SCC0601 – Introdução à Ciência
da Computação II

Árvores binárias de busca

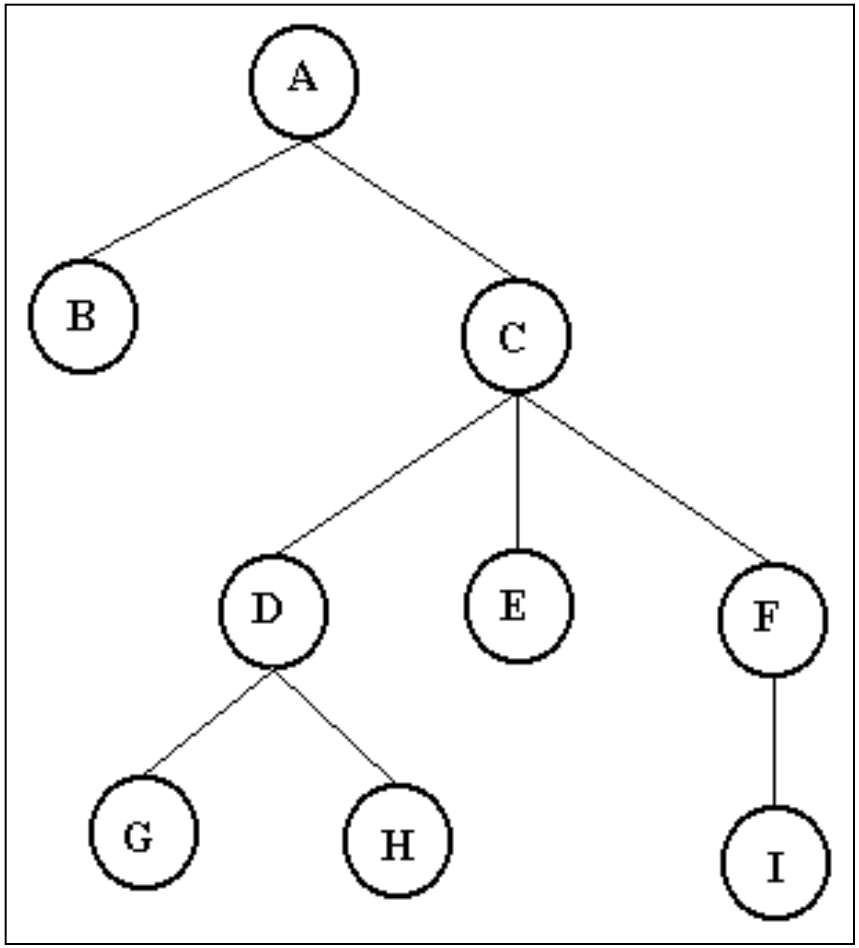
Prof. Lucas Antiqueira

Listas e árvores

- Listas lineares
  - Um nó após o outro, adjacentes
  - antecessor -> nó -> sucessor
- Diversas aplicações necessitam de estruturas mais complexas do que as listas lineares
  - Listas não lineares: árvores, grafos, etc.

Árvores

- Exemplo



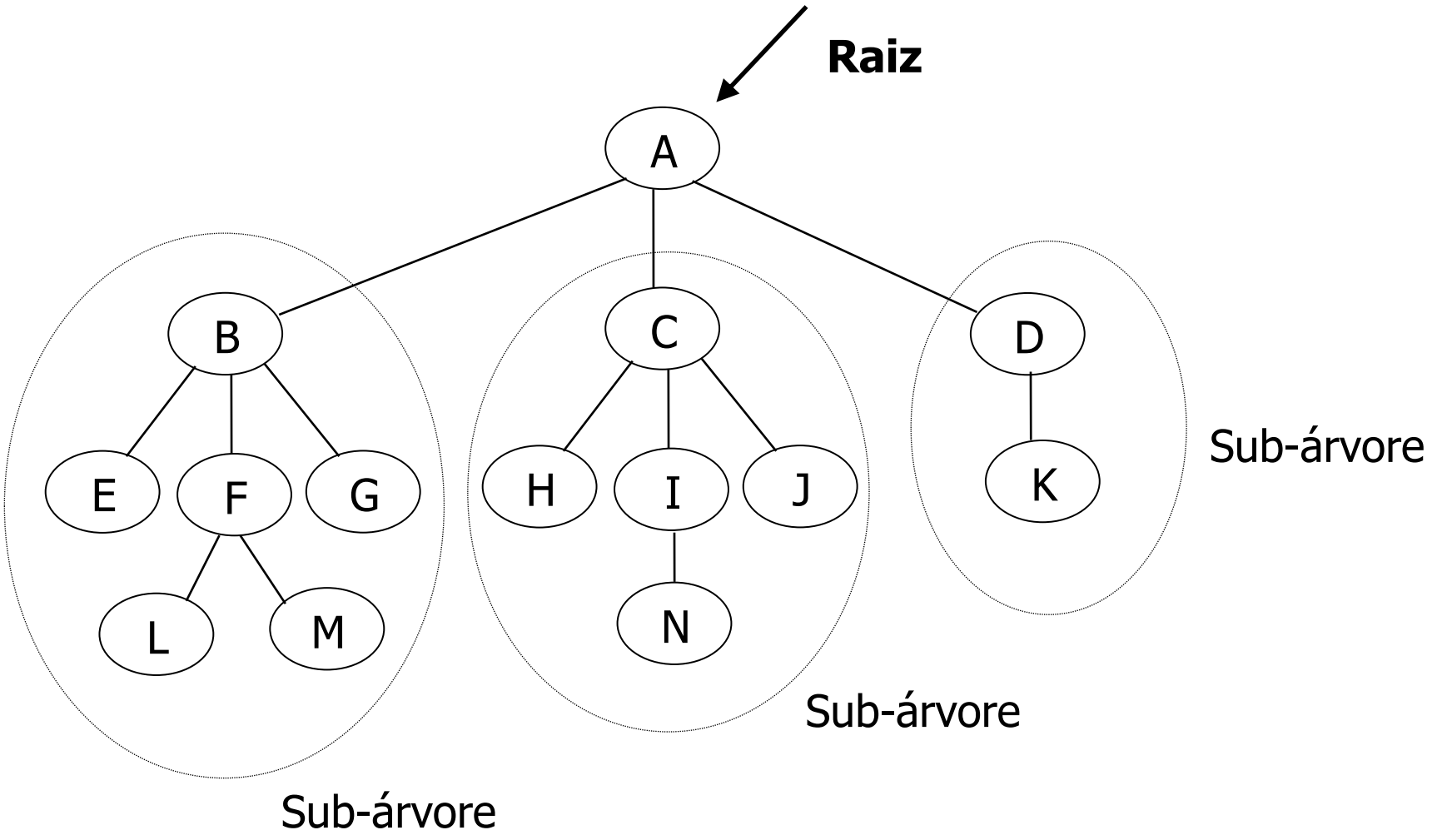
Árvores

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

Árvores

- **Definição**
  - Uma árvore T, ou simplesmente uma árvore, é um conjunto finito de elementos denominados nós ou vértices tais que
    - A árvore é 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 (>=1) conjuntos não vazios que são as sub-árvores de R, sendo que cada sub-árvore é, por sua vez, uma árvore

Árvores



Árvores

- Nós filhos, pais, tios, irmãos e avô
  - Seja V o nó raiz de uma sub-árvore de T
    - Os nós raízes w1, w2, ... wj das sub-árvores 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 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

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 um de seus descendentes
  - As folhas têm altura 1

Árvores

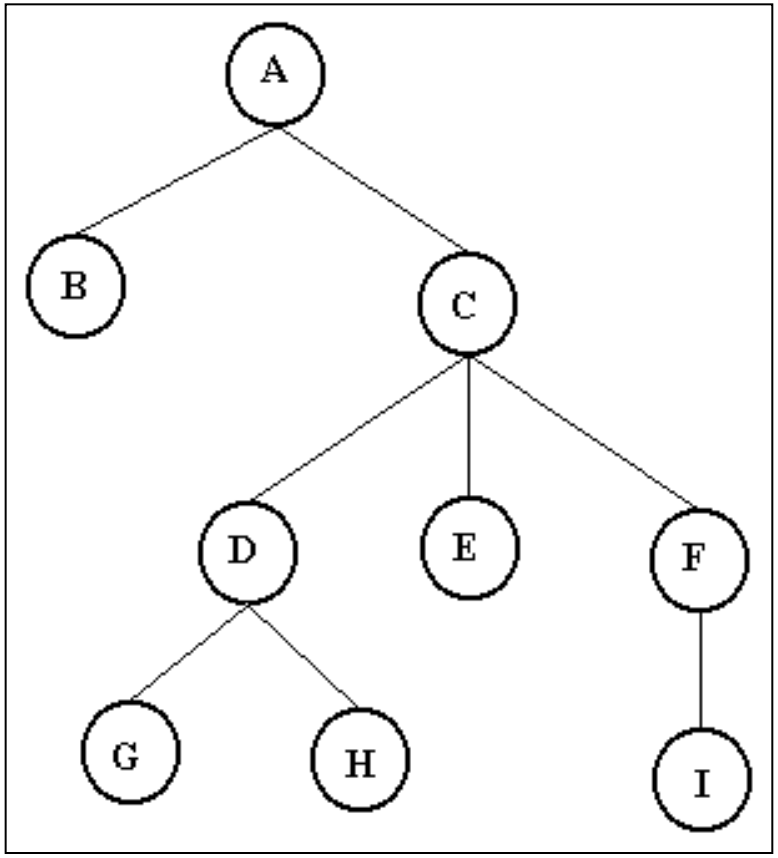
- Altura de uma árvore
  - A altura de uma árvore T é igual ao máximo nível de seus nós
  - Em geral, representa-se a altura de T por h(T) e a altura da subárvore de raiz V por h(V)

