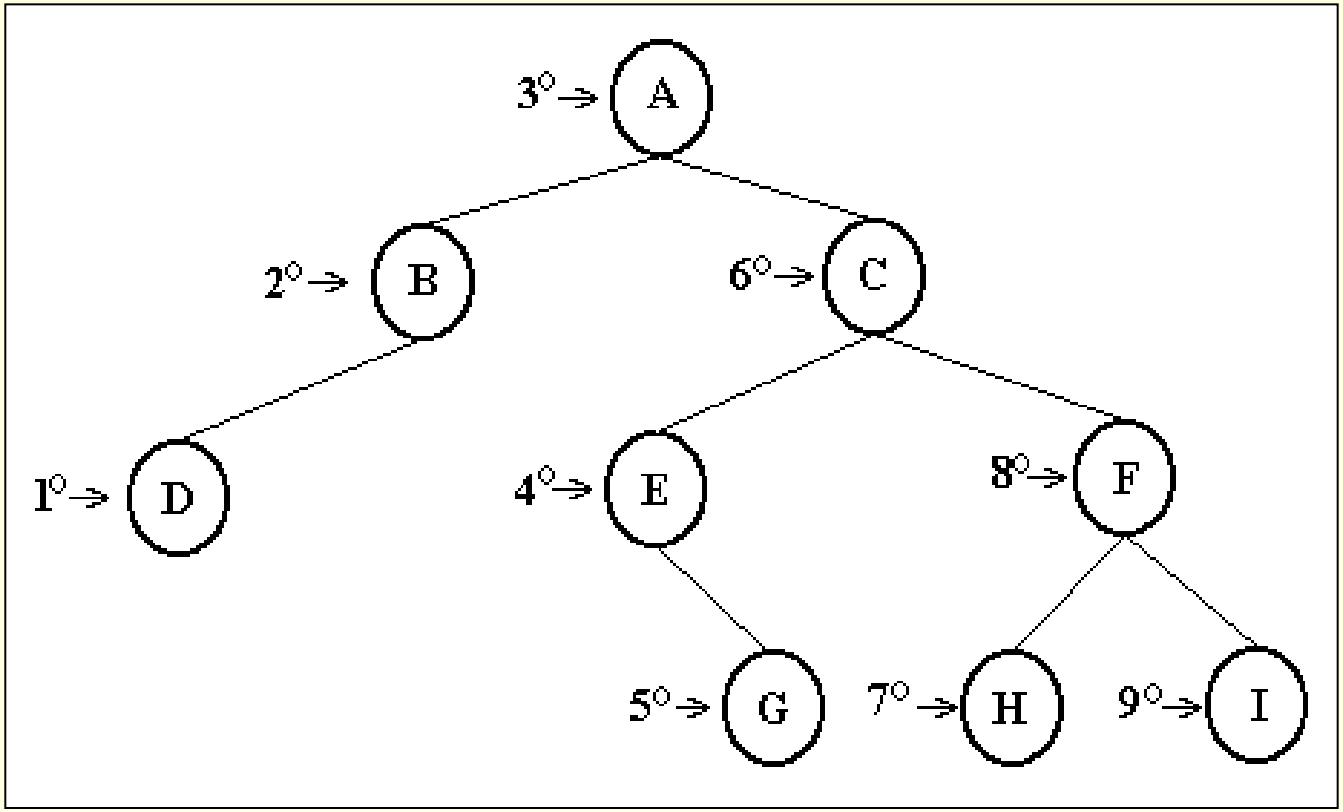


Percurso em árvores binárias

- Em-ordem
  1. se árvore vazia, então fim
  2. percorrer em em-ordem a subárvore esquerda
  3. visitar o nó raiz
  4. percorrer em em-ordem a subárvore direita

