

Árvores

SCC-502 – Algoritmos e Estruturas de Dados I

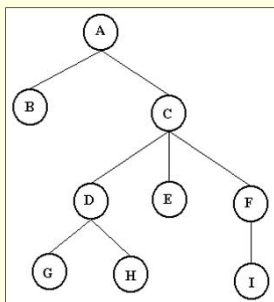
Thiago A. S. Pardo

Listas e árvores

- **Listas lineares**
 - Um nó após o outro, adjacentes
 - Sem relações hierárquicas entre os nós, em geral
- Diversas aplicações necessitam de estruturas mais complexas do que as listas estudadas até agora
 - **Listas não lineares:** árvores, grafos, etc.

Árvores

- Exemplo



Árvores

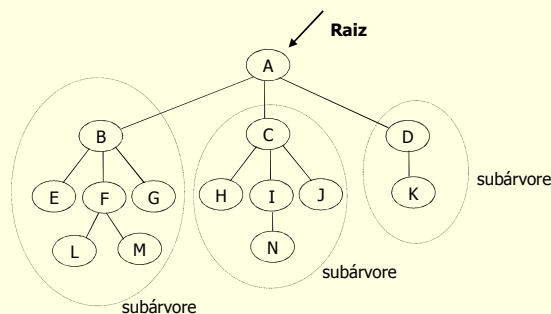
- Motivações para usá-las
 - Inúmeros problemas podem ser representados e tratados por árvores
 - Árvores admitem tratamento computacional eficiente quando comparadas a estruturas mais genéricas como os grafos (os quais, por sua vez são mais flexíveis e, portanto, complexos)
 - Ótimas para busca!

Árvores

- **Definição**

- Uma árvore T , ou simplesmente uma árvore, é um conjunto finito de elementos denominados nós ou vértices tais que
 - $T=0$ 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 (≥ 1) 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 subárvore de T
 - Os nós raízes w_1, w_2, \dots, w_j das subárvores de V são chamados filhos de V
 - V é chamado pai de w_1, w_2, \dots, w_j
 - Os nós w_1, w_2, \dots, w_j são irmãos
 - Se Z é filho de w_1 , então w_2 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 seus nós

Árvores

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

Árvores

- Caminho, comprimento do caminho
 - Uma seqüência de nós distintos v_1, v_2, \dots, v_k , tal que existe sempre entre nós consecutivos (isto é, entre v_1 e v_2 , entre v_2 e $v_3, \dots, v_{(k-1)}$ e v_k) a relação "é filho de" ou "é pai de" é denominada um caminho na árvore; diz-se que v_1 alcança v_k e que v_k é alcançado por v_1
 - Um caminho de v_k 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 $h(V)$

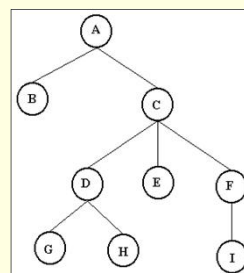
Árvores

■ Árvore ordenada

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

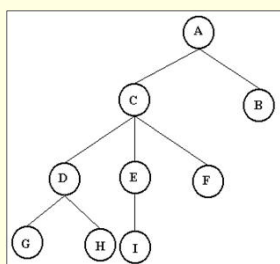
Árvores

■ Árvore ordenada



Árvores

■ Árvore não ordenada



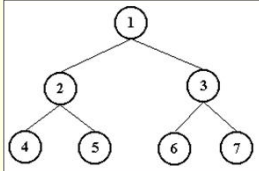
Árvores

■ 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

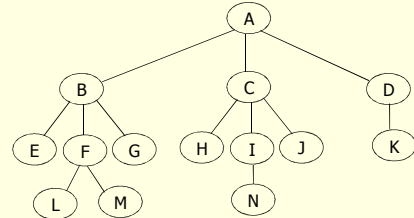
Árvores

- Exemplo de árvore cheia de grau 2



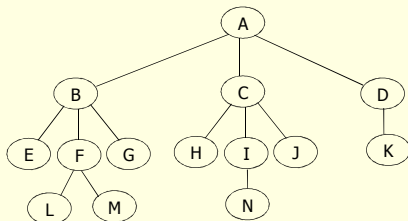
Árvores

- Considere a árvore abaixo
 - Quantas subárvores A tem?