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

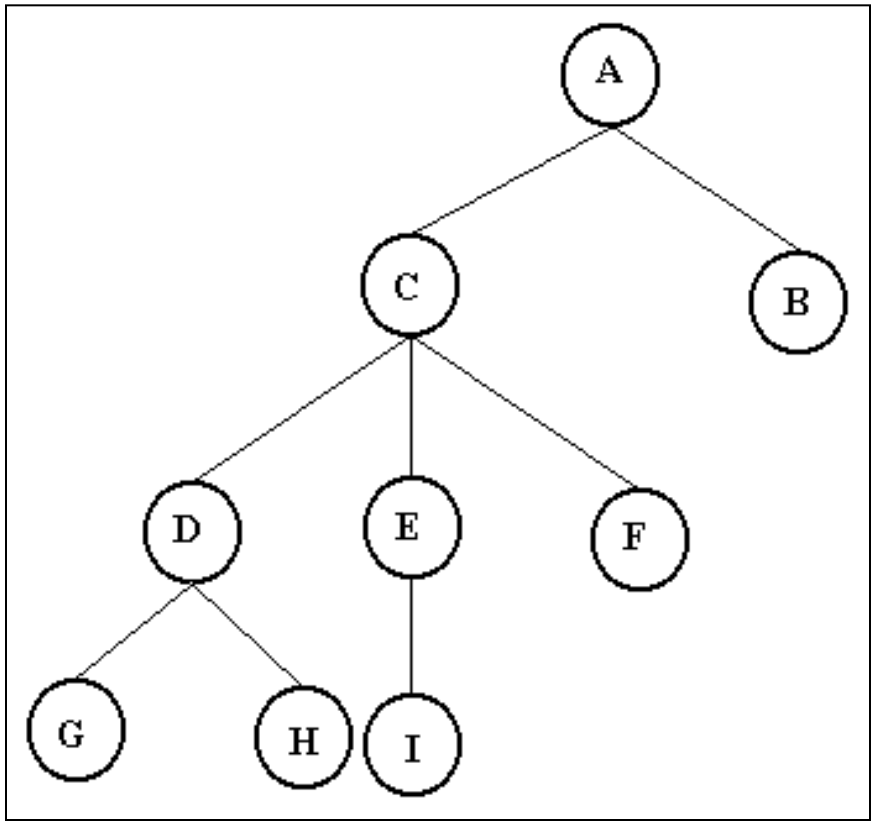
Árvores

- Árvore ordenada



Árvores

- Árvore não ordenada

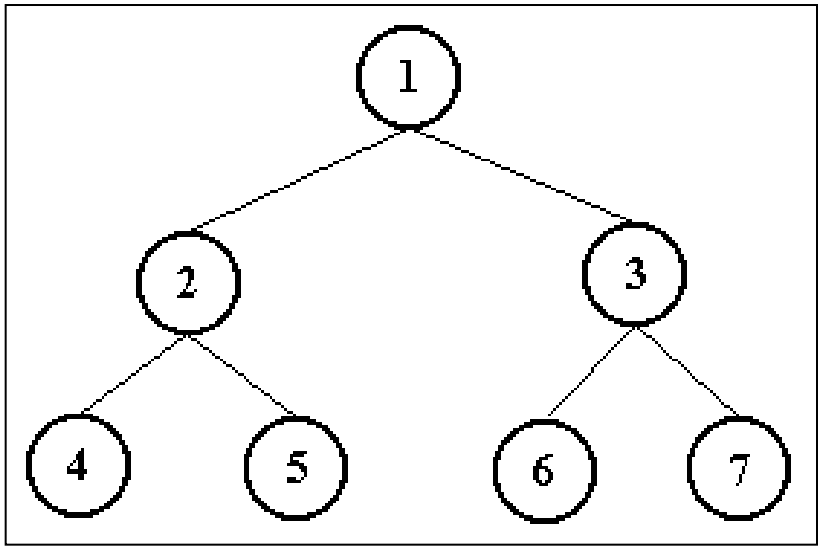


Árvores

- 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

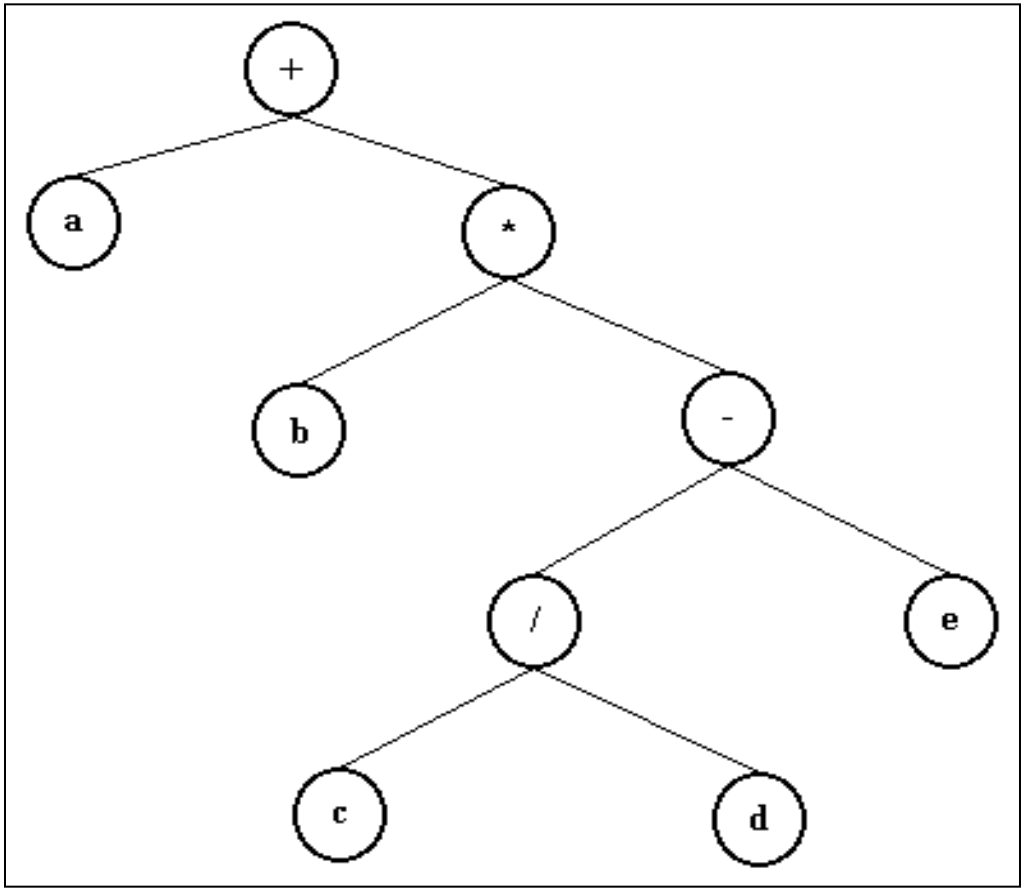
Árvores

- Exemplo de árvore cheia de grau 2



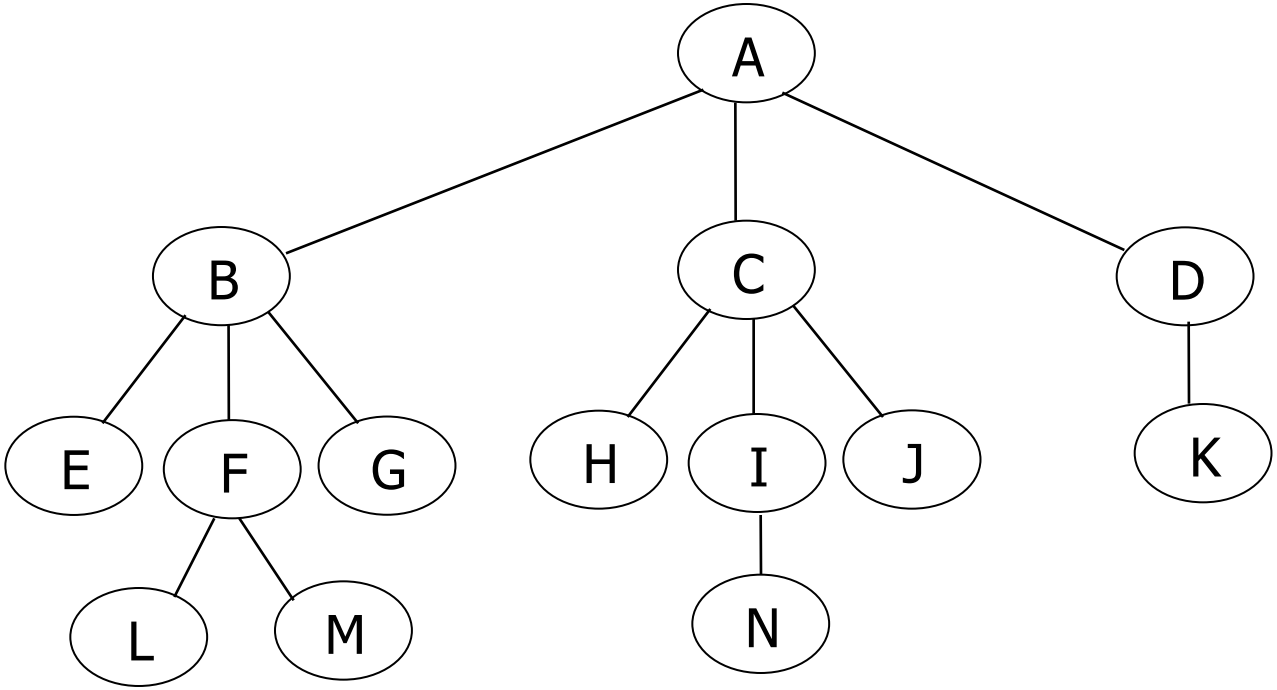
Árvores: exemplo

- Representação da expressão aritmética (a + (b * ((c / d) - e)))



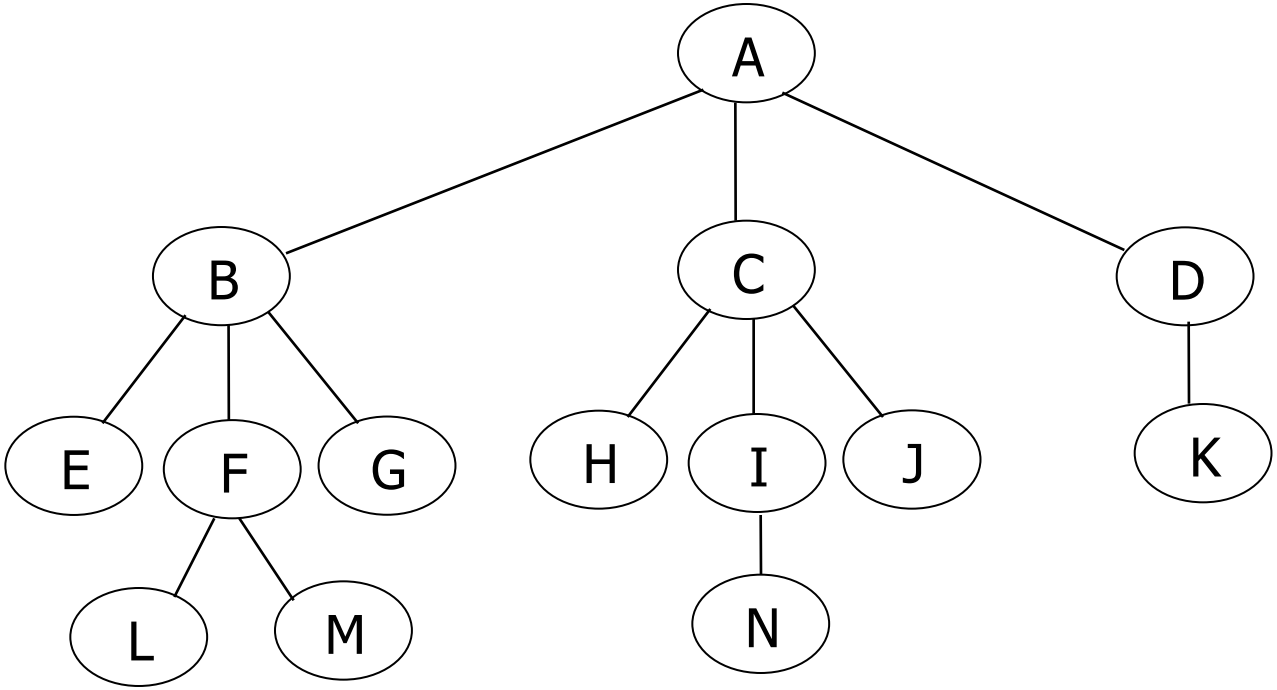
Árvores

- Considere a árvore abaixo
  - Quantas subárvores A tem?



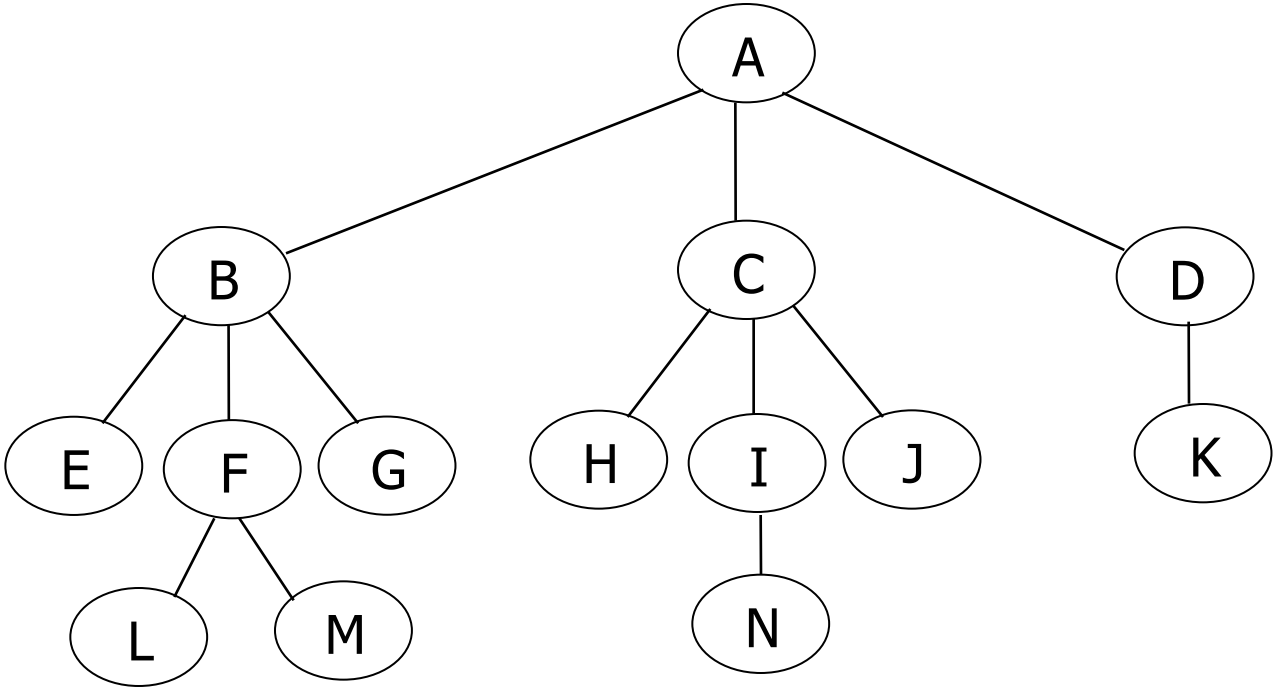
Árvores

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



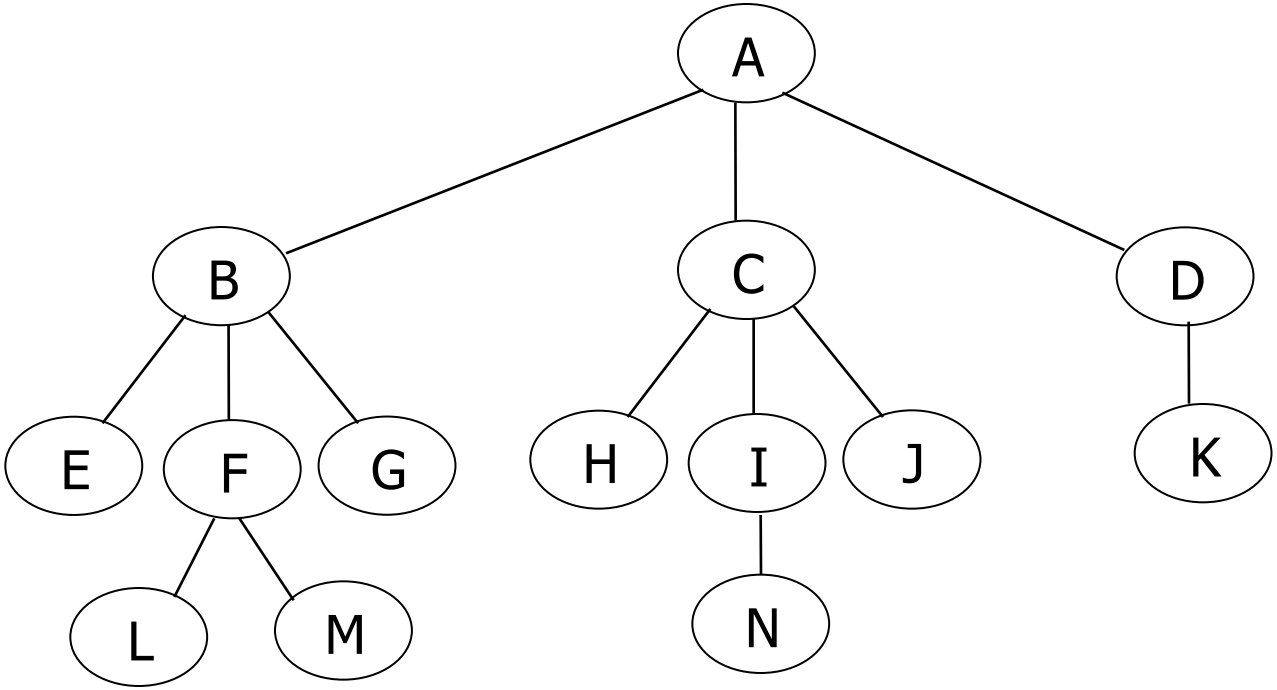
Árvores

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



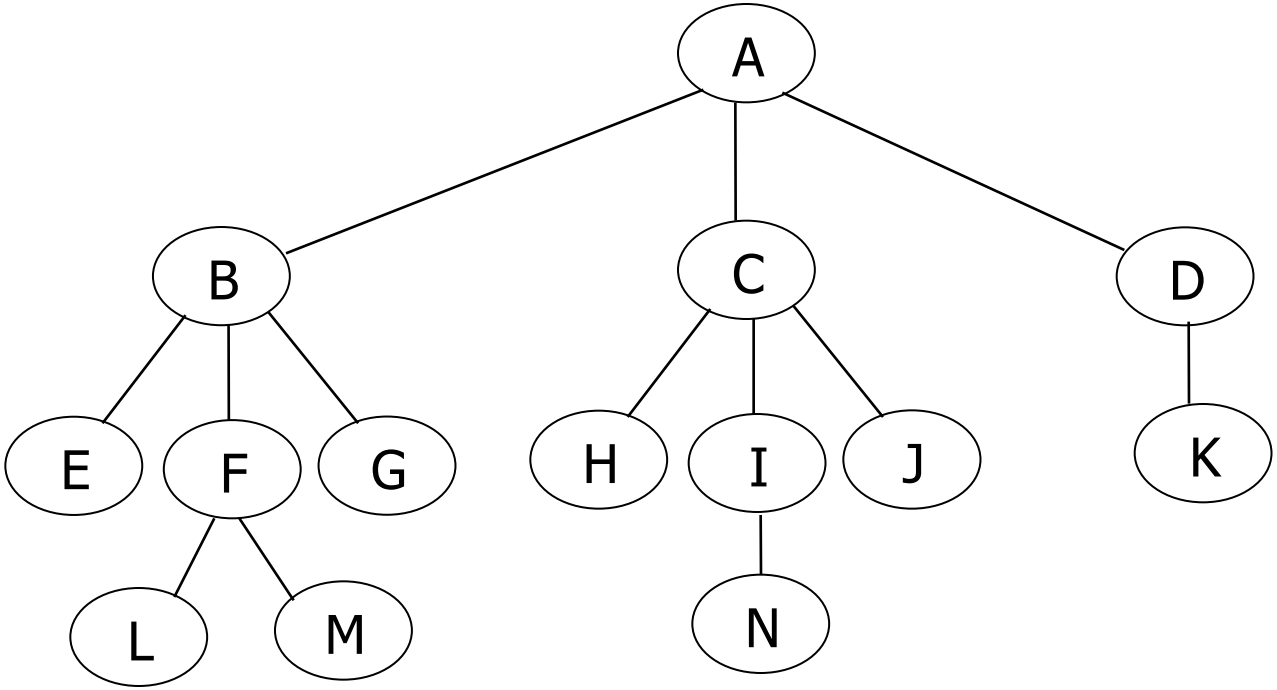
Árvores

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



Árvores

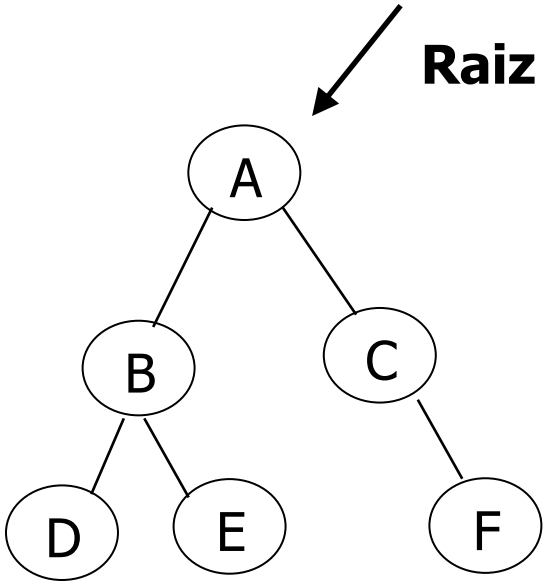
- Considere a árvore abaixo
  - Qual a altura dessa árvore?