Percurso em árvores binárias

- Em-ordem

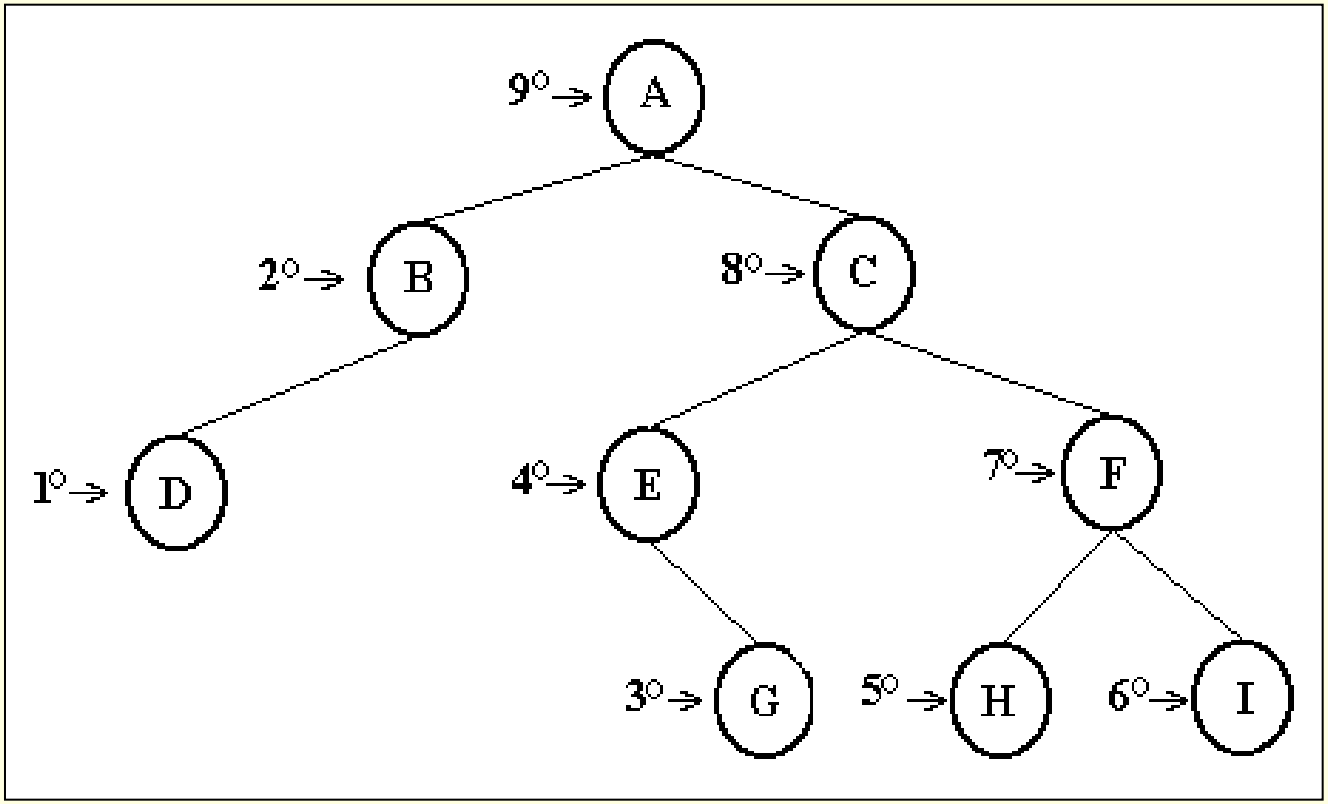


Percurso em árvores binárias

- Pós-ordem
  1. se árvore vazia, então fim
  2. percorrer em pós-ordem a subárvore esquerda
  3. percorrer em pós-ordem a subárvore direita
  4. visitar o nó raiz

Percurso em árvores binárias

- Pós-ordem



Percurso em árvores binárias

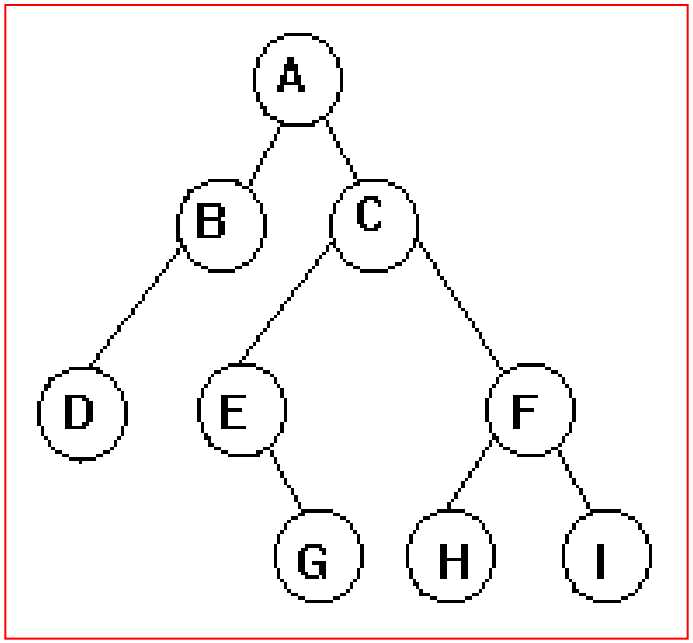
- Exercícios: Implementar sub-rotinas recursivas de percurso pré-ordem, em-ordem e pós-ordem

Percurso em árvores binárias

Implementação não recursiva para o percurso em-ordem

```
...
#include "pilha.h"
...

void EmOrdem(no *raiz) {
   no *p=raiz;
   Pilha s;
   cria_pilha(s);
   do
        while (p!=NULL) {
               push(s,p);
               p=p->esq;
        }
        if (!IsEmpty(s)) {
               pop(s,p);
               printf("%d\n",p->info);
               p=p->dir;
        }
   while ((!IsEmpty(s)) || (p!=NULL));
}
```