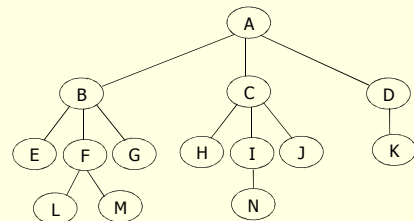
Árvores

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



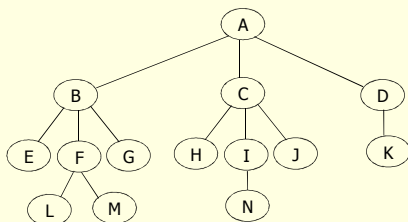
Árvores

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



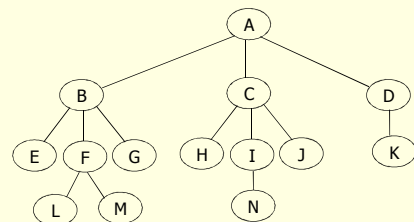
Árvores

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



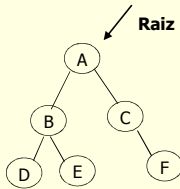
Árvores

- 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

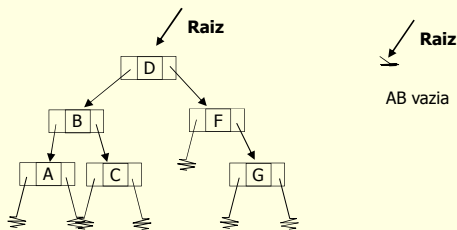
Árvores binárias

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



Árvores binárias

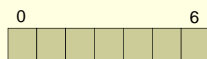
Pergunta

- Quantos ponteiros são necessários para se percorrer uma árvore binária completamente?
- Quantos são necessários para percorrer qualquer tipo de árvore?

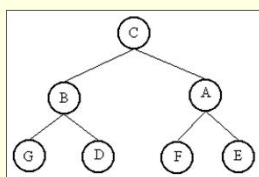
Árvores binárias

- Representação estática e seqüencial de árvores binárias

- Vetor



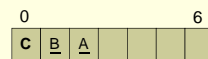
- Como colocar a árvore abaixo nesse vetor?



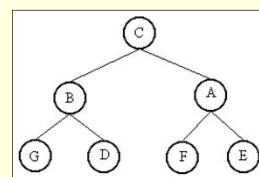
Árvores binárias

- Representação estática e seqüencial de árvores binárias

- Vetor



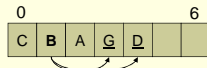
- Como colocar a árvore abaixo nesse vetor?



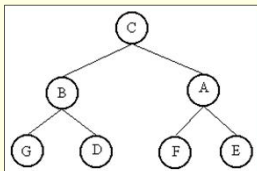
Árvores binárias

■ Representação estática e seqüencial de árvores binárias

■ Vetor



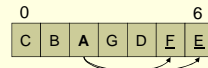
■ Como colocar a árvore abaixo nesse vetor?



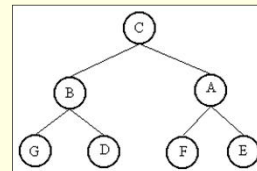
Árvores binárias

■ Representação estática e seqüencial de árvores binárias

■ Vetor

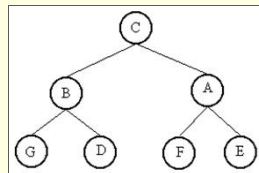


■ Como colocar a árvore abaixo nesse vetor?



Árvores binárias

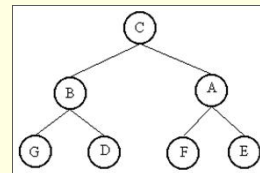
■ Representação estática e seqüencial de árvores binárias



■ Como saber quem é filho de quem?

Árvores binárias

■ Representação estática e seqüencial de árvores binárias



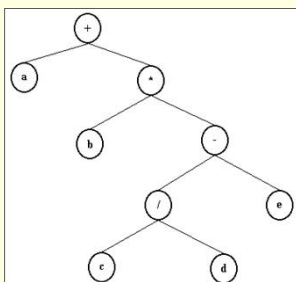
■ Como saber quem é filho de quem?

- Filhos de i são $2i+1$ e $2i+2$

Árvores binárias

■ Exercício: represente a árvore abaixo em um vetor

■ 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
 - Criar árvore
 - Verificar se a árvore está vazia
 - Imprimir elementos da 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
 - Finalizar árvore

Árvores binárias

- **Exercício**
 - Implementar o TAD árvore binária

Exercício

- Em duplas (**valendo nota**)
 - Esquematize/desenhe/explique como seria a função de remoção de um elemento da árvore

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

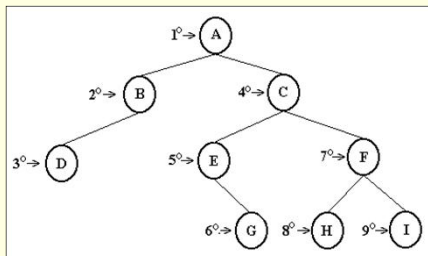
- Há três maneiras de se percorrer árvores binárias
 - Em função da ordem de visitas aos nós
 - **Pré-ordem**: visita-se nó raiz primeiro e depois as subárvores esquerda e direita, nessa ordem
 - **Em-ordem**: visita-se subárvore esquerda, nó raiz e subárvore direita, nessa ordem
 - **Pós-ordem**: visita-se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