Árvores binárias

Árvores binárias

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



Terminologia:

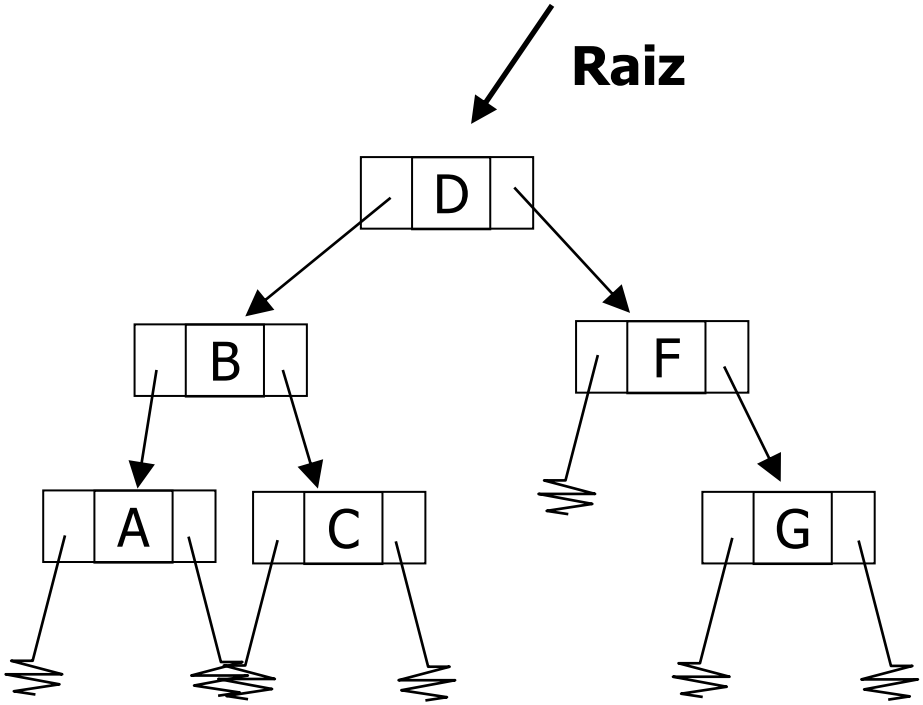
- filho esquerdo
- filho direito
- informação

Árvores binárias

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

Árvores binárias

- Representação dinâmica e encadeada de uma árvore binária



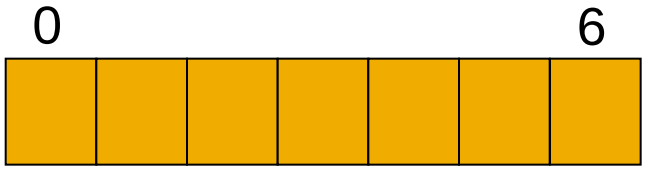
AB vazia

Árvores binárias

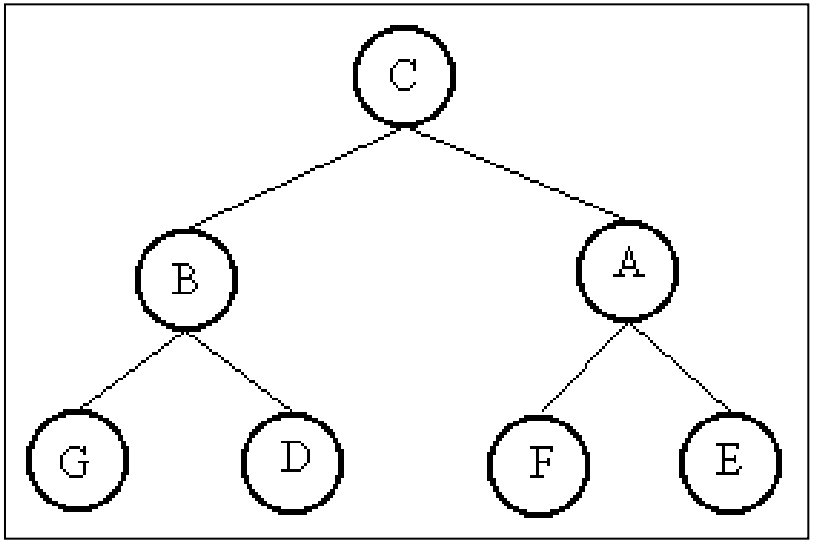
```
typedef int elem;

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

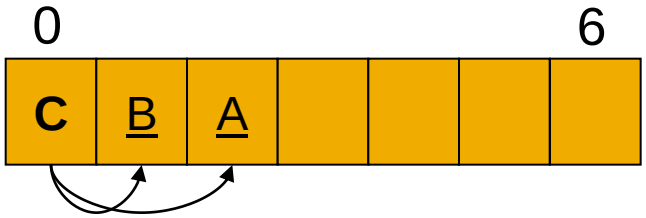


  - Como colocar a árvore abaixo nesse vetor?

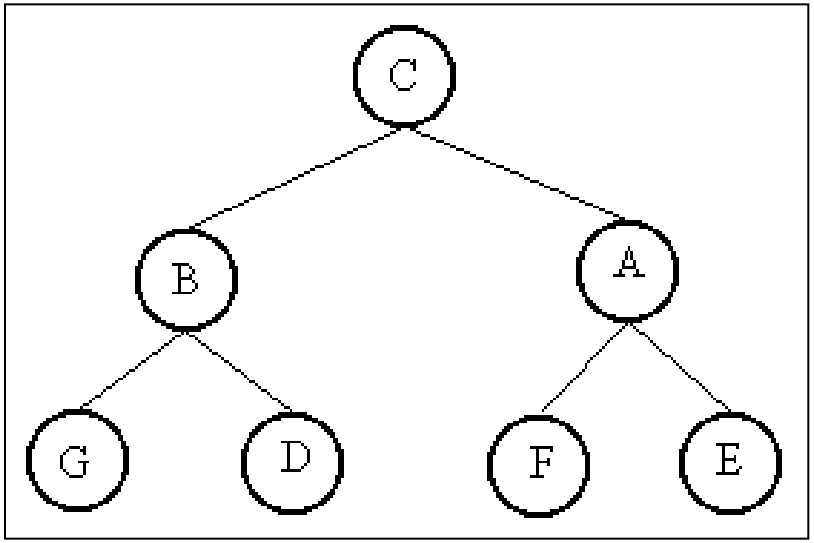


Árvores binárias

- Representação estática e seqüencial de árvores binárias
  - Vetor

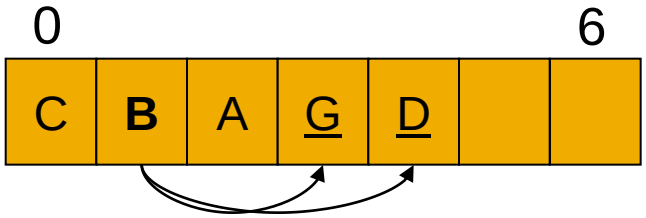


  - Como colocar a árvore abaixo nesse vetor?

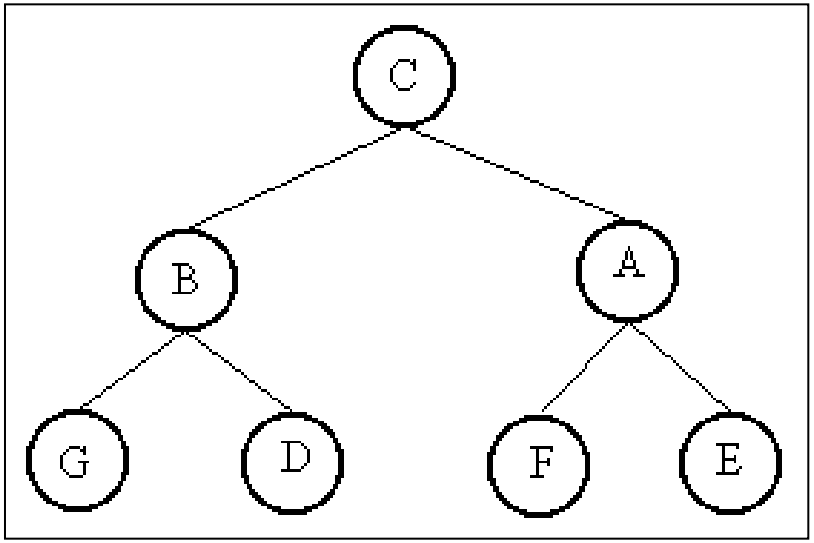


Árvores binárias

- Representação estática e seqüencial de árvores binárias
  - Vetor

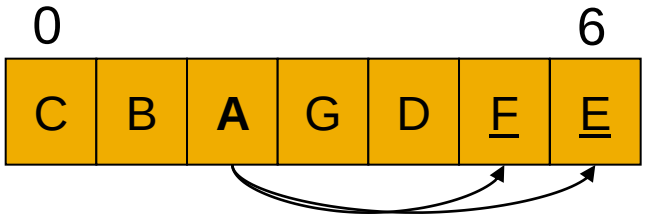


  - Como colocar a árvore abaixo nesse vetor?

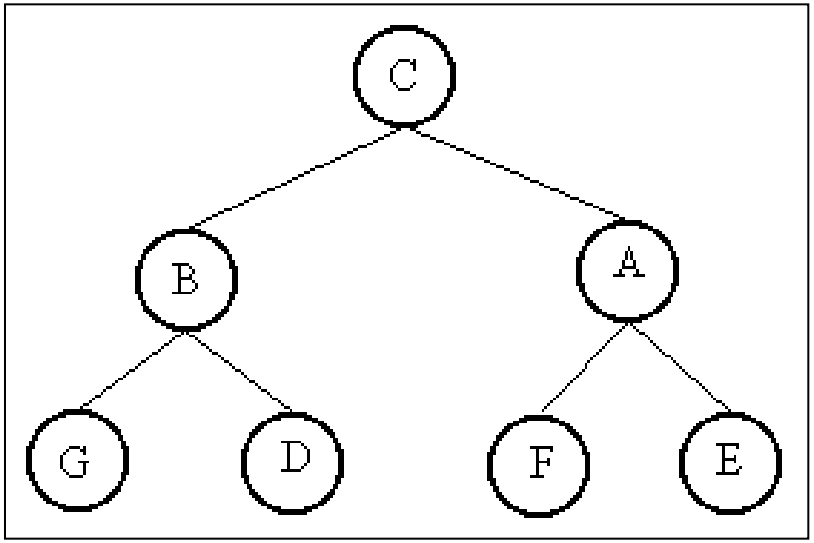


Árvores binárias

- Representação estática e seqüencial de árvores binárias
  - Vetor

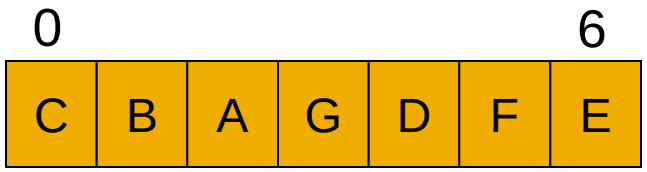


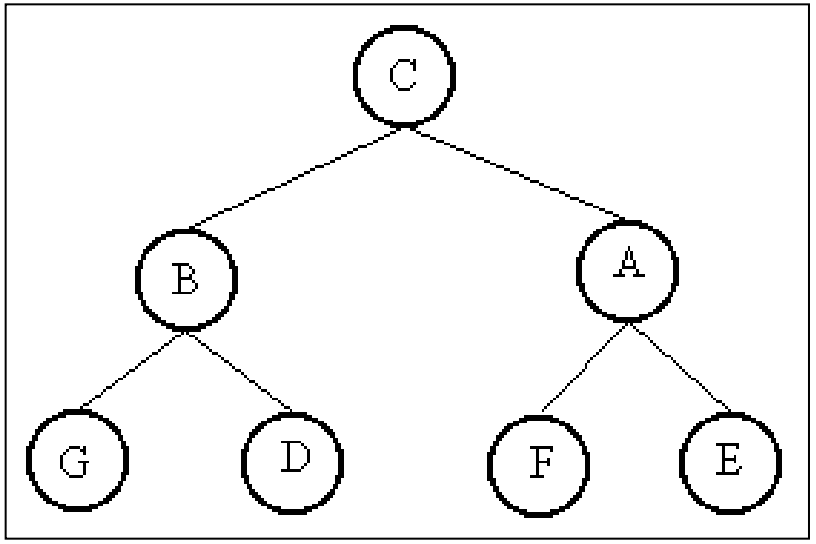
  - Como colocar a árvore abaixo nesse vetor?



Árvores binárias

- Representação estática e seqüencial de árvores binárias

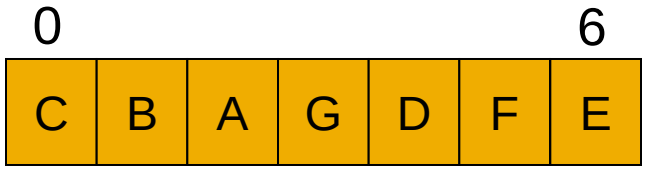


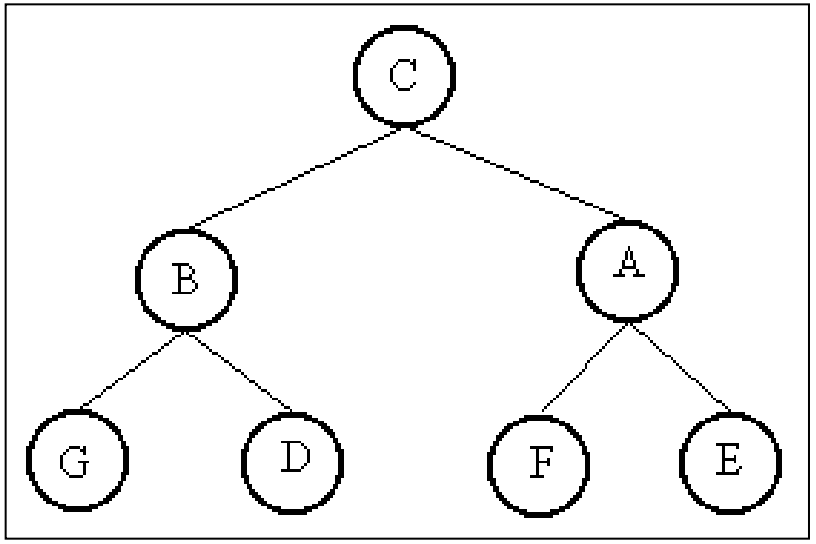


  - Como saber quem é filho de quem?

Árvores binárias

- Representação estática e seqüencial de árvores binárias

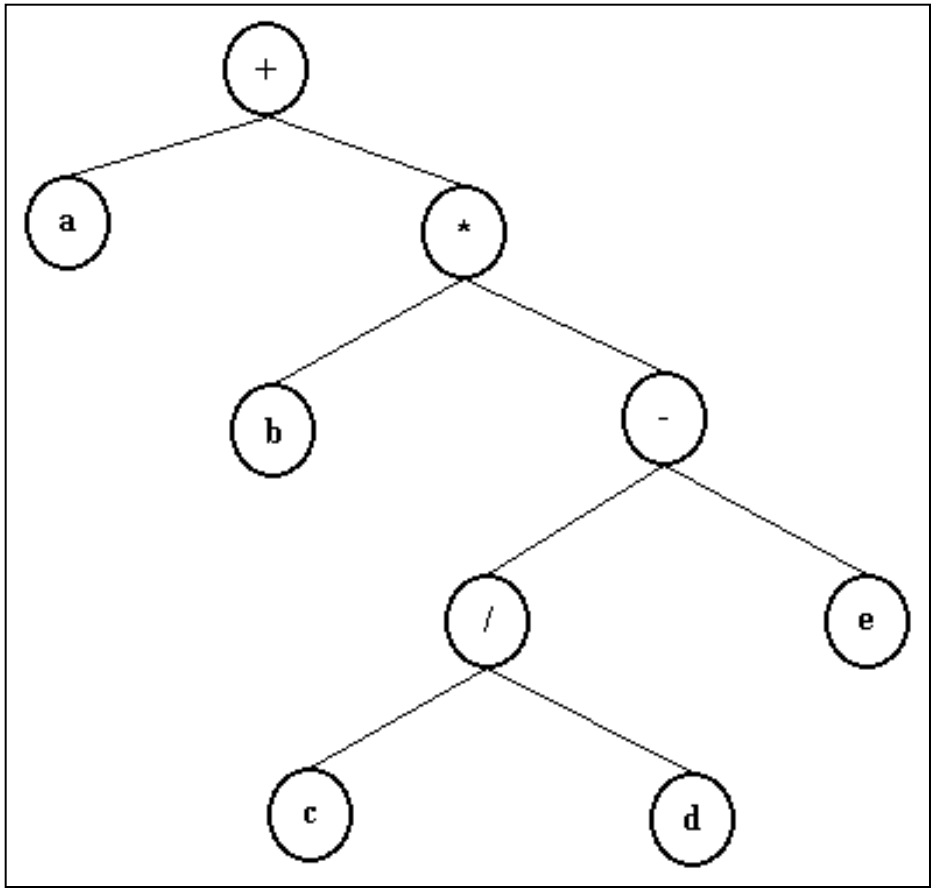




  - Como saber quem é filho de quem?
    - **Filhos de i são 2i+1 e 2i+2**

Árvores binárias

- Exercício: represente a árvore abaixo em um vetor



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?

Árvores binárias

- Exercício: Implemente uma sub-rotina para percorrer uma árvore binária (dinâmica e encadeada) e imprimir todos os seus elementos

Árvores binárias