Execute a sub-rotina para a árvore abaixo

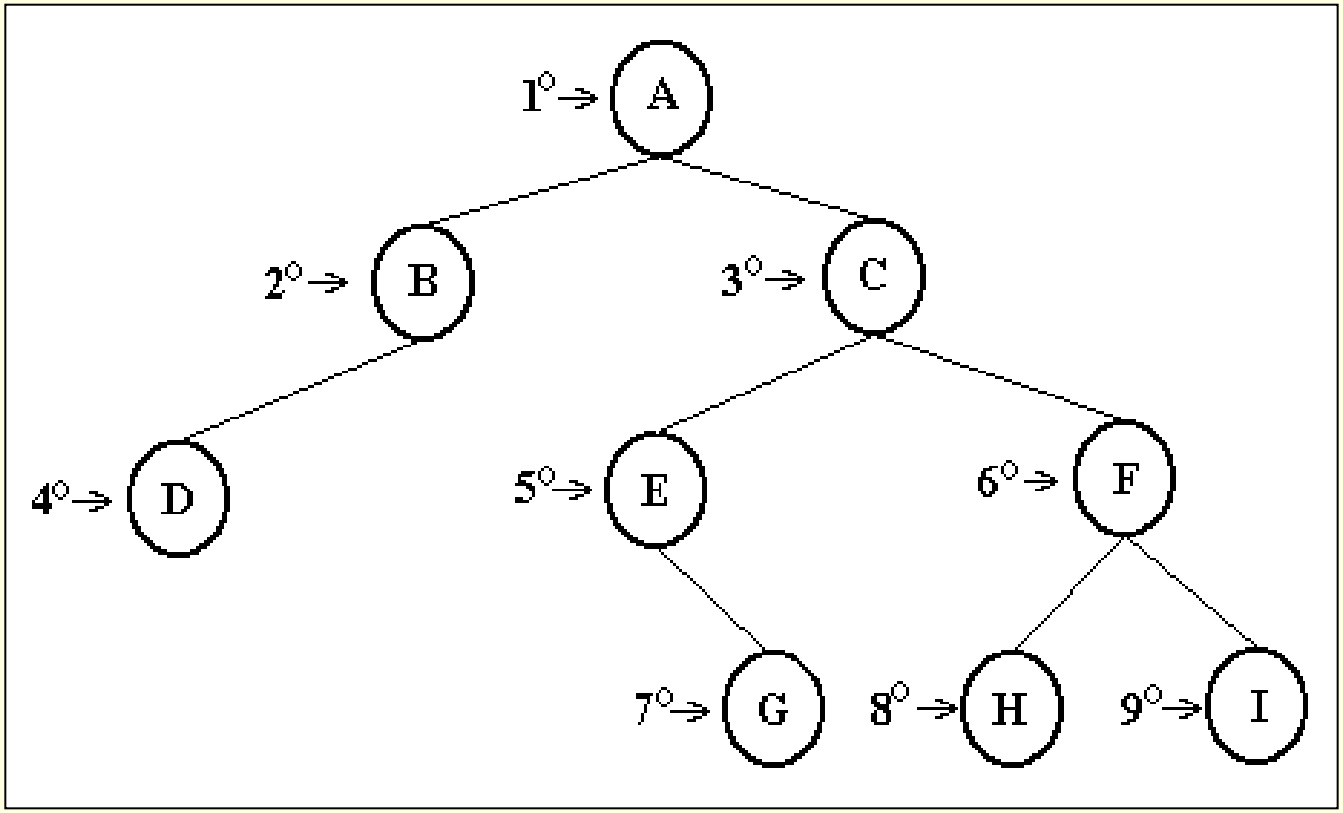


Percurso em árvores

- Outras formas:
  - Busca/percurso em largura
  - Busca/percurso em profundidade
- Independente do tipo de árvore
- Tradicionalmente da esquerda para a direita