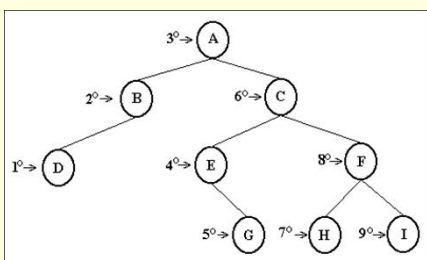
Percurso em árvores binárias

■ 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



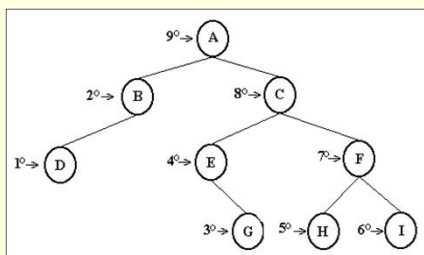
Percurso em árvores binárias

■ 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

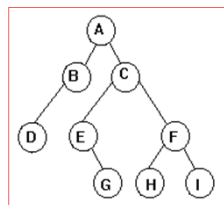
Percurso em árvores binárias

- Possível 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

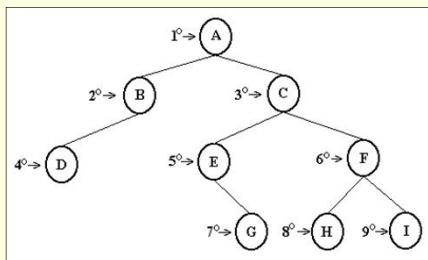


Percurso em árvores

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

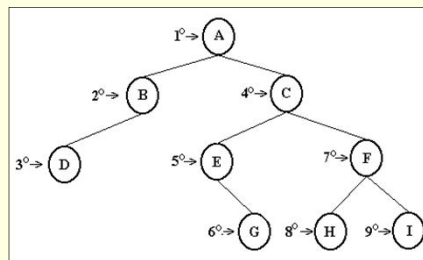
Percurso em árvores

- **Em largura**
 - Um nível de cada vez



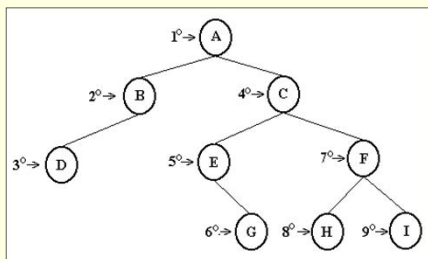
Percurso em árvores

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



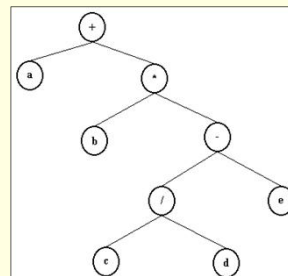
Percurso em árvores

- **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

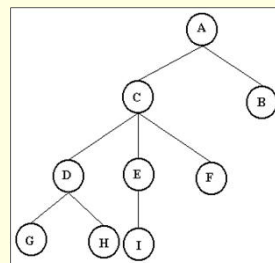


Percurso em árvores

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

Percurso em árvores

- Teste a estratégia com a pilha explícita



Percurso em árvores

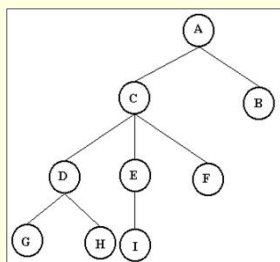
- Em **largura**
 - Como implementar?

Percurso em árvores

- Em **largura**
 - **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



Largura vs. profundidade

- Ao buscar um elemento

- **Profundidade**
 - **Vantagens?**
 - **Desvantagens?**

Largura vs. profundidade

- Ao buscar um elemento
 - **Profundidade**
 - Menos memória para guardar nós não visitados
 - Pode buscar “para sempre”, demorando mais para achar o elemento

Largura vs. profundidade

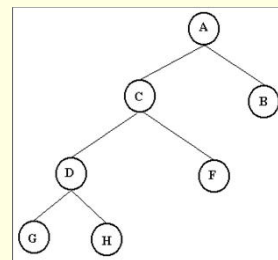
- Ao buscar um elemento
 - **Largura**
 - Vantagens?
 - Desvantagens?

Largura vs. profundidade

- Ao buscar um elemento
 - **Largura**
 - Acha o elemento mais rapidamente
 - Mais memória para guardar nós não visitados

Largura vs. profundidade

- Use as estratégias para buscar o nó F
 - Quem se sai melhor em termos de nós visitados e de memória?



Percurso em árvores

- **Árvores encadeadas**
 - Mais ponteiros, mas pode facilitar busca e outras operações
 - Ponteiros podem ser configurados para qualquer tipo de busca
 - Filhos podem apontar para pais, para irmãos, para avós, etc.