```
void imprimir(no *p) {
   if (p != NULL) {
      printf("%d(", p->info);
      imprimir(p->esq);
      printf(",");
      imprimir(p->dir);
      printf(")");
   } else
      printf("null");
}
```

Árvores binárias

- Exercício: Implemente uma sub-rotina para determinar a altura de uma árvore binária (dinâmica e encadeada)

Árvores binárias

```
int altura(no *p) {
   int alt_esq, alt_dir;

   if (p == NULL)
      return 0;
   else {
      alt_esq = 1 + altura(p->esq);
      alt_dir = 1 + altura(p->dir);
      if (alt_esq > alt_dir)
            return alt_esq;
      else
            return alt_dir;
   }
}
```

Árvores binárias de busca (ABB)

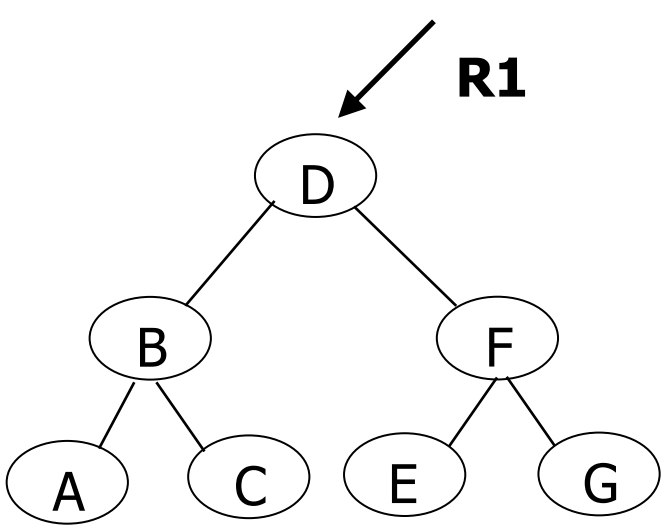
ABB

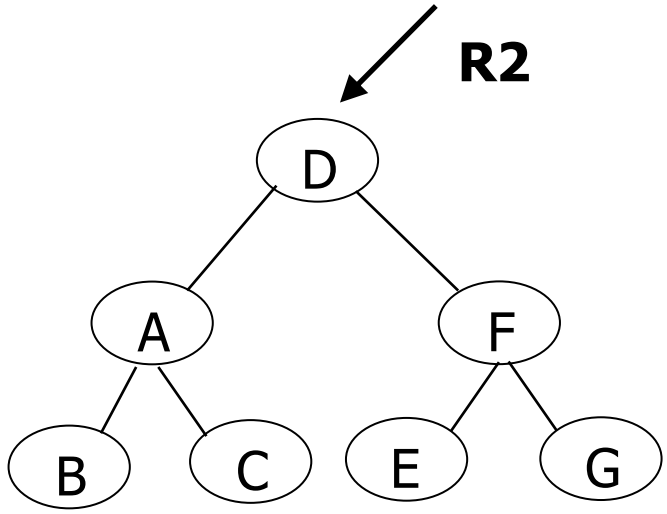
- Também chamadas "*árvores de pesquisa*" ou "*árvores ordenadas*"

- *Definição*
  - Uma árvore binária com **raiz R** é uma **ABB** se:
    - a chave (informação) de cada nó da sub-árvore esquerda de R é menor do que a chave do nó R (em ordem alfabética, por exemplo)
    - a chave de cada nó da sub-árvore direita de R é maior do que a chave do nó R
    - as subárvores esquerda e direita também são ABBs

ABB

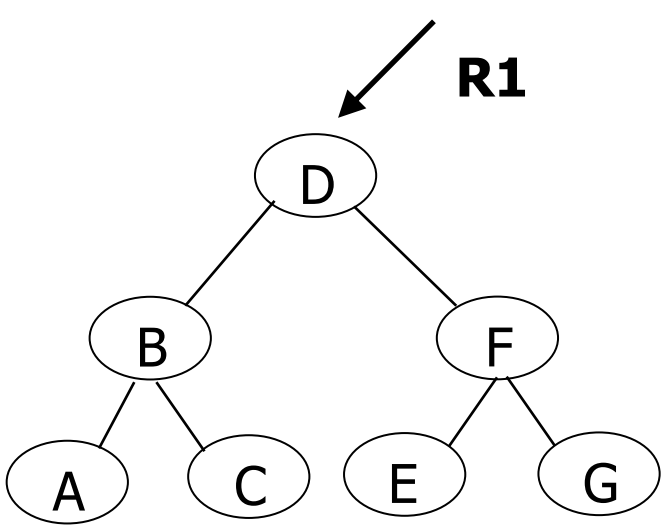
- Exemplos



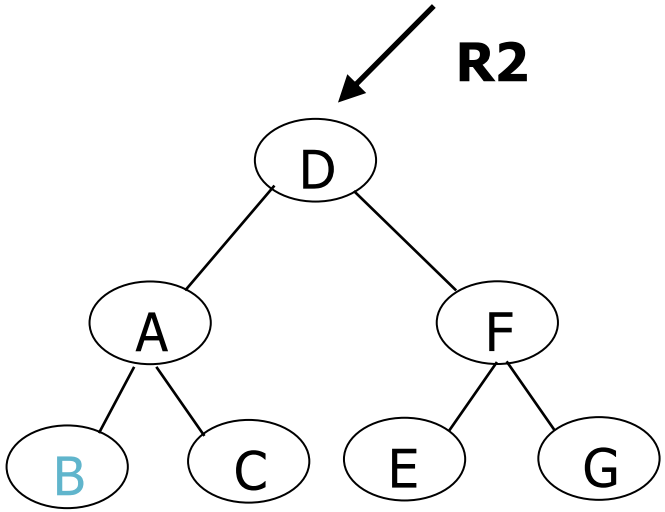


ABB

- Exemplos



é ABB



não é ABB

ABB

- Para que uma ABB é útil?

- Imagine a situação
  - Sistema de votação por telefone
    - Cada número só pode votar uma vez
    - Um sistema deve armazenar todos os números que já ligaram
    - A cada nova ligação, deve-se consultar o sistema para verificar se aquele número já votou; o voto é computado apenas se o número ainda não votou
    - A votação deve ter resultado on-line

ABB

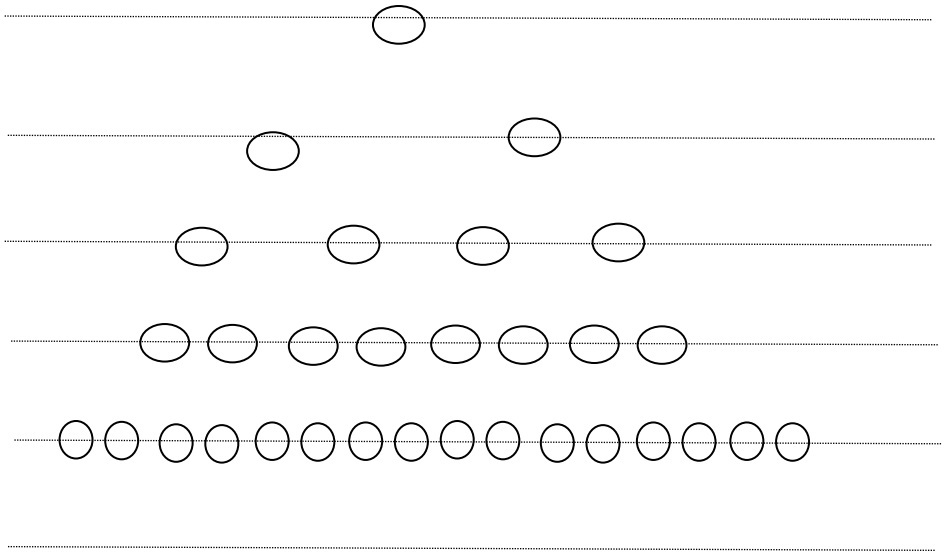
- Para que uma ABB é útil?

- Solução com ABBs
  - Cada número de telefone é armazenado em uma ABB
  - Suponha que em um determinado momento, a ABB tenha 1 milhão de telefones armazenados
  - Surge nova ligação e é preciso saber se o número está ou não na árvore (se já votou ou não)

ABB

- Para que uma ABB é útil?

- Considere uma ABB com chaves uniformemente distribuídas (árvore cheia)

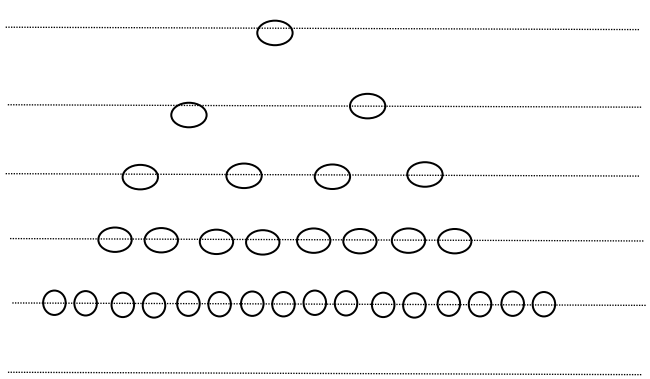


ABB

- Para que uma ABB é útil?
- *Responda*
  - Quantos elementos cabem em uma árvore de N níveis, como a anterior?
  - Como achar um elemento em uma árvore assim a partir da raiz?
  - Quantos nós se tem que visitar, no máximo, para achar o telefone na árvore, ou ter certeza de que ele não está na árvore?

ABB

- Para que uma ABB é útil?



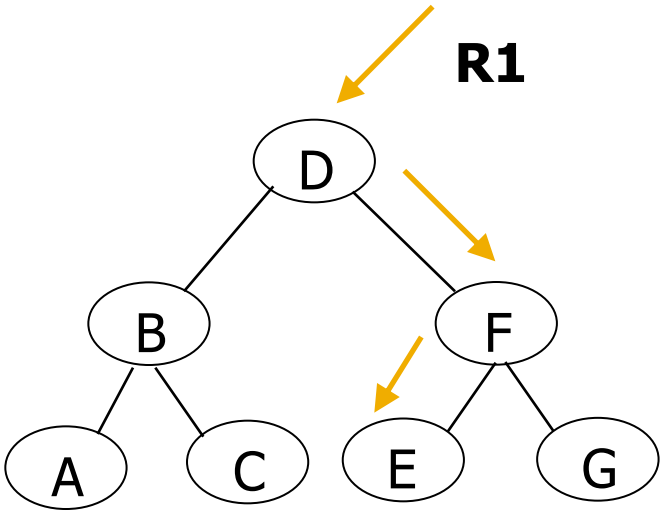
Nível	Quantos nós cabem?
1	1
2	3
3	7
4	15
10	1.023
13	8.191
16	65.535
18	262.143
20	1.048.575
30	1.073.741.823
...	...
N	**$2^N - 1$**

ABB

- Para que uma ABB é útil?
- Para se **buscar** em uma ABB
  - Em cada nó, compara-se o elemento buscado com o elemento presente
    - Se for menor, percorre-se a sub-árvore esquerda
    - Se for maior, percorre-se a sub-árvore direita
  - Desce até as folhas, no pior caso, sem passar por mais de um nó em um mesmo nível
  - Portanto, no pior caso, a busca passa por todos os níveis da árvore

ABB

- Exemplo: busca pelo elemento E nas árvores abaixo



3 consultas

ABB

- Declaração igual a das árvores binárias convencionais. Contudo, a manipulação é diferente

```
typedef int elem;

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

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

ABB

- Operações sobre a ABB
  - Devem considerar a ordenação dos elementos da árvore
    - Por exemplo, na inserção, deve-se procurar pelo local certo na árvore para se inserir um elemento

ABB

- Operações básicas
  - Busca
  - Inserção
  - Remoção

ABB

- Está na árvore?
  - Comparando o parâmetro "chave" com a informação no nó "raiz", 4 casos podem ocorrer
    - A árvore é vazia => a chave não está na árvore => fim do algoritmo
    - Elemento da raiz é a chave => achou o elemento (está no nó raiz) => fim do algoritmo
    - chave < elemento da raiz => chave pode estar na subárvore esquerda
    - chave > info(raiz) => chave pode estar na subárvore direita
  - Pergunta: quais os casos que podem ocorrer para a subárvore esquerda? E para a subárvore direita?