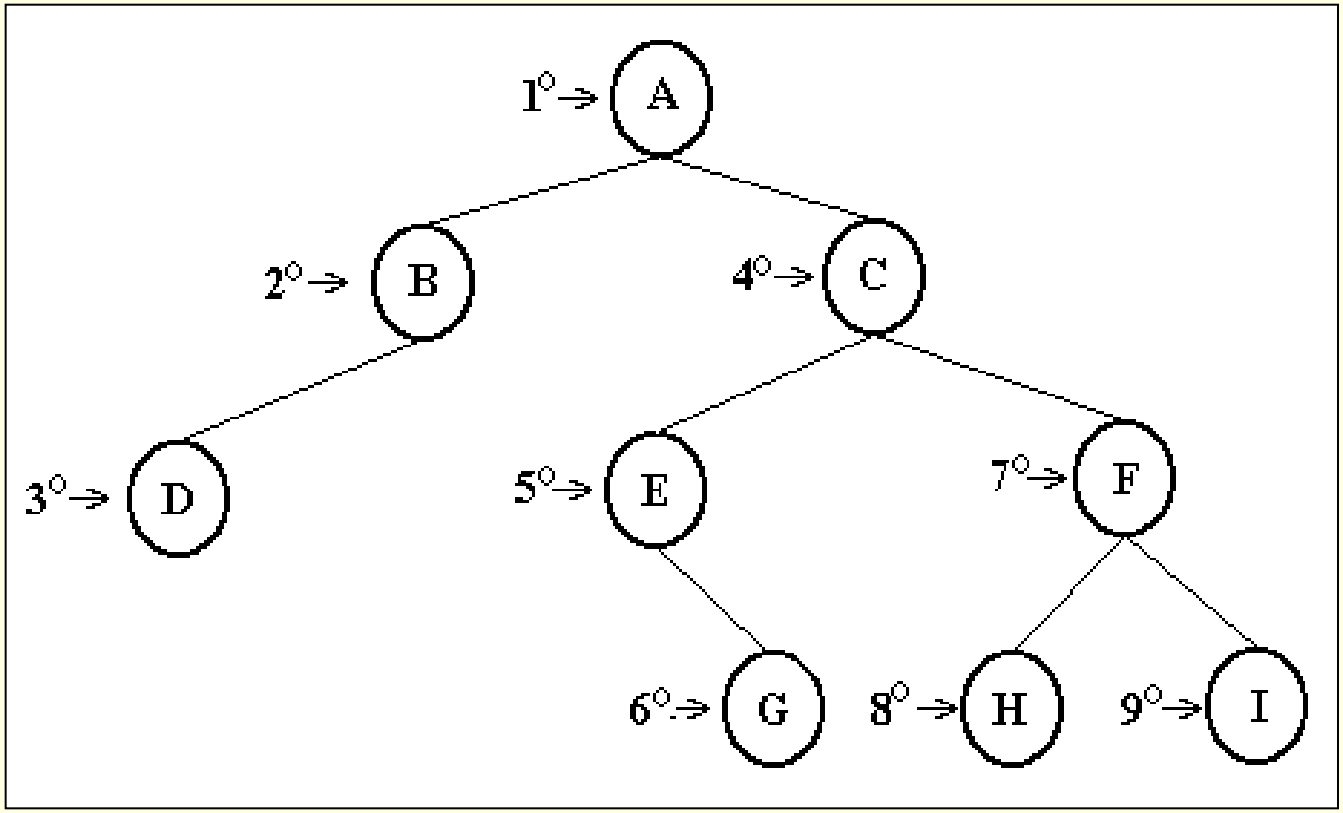
Percurso em árvores

- Em largura
  - Um nível de cada vez



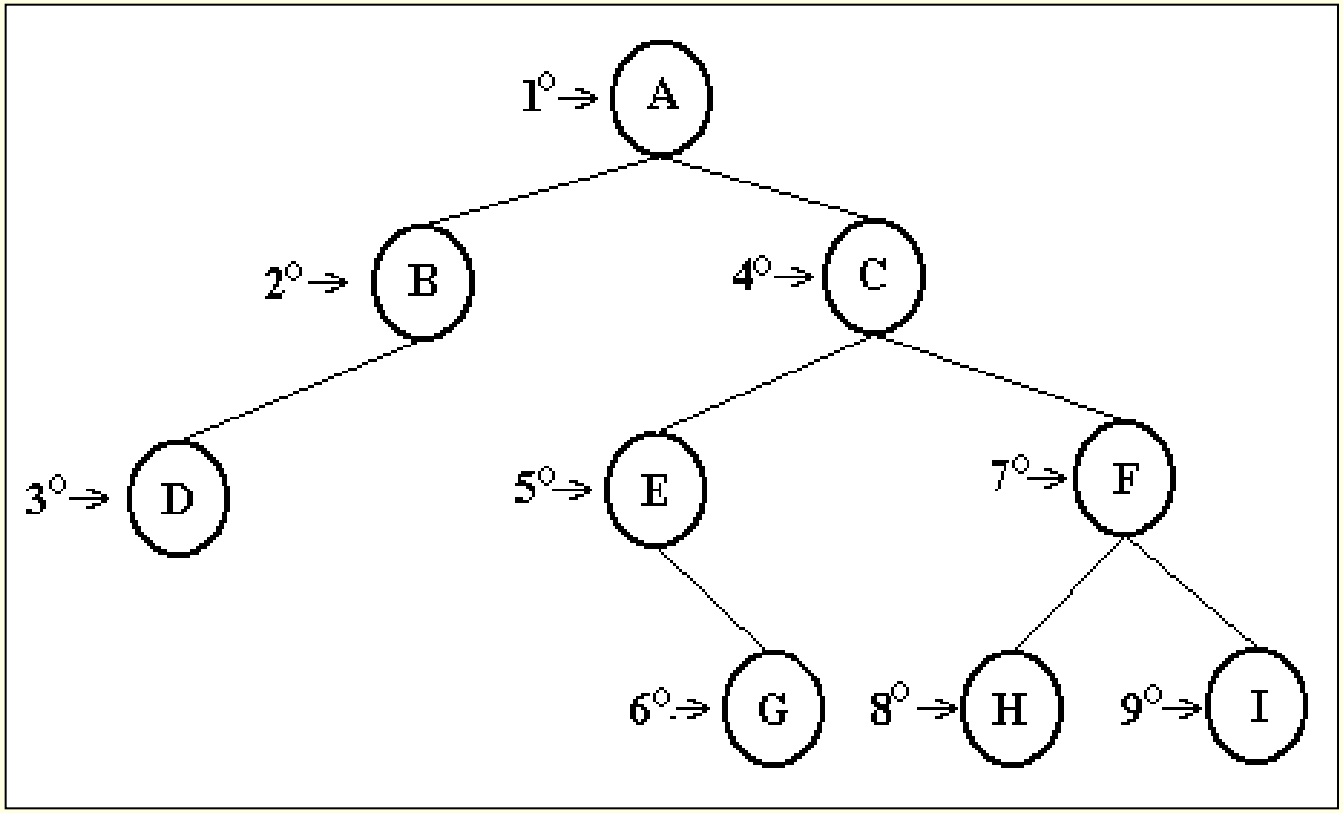
Percurso em árvores

- Em profundidade
  - Um ramo da árvore de cada vez (= ???)



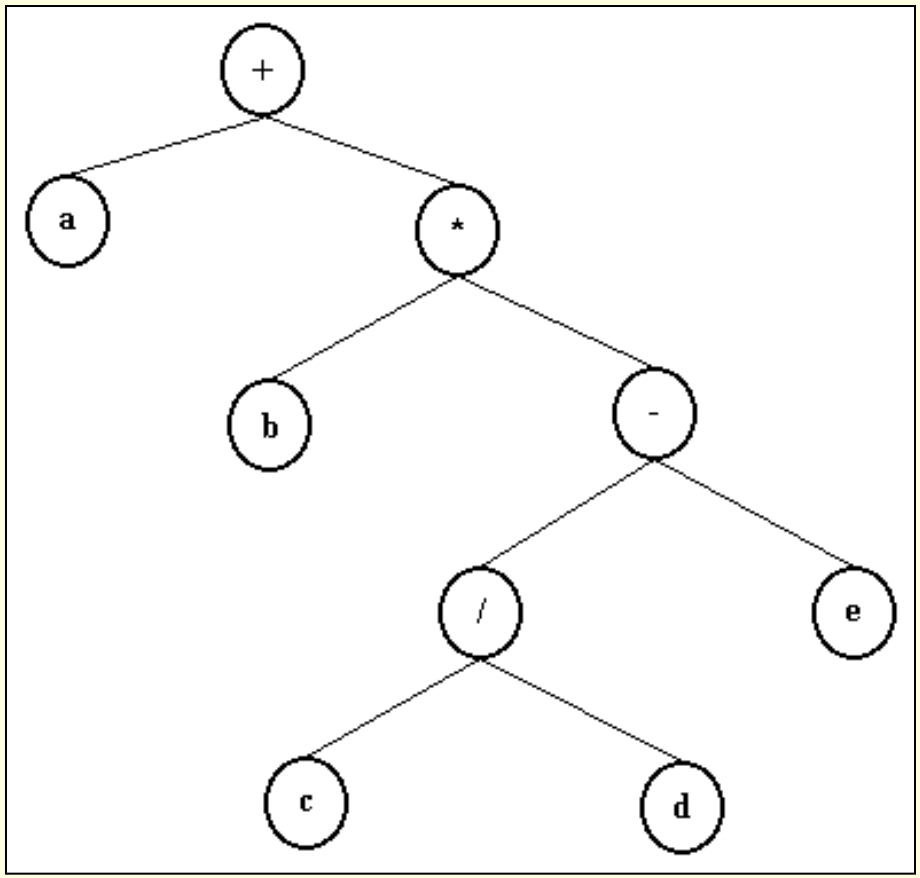
Percurso em árvores

- Em profundidade
  - Um ramo da árvore de cada vez (= pré-ordem)



Exercício: percurso em árvores

- Para a árvore abaixo, mostre quais seriam as saídas para os percursos em largura, profundidade (pré-ordem), em-ordem e pós-ordem

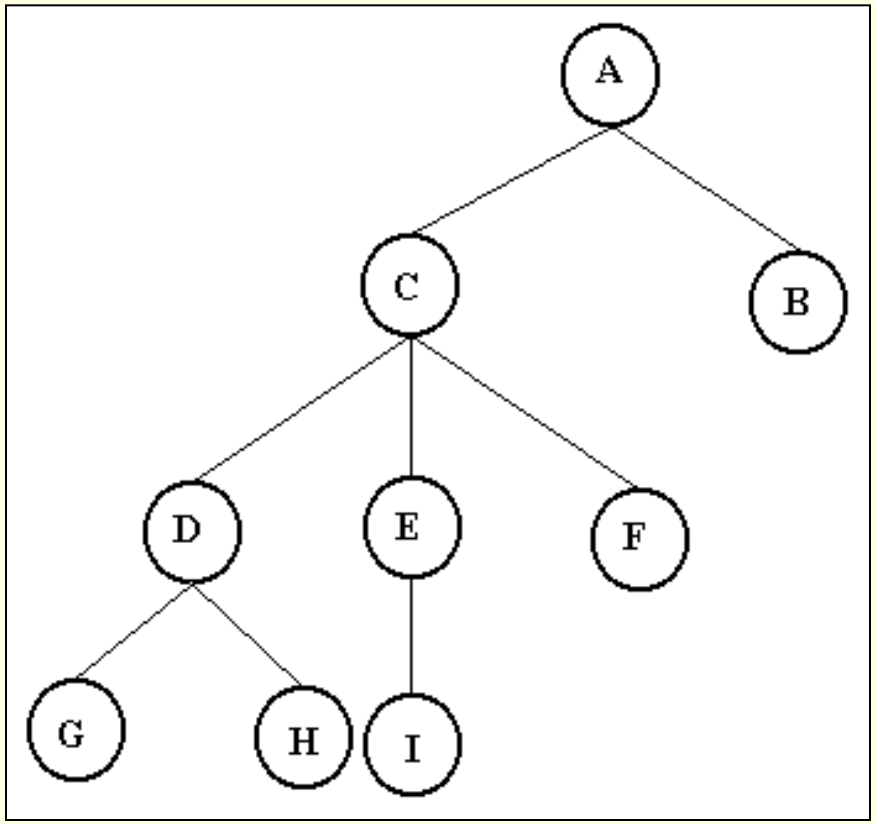


Percurso em árvores

- Em profundidade = pré-ordem
  - Usa pilha (explicita ou implicitamente)

Percurso em árvores

- Teste a estratégia com a pilha explícita



Percurso em árvores

- Em largura
  - Como implementar?