ABB

- Exercício: Implementação da sub-rotina de busca por um elemento na árvore

ABB

```
int busca(no *p, elem x) {
  if (p == NULL)
     return 0;
  else if (x == p->info)
     return 1;
  else if (x < p->info)
     return busca(p->esq, x);
  else
     return busca(p->dir, x);
}
```

ABB

- Exercício: Construa graficamente uma ABB, inicialmente vazia, em que os elementos
  11, 5, 3, 16, 7, 6, 1, 20, 13, 12, 17, 15
  são inseridos um por um na sequencia fornecida.

ABB

- Inserção
  - Estratégia geral
    - Inserir elementos como nós folha (sem filhos)
    - Procurar o lugar certo e então inserir
  - Comparando o parâmetro "chave" com a informação no nó "raiz", 4 casos podem ocorrer
    - A árvore é vazia => insere o elemento, que passará a ser a raiz; fim do algoritmo
    - Elemento da raiz = chave => o elemento já está na árvore; fim do algoritmo
    - Chave < elemento da raiz => insere na subárvore esquerda
    - Chave > elemento da raiz => insere na subárvore direita

ABB

- Exercício: Implementação da sub-rotina de inserção de um elemento na árvore

ABB

```
1. int inserir(no **p, elem x) {
2.     if (*p == NULL) {
3.             *p = (no *) malloc(sizeof(no));
4.             (*p)->info = x;
5.             (*p)->esq = NULL;
6.             (*p)->dir = NULL;
7.             return 1;
8.     } else if (x < (*p)->info)
9.             return inserir(&(*p)->esq, x);
10.    else if (x > (*p)->info)
11.            return inserir(&(*p)->dir, x);
12.    else
13.            return 0;     /*elemento já está na árvore*/
14.}
```

ABB

Exemplo de chamada:

```
int main () {
   Arvore A;
   /* Deve inicializar a árvore aqui */
   inserir(&A.raiz, 11);
   return 0;
}
```

ABB

Como inicializar uma árvore?

```
void criar(Arvore *A) {
   A->raiz = NULL;
}
```

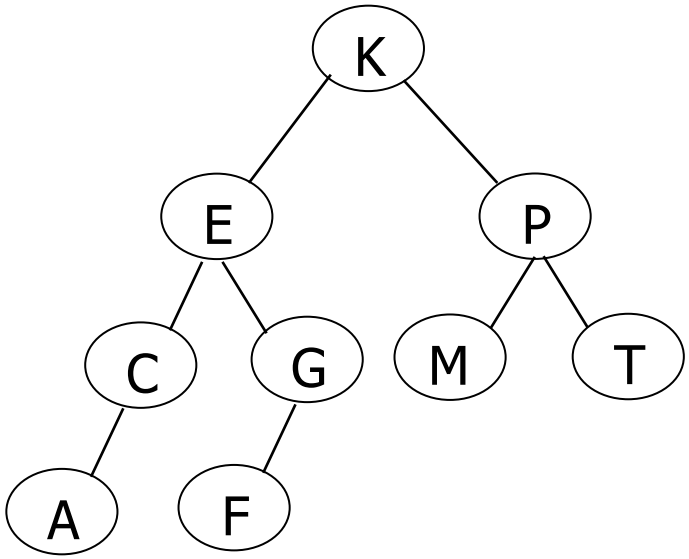
ABB

Exemplo de chamada:

```
int main () {
   Arvore A;
   criar(&A);
   inserir(&A.raiz, 11);
   return 0;
}
```

ABB

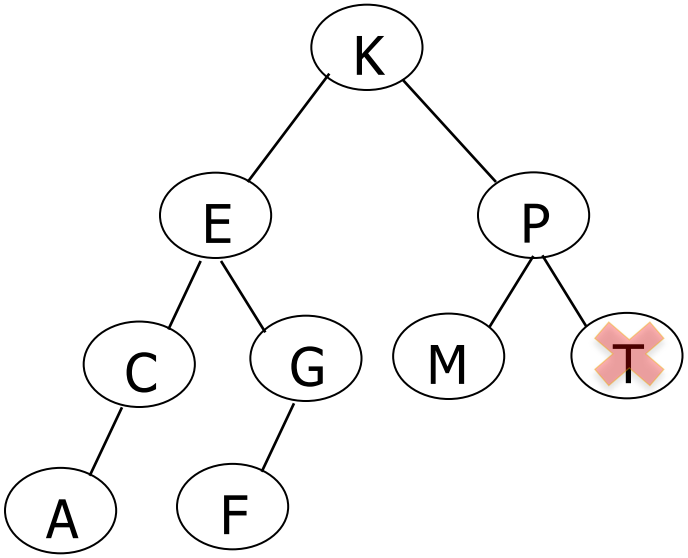
- Remoção
  - Para a árvore abaixo, remova os elementos T, C e K, nesta ordem



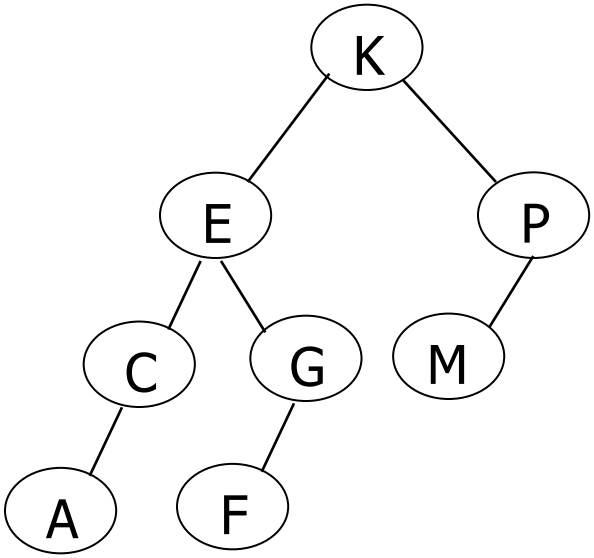
ABB

- Caso 1 (remover T): o nó p a ser removido não tem filhos
  - Remove-se o nó
  - p aponta para NULL

ABB



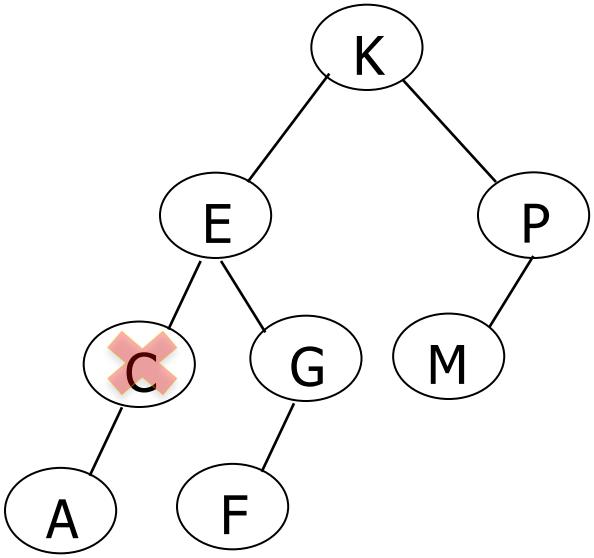
ABB



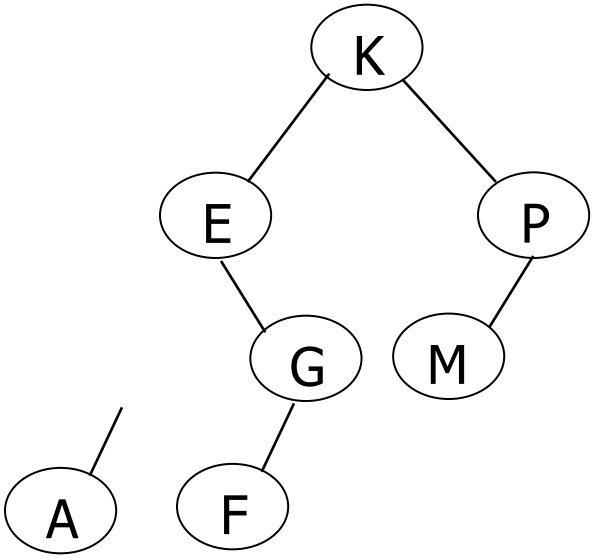
ABB

- Caso 2 (remover C): o nó a ser removido tem 1 único filho
  - Remove-se o nó
  - "Puxa-se" o filho para o lugar do pai

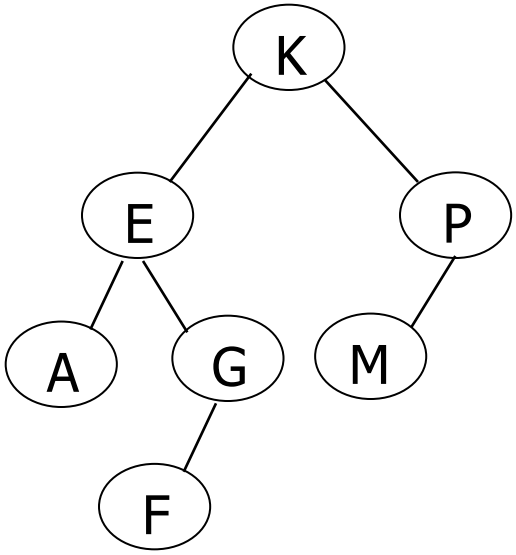
ABB



ABB



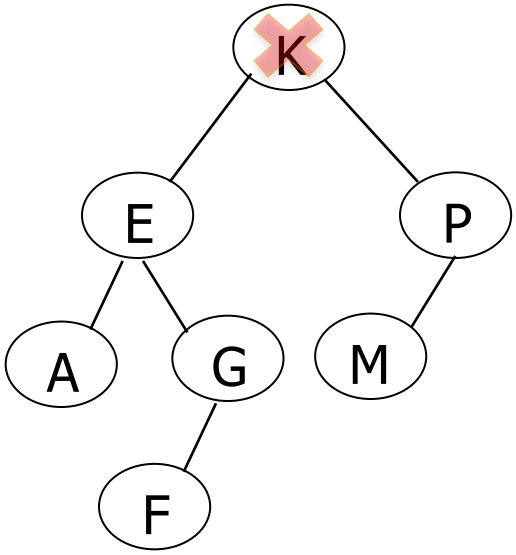
ABB



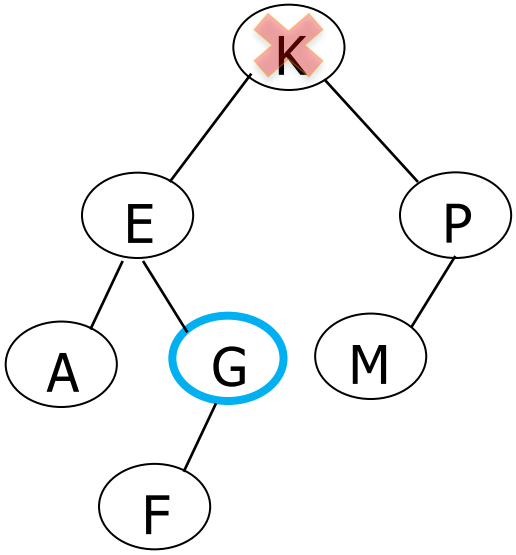
ABB

- Caso 3 (remover K): o nó p a ser removido tem 2 filhos
  - Acha-se a maior chave da subárvore esquerda
  - p recebe o valor dessa chave
  - Remove-se a maior chave da subárvore esquerda

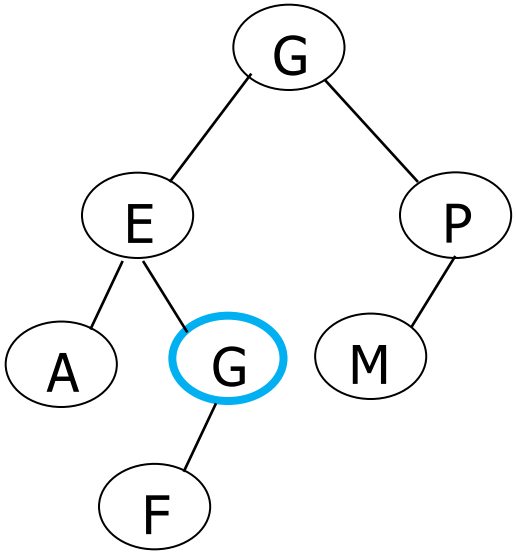
ABB



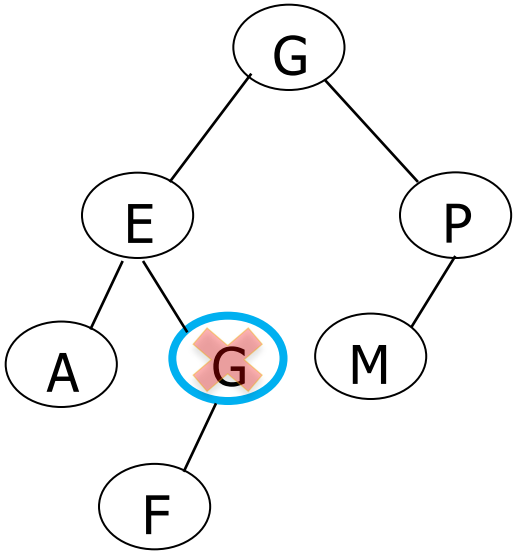
ABB



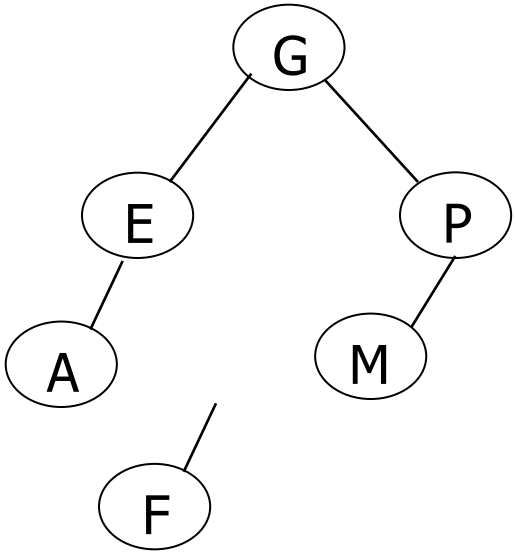
ABB



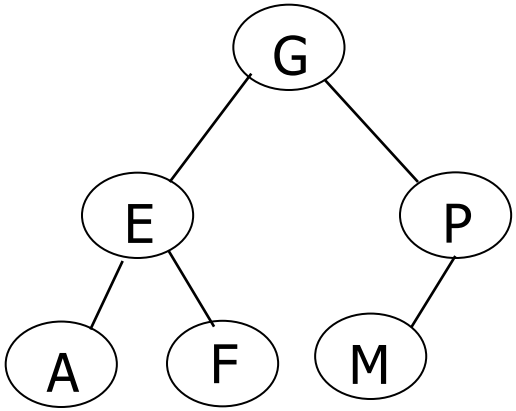
ABB



ABB



ABB



ABB

- Exercício para casa: Implementação da sub-rotina de remoção de um elemento da árvore binária de busca

ABB

- Vantagens

  - Se os nós estão espalhados **uniformemente**, consulta é rápida mesmo para uma grande quantidade de dados

    - Divide-se em dois o espaço de busca restante em cada passo da busca
    - $O(\log_2 N)$

ABB

- Contra-exemplos
  - Inserção dos elementos de maneira ordenada
    - A, B, C, D, E, ..., Z
    - 1000, 999, 998, ..., 1

ABB

- O desbalanceamento da árvore pode tornar a busca tão ineficiente quanto a busca seqüencial (no pior caso)
  - O(N)
- Solução?