Percurso em árvores

- Em largura
  - Utiliza-se FILA

    Se o nó raiz for diferente de NULL, ele entra na fila

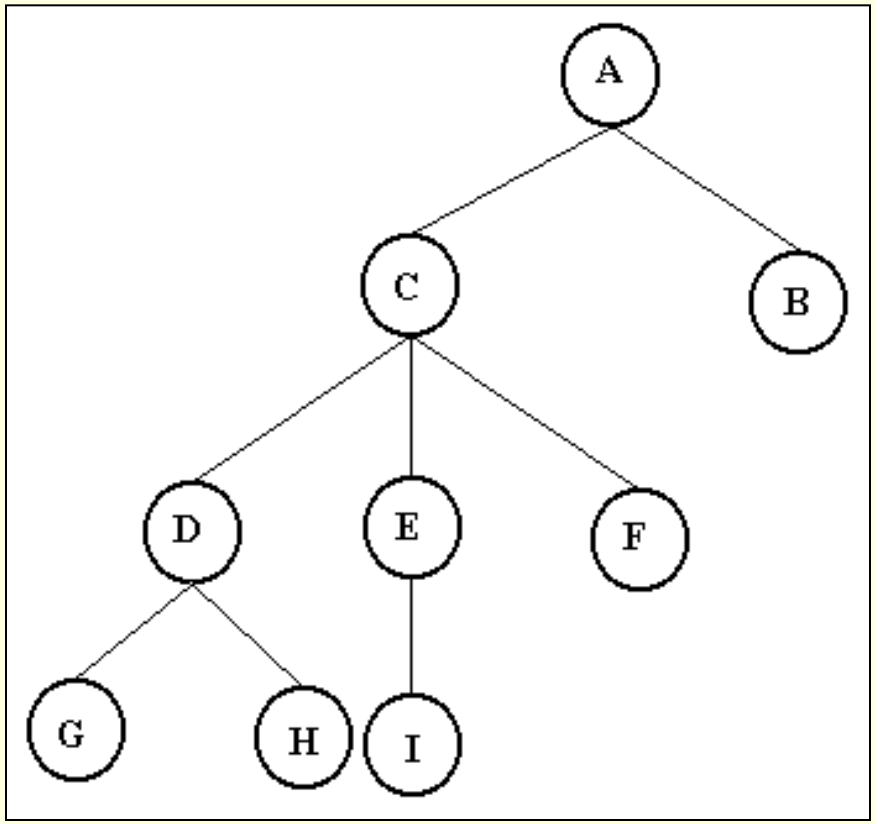
    Enquanto fila não vazia

    Retira-se/visita-se o primeiro da fila

    Se houver filhos desse nó, eles entram na fila

Percurso em árvores

- Teste a estratégia



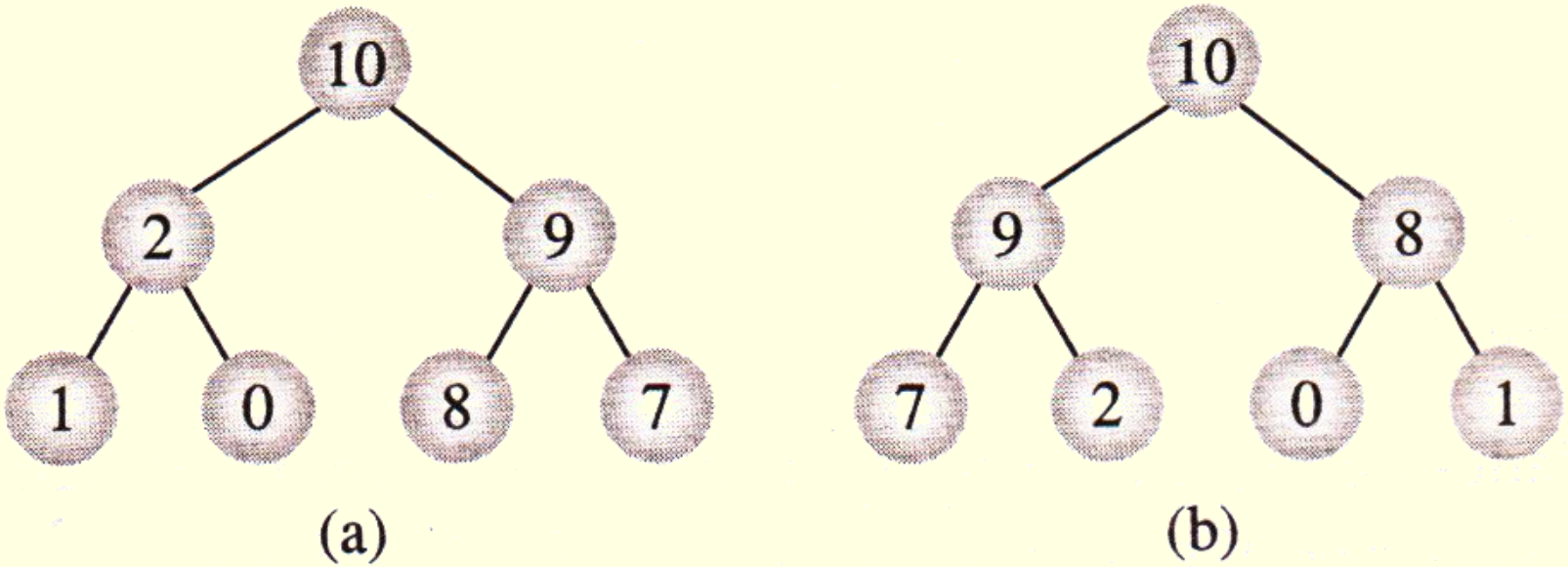
Fila de prioridade com heap

Fila de prioridade com heap

- Um heap é uma árvore binária, e observa conceitos de ordem e de forma
  - Ordem: o item de qualquer nó deve satisfazer uma relação de ordem com os itens dos nós filhos
    - Heap máximo (ou descendente): pai >= filhos, sendo que a raiz é o maior elemento
    - Heap mínimo (ou heap ascendente): pai <= filhos, sendo que a raiz é o menor elemento