ABB

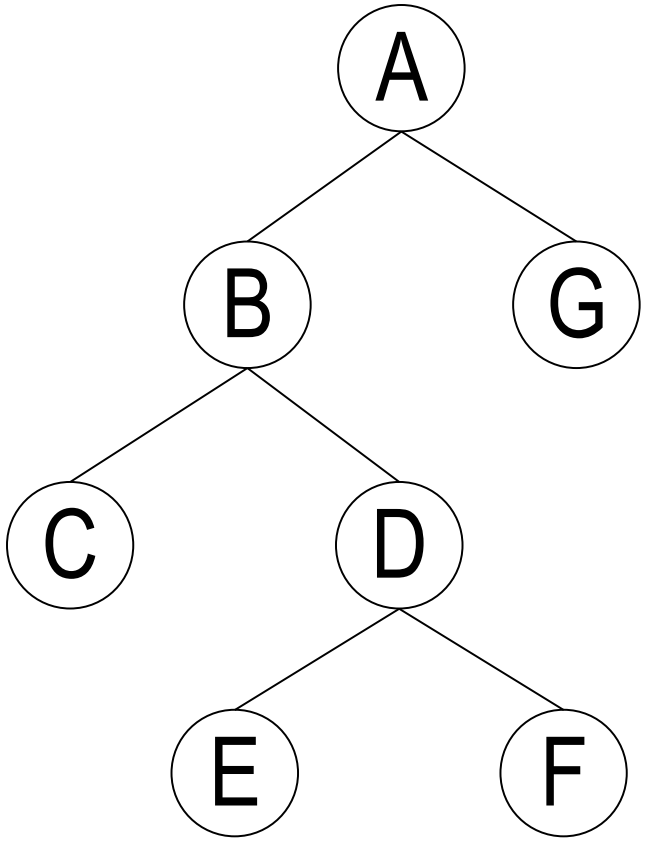
- O desbalanceamento da árvore pode tornar a busca tão ineficiente quanto a busca seqüencial (no pior caso)
  - O(N)

- Solução?
  Balancear a árvore quando necessário

Árvores AVL

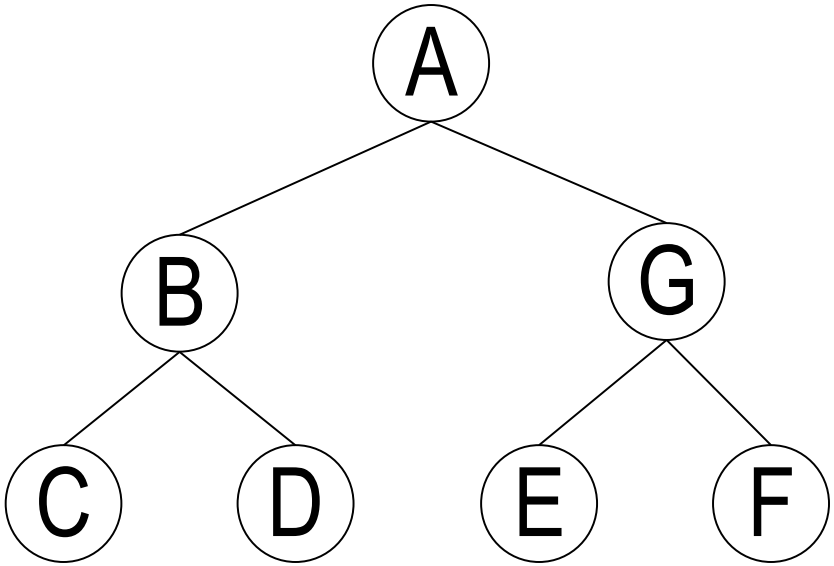
Conceitos

- Árvore estritamente binária
  - Os nós tem 0 ou 2 filhos
    - Todo nó interno tem 2 filhos
    - Somente as folhas não têm filhos



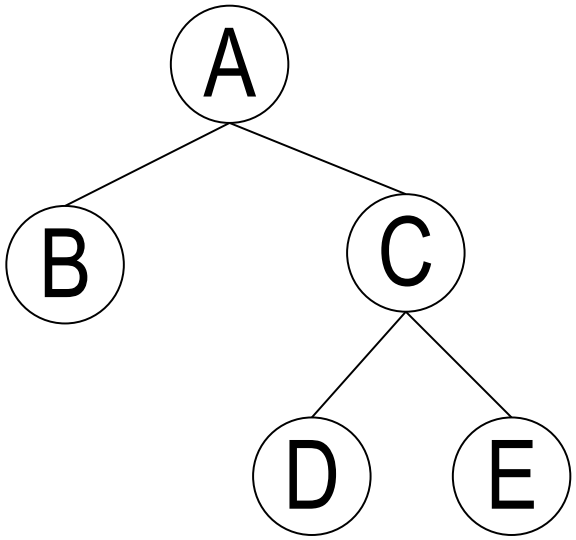
Conceitos

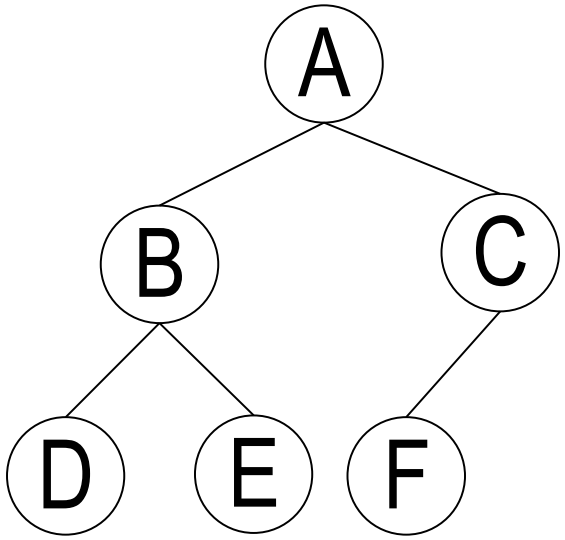
- Árvore binária completa (ou cheia)
  - Árvore estritamente binária
  - Todos os nós folha no mesmo nível



Conceitos

- Uma árvore binária é dita balanceada se, para cada nó, as alturas de suas duas subárvores diferem de, no máximo, 1



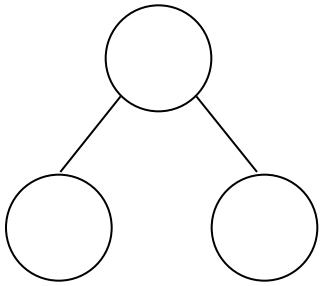


AVL

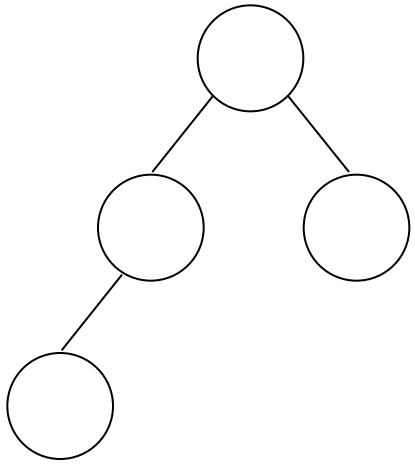
- Árvore binária de busca balanceada
  - Para cada nó, as alturas das subárvores diferem em 1, no máximo
  - Proposta em 1962 pelos matemáticos russos G.M. Adelson-Velskki e E.M. Landis
    - Métodos de inserção e remoção de elementos da árvore são realizados de forma que ela se mantenha balanceada

AVL: quem é e quem não é?

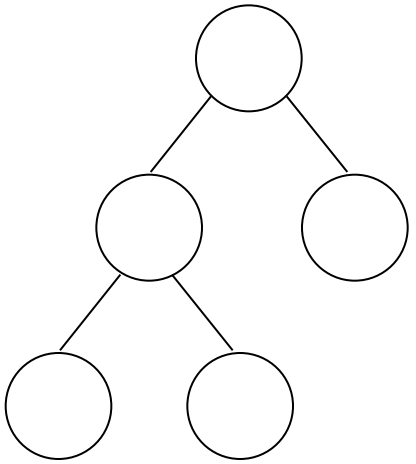
(a)



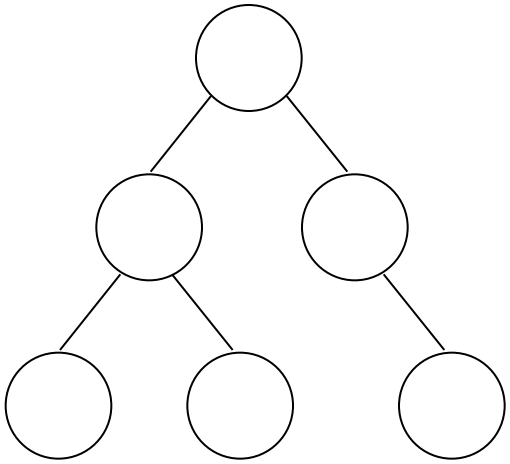
(b)



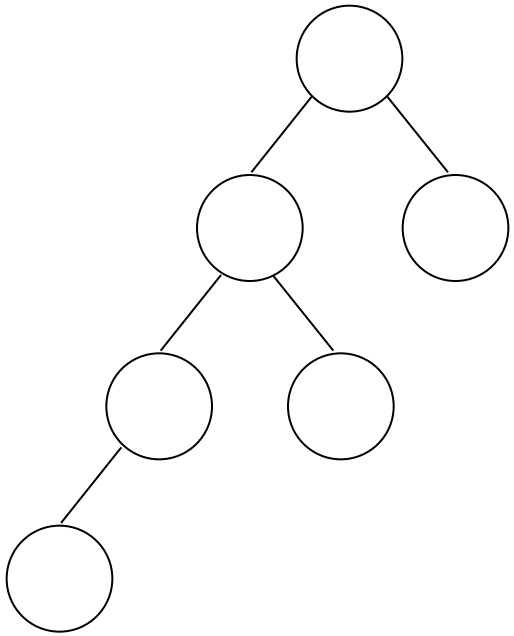
(c)



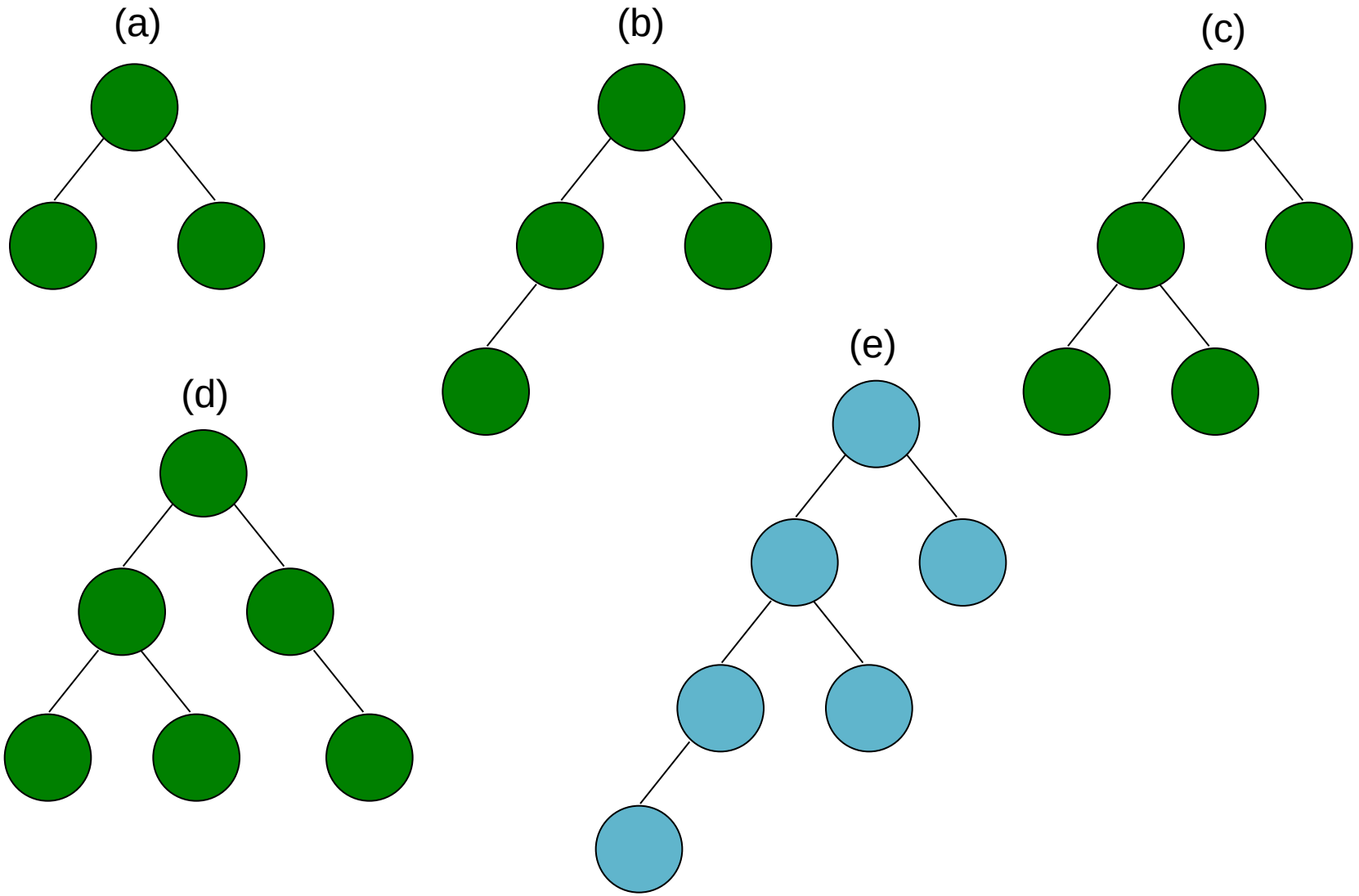
(d)



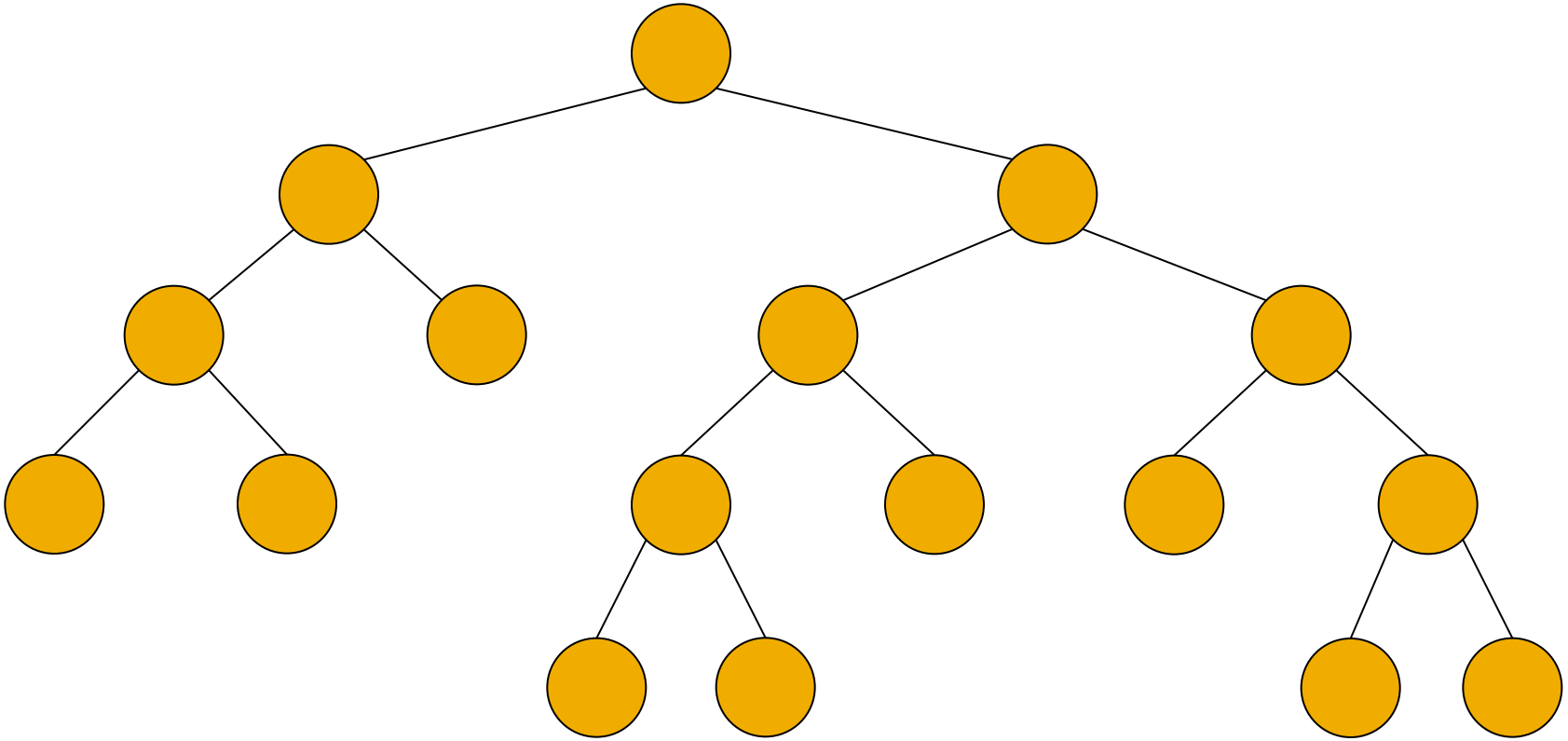
(e)



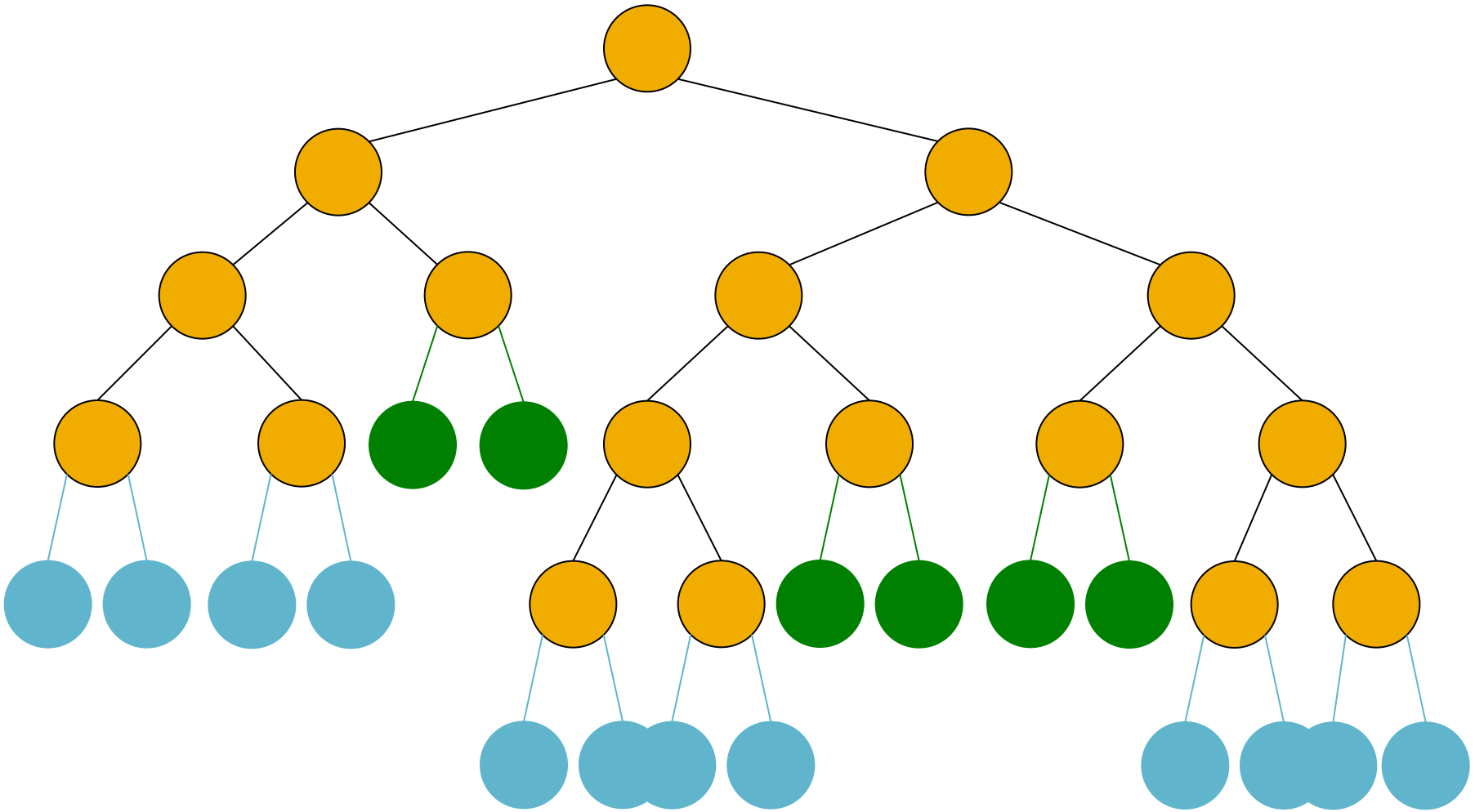
AVL: quem é e quem não é?



Pergunta: a árvore abaixo é AVL?



Exercício: onde se pode incluir um nó para a AVL continuar sendo AVL?



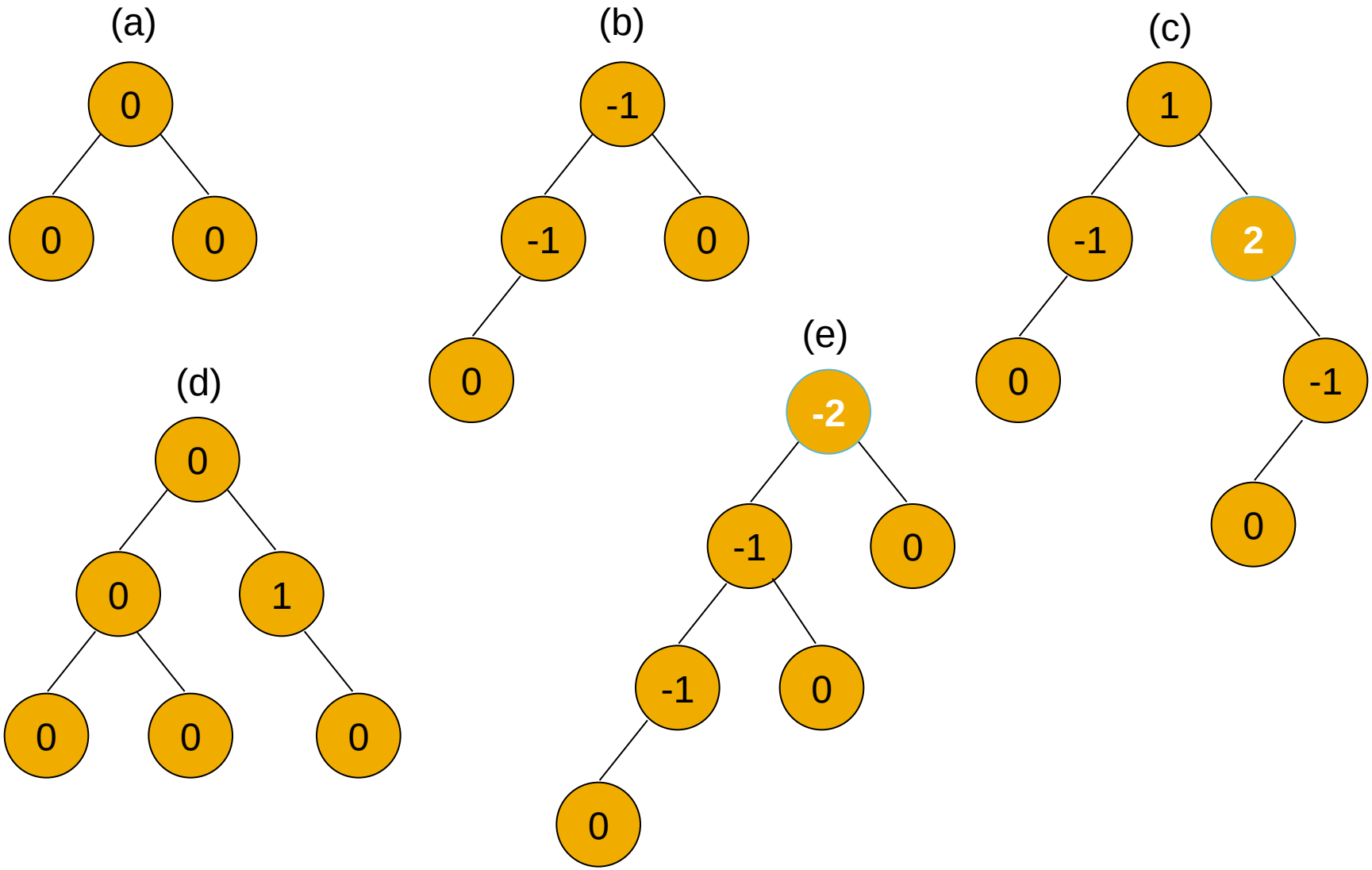
AVL

- Como é que se sabe quando é necessário balancear a árvore?
  - Se a diferença de altura das subárvores deve ser 1, no máximo, então temos que procurar diferenças de altura maiores do que isso
  - Possível solução: cada nó pode armazenar a diferença de altura de suas subárvores
    - Convencionalmente chamada de fator de balanceamento (FB) do nó

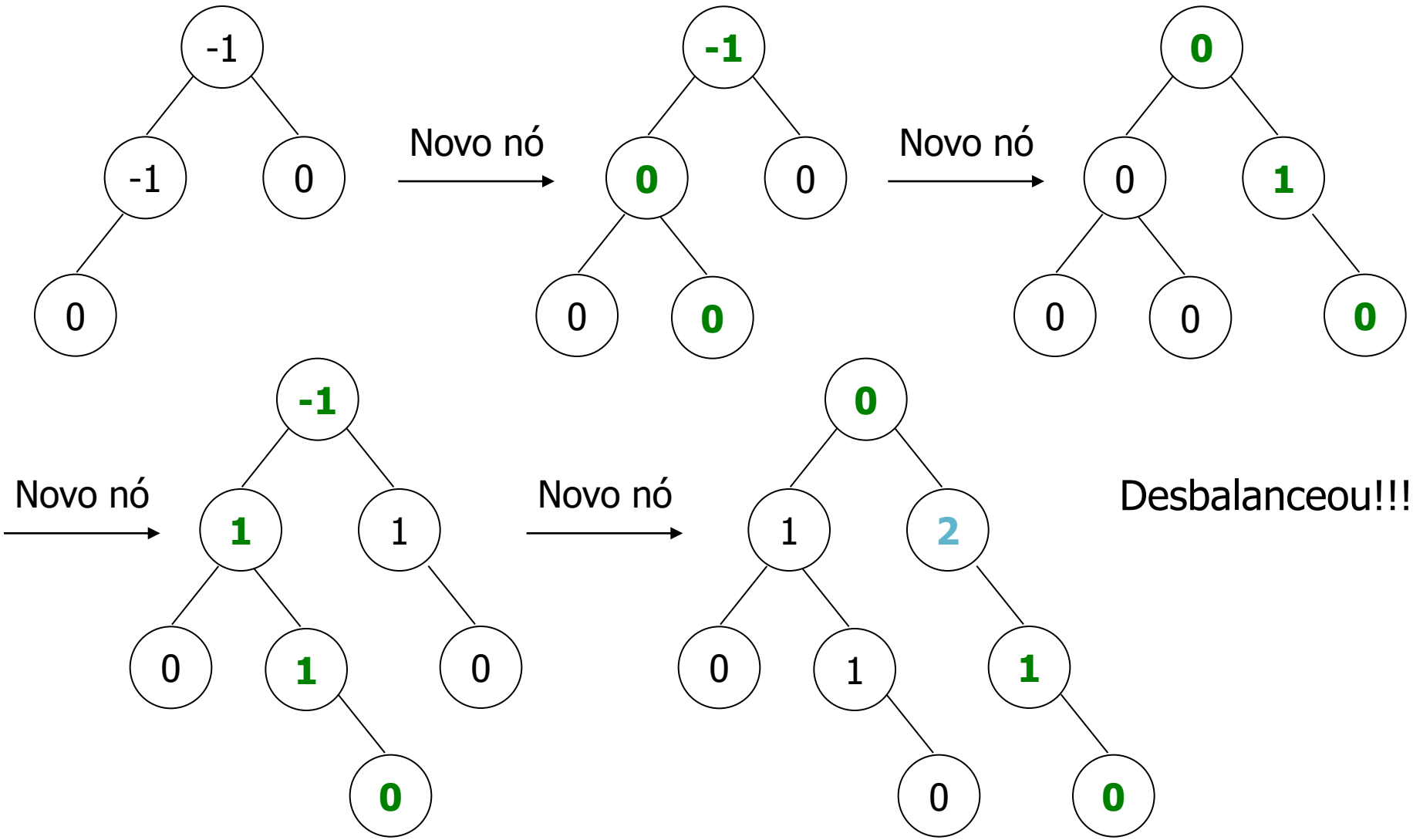
AVL

- Fatores de balanceamento dos nós
  - Altura da subárvore direita menos altura da subárvore esquerda
    - Hd-He
  - Atualizados sempre que a árvore é alterada (elemento é inserido ou removido)
  - Quando um fator é 0, 1 ou -1, a árvore está balanceada
  - Quando um fator se torna 2 ou -2, a árvore não está mais balanceada