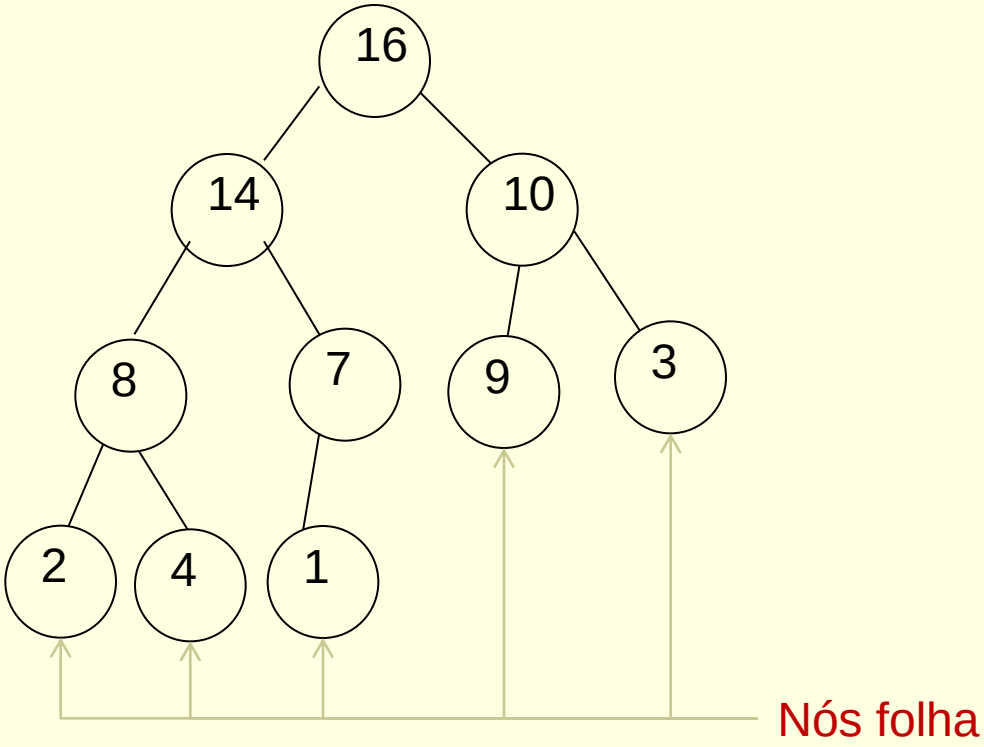
Fila de prioridade com heap

- Heaps máximos

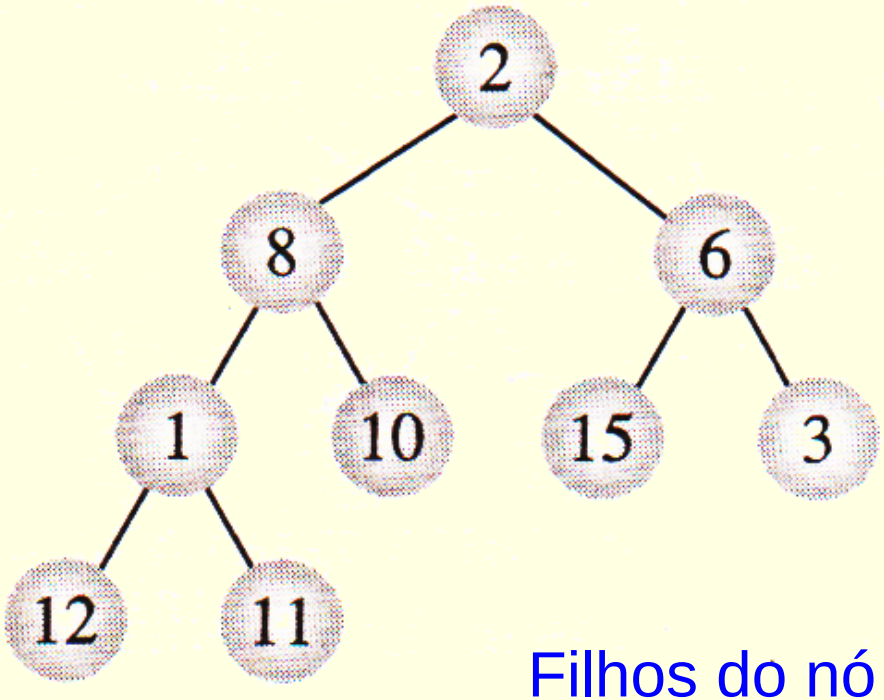


Fila de prioridade com heap

- Um heap é uma árvore binária, e observa conceitos de ordem e de forma
  - Forma: a árvore binária tem seus nós folha, no máximo, em dois níveis (ou seja, somente o último nível pode estar incompleto), sendo que as folhas devem estar o mais à esquerda possível


16
14
10
8
7
9
3
2
4
1
Nós folha

- Vetor (2 8 6 1 10 15 3 12 11) visto como um heap



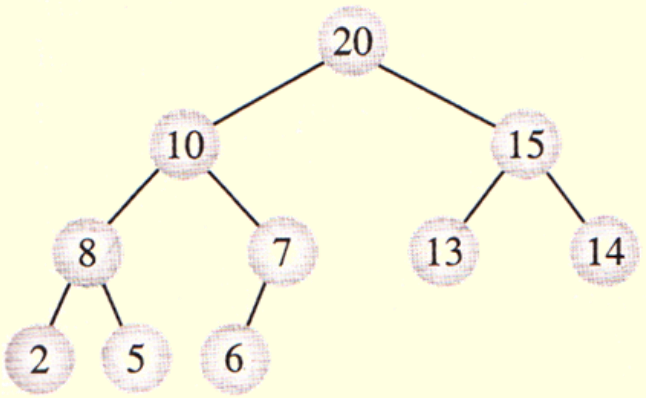
Filhos do nó i:

- 2i + 1 = filho esquerdo
- 2i + 2 = filho direito

Fila de prioridade com heap

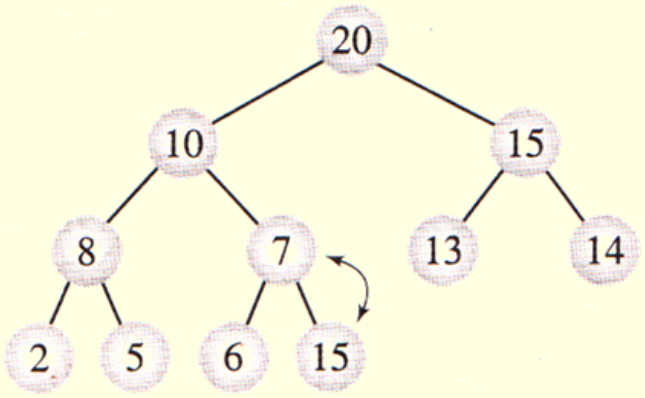
- Como utilizar um heap em uma fila de prioridade?

Inserindo o número 15



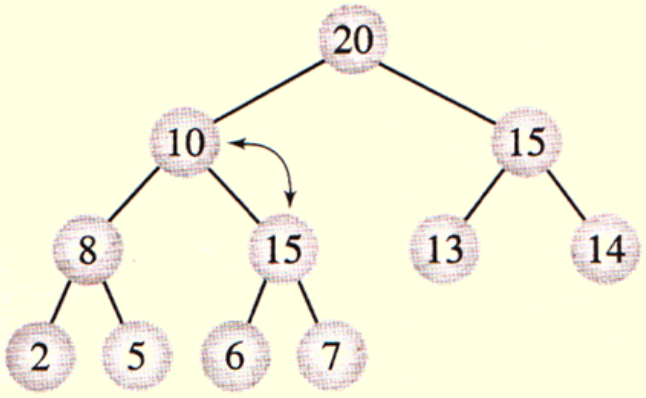
(a)