AVL: quem é e quem não é



AVL: exemplo de desbalanceamento

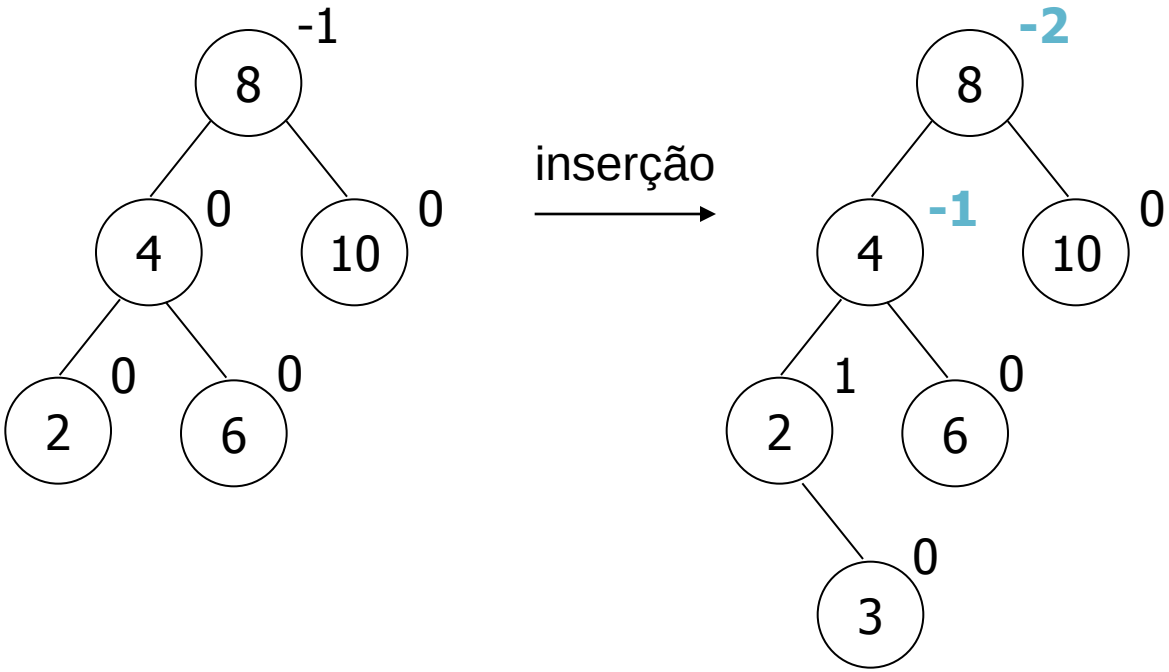


AVL

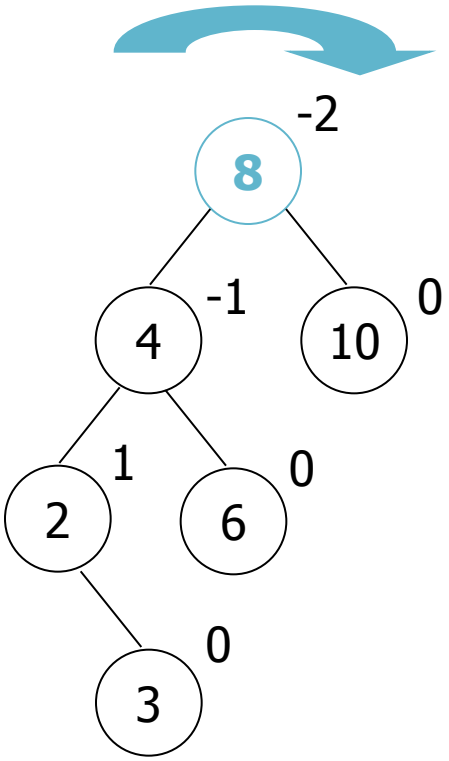
- Controle do balanceamento
  - Altera-se o algoritmo de inserção para balancear a árvore quando ela se tornar desbalanceada após uma inserção (nó com FB 2 ou -2)
    - Rotações
      - Se árvore pende para esquerda (FB negativo), rotaciona-se para a direita
      - Se árvore pende para direita (FB positivo), rotaciona-se para a esquerda
    - 2 casos podem acontecer

AVL: primeiro caso

- Raiz de uma subárvore com FB -2 (ou 2) e um nó filho com FB -1 (ou 1)
  - Os fatores de balanceamento têm sinais iguais: subárvores de nó raiz e filho pendem para o mesmo lado

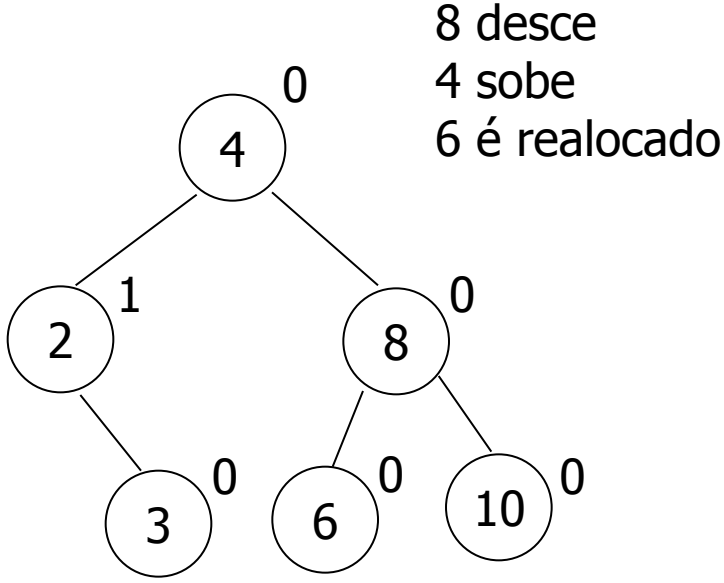


AVL: primeiro caso



Pendendo para
a esquerda

Rotação do nó pai
para a direita



8 desce
4 sobe
6 é realocado

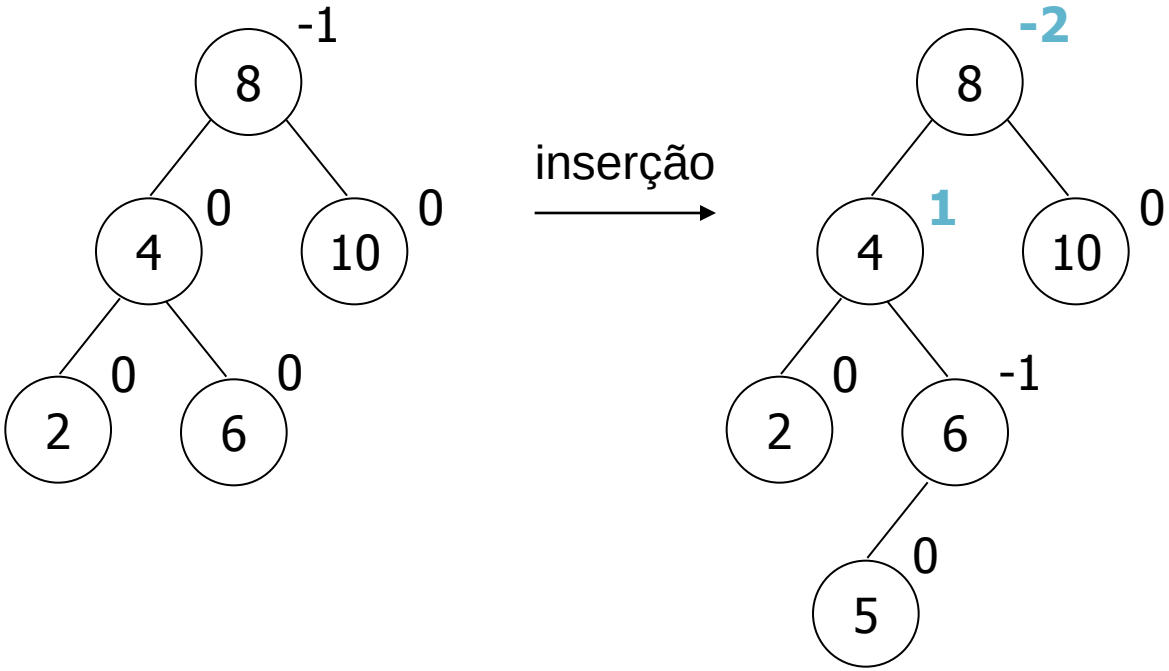
Árvore balanceada!!!

AVL: primeiro caso

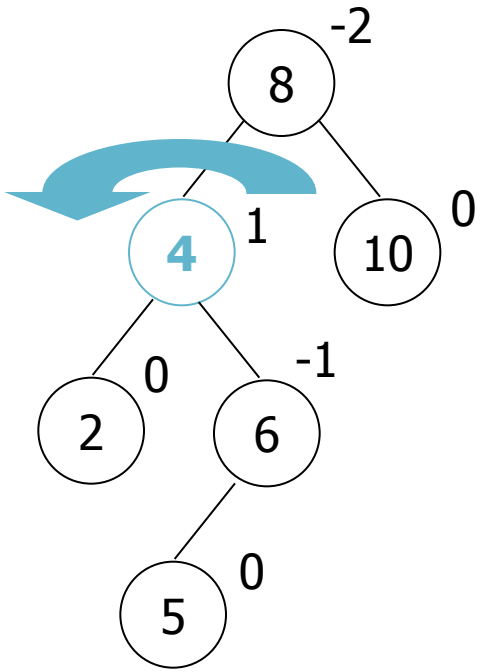
- Quando subárvores do pai e filho pendem para um mesmo lado
  - Rotação simples para o lado oposto
  - Às vezes, é necessário realocar algum elemento, pois ele perde seu lugar correto na árvore

AVL: segundo caso

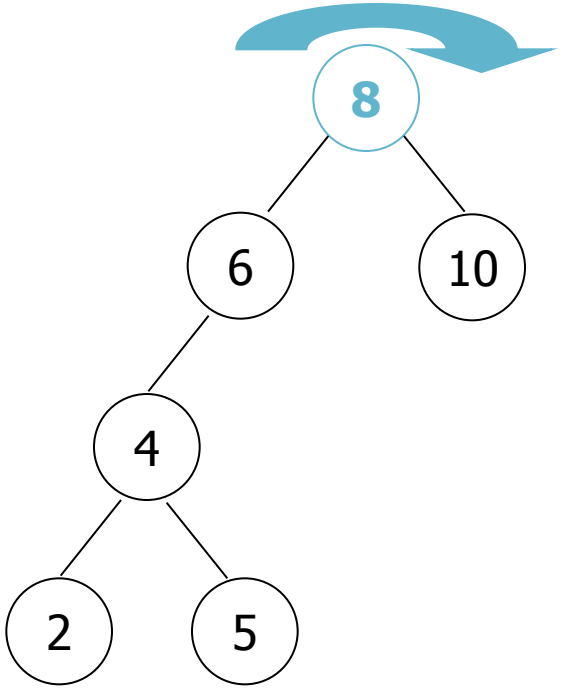
- Raiz de uma subárvore com FB -2 (ou 2) e um nó filho com FB 1 (ou -1)
  - Os fatores de balanceamento têm sinais opostos: subárvore de nó raiz pende para um lado e subárvore de nó filho pende para o outro



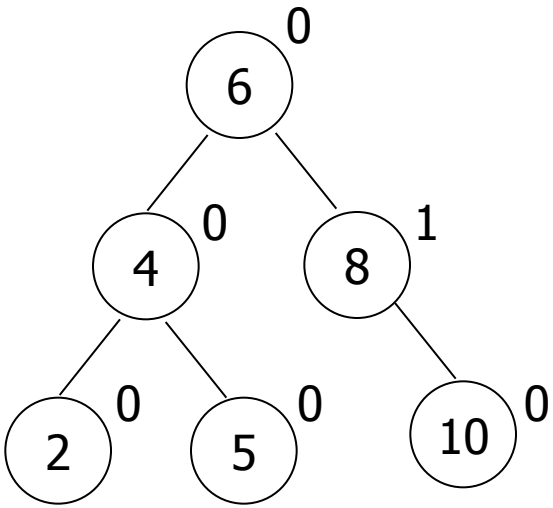
AVL: segundo caso



Rotação do nó filho
para a esquerda



Rotação do nó pai
para a direita



Árvore balanceada!!!

AVL: segundo caso

- Quando subárvores do pai e filho pendem para lados opostos
  - Rotação dupla
    - Primeiro, rotaciona-se o filho para o lado do desbalanceamento do pai
    - Em seguida, rotaciona-se o pai para o lado oposto do desbalanceamento
  - Às vezes, é necessário realocar algum elemento, pois ele perde seu lugar na árvore

AVL

- Algoritmo de inserção em AVL
  - A cada inserção, verifica-se o balanceamento da árvore
    - Se necessário, fazem-se as rotações de acordo com o caso (sinais iguais ou não)
  - Em geral, armazena-se uma variável de balanceamento em cada nó para indicar o FB

AVL

- Declaração

```
typedef int elem;

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

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

AVL

- Exercício: Inserir os elementos 10, 3, 2, 5, 7 e 6 em uma árvore AVL e balancear quando necessário

AVL

- Exercício: Inserir os elementos 1, 2, 3, ..., 10 em uma árvore AVL e balancear quando necessário

Créditos

Material gentilmente cedido pelo
prof. Thiago A. S. Pardo