(20 10 15 8 7 13 14 2 5 6)



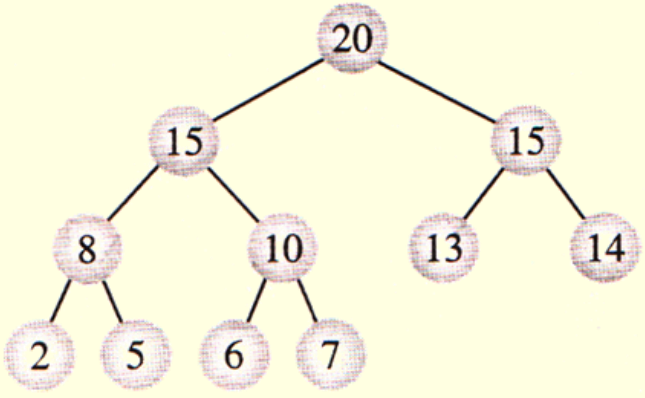
(b)

(20 10 15 8 7 13 14 2 5 6 15)



(c)

(20 10 15 8 15 13 14 2 5 6 7)



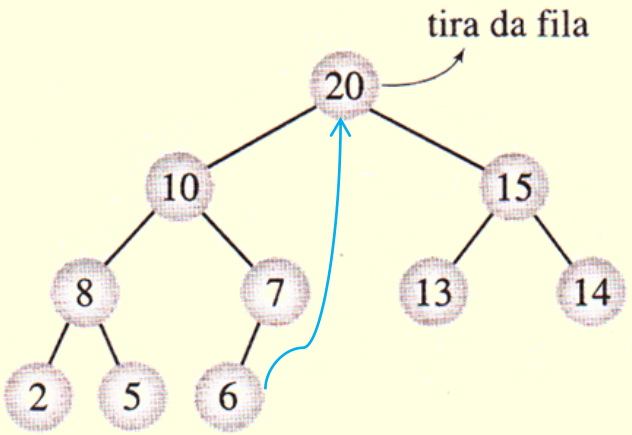
(d)

(20 15 15 8 10 13 14 2 5 6 7)

Fila de prioridade com heap

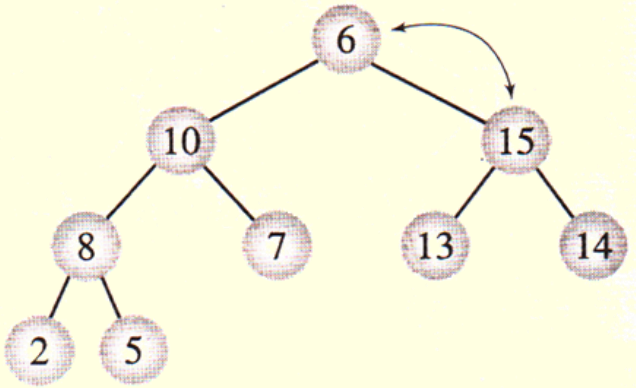
- Inserção em heap:
  - O($\log_2 n$) – *método de subida da chave*
- Inserção em lista linear ordenada:
  - O(n)
- Inserção em lista linear não ordenada:
  - O(1), mas a remoção exige busca O(n)

Removendo o elemento de maior prioridade (20)



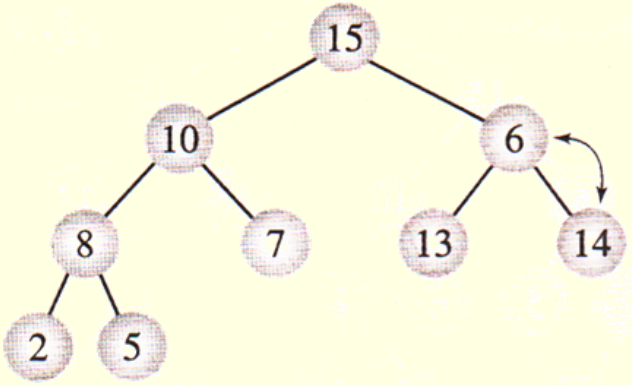
(a)

(20 10 15 8 7 13 14 2 5 6)



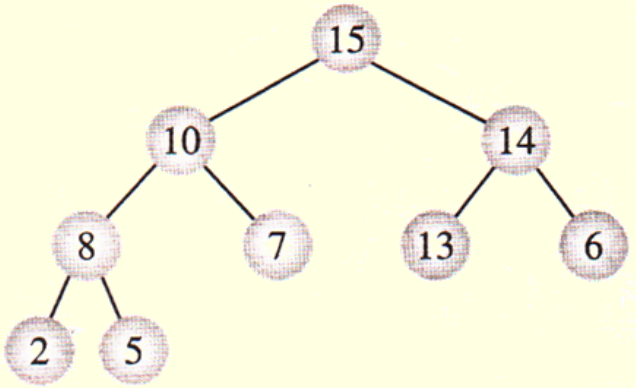
(b)

(6 10 15 8 7 13 14 2 5)



(c)

(15 10 6 8 7 13 14 2 5)



(d)

(15 10 14 8 7 13 6 2 5)

Fila de prioridade com heap

- Remoção em heap:
  - O($\log_2 n$) – *método de descida da chave*
- Remoção em lista linear ordenada:
  - O(*1*), mas inserção é O(*n*)
- Remoção em lista linear não ordenada:
  - O(*n*)

Fila de prioridade com heap

- Exercício
  - Implementar o TAD Fila de Prioridade utilizando heap.

Créditos

- *Material gentilmente cedido pelo Prof. Thiago A. S. Pardo*
- *Algumas figuras sobre heap foram utilizadas do livro A. Drozdek. Estrutura de Dados e Algoritmos em C++. Cengage Learning. 2002*