Teoria dos Grafos

Árvores

Árvore - Introdução

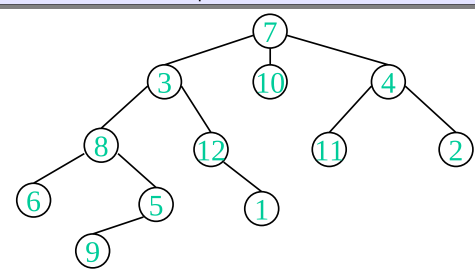
- Em nosso dia-a-dia nos deparamos com muitos exemplos de árvores:
  - Árvore genealógica.
  - Organograma de uma empresa.
  - Tabela de um torneio esportivo.
- Na computação:
  - Organização da estrutura de arquivos (diretório).
  - Armazenamento e busca eficiente de dados.
  - Ordenação.
  - Árvores de decisão.

Árvore Livre

- Uma árvore (livre) é um grafo acíclico, não orientado e conectado.
- Uma floresta é um grafo acíclico, não orientado mas, possivelmente, desconectado.
- Considerando que $G = (V, E)$ é um grafo não orientado, é equivalente dizer:
  1. G é uma árvore.
  2. Um par de vértice qualquer (v, w) de G está conectado por apenas um caminho.
  3. G é conectado. A remoção de uma aresta desconecta G.
  4. G é conectado e $|E| = |V| - 1$.
  5. G é acíclico e $|E| = |V| - 1$.
  6. G é acíclico. A adição de uma aresta cria um ciclo em G.

Árvore Enraizada

- Tipo especial de árvore que apresenta um vértice (raiz) que se distingue dos demais.
- Utilizamos o termo nó para fazer referência aos vértices.



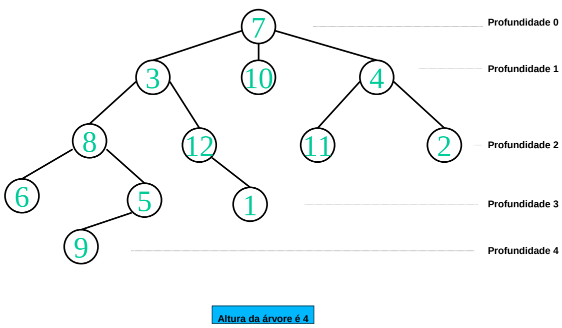
Algumas Definições

- Seja x um nó de uma árvore enraizada T com raiz r:
  - **Ancestral:** é qualquer nó y no caminho de r a x.
  - **Descendente:** x é um descendente de y se y é ancestral de x.
  - **Ancestral Próprio:** y é ancestral próprio de x se y é ancestral de x e $y \neq x$.
  - **Descendente Próprio:** y é descendente próprio de x se y é descendente de x e $y \neq x$.
  - **Sub-árvore enraizada em x:** árvore induzida pelos descendentes de x, com x sendo a raiz.
  - **Filho:** x é filho de y se ele é um descendente direto.
  - **Pai:** é o ancestral próprio mais próximo. A raiz é o único nó sem pai.

Algumas Definições

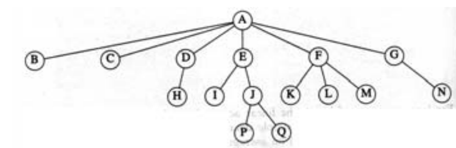
- **Folha:** um nó sem filhos.
- **Nó interno:** um nó que não é folha.
- **Grau:** o grau de y é o número de filhos de y.
- **Profundidade:** o comprimento desde a raiz r até x é a profundidade de x em T.
- **Altura:**
  - a altura de um nó em uma árvore é o maior comprimento do nó até uma folha.
  - A altura de uma árvore é a altura de sua raiz.
  - Altura da árvore é a maior profundidade de qualquer nó da árvore.

Exemplo



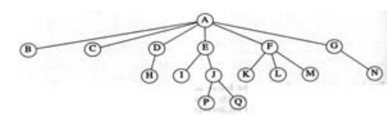
Implementação de Árvores

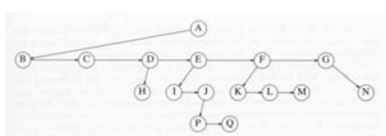
- Além da informação de cada nó, um link para cada um dos filhos.
- Inconveniente: não sabemos a priori a quantidade de filhos em cada nó.



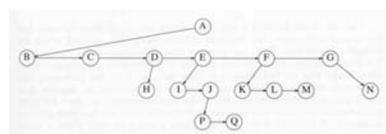
Implementação de Árvores

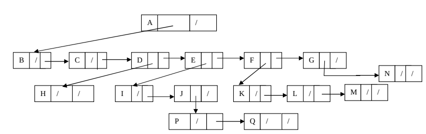
- Os filhos de um nó são mantidos em uma lista encadeada.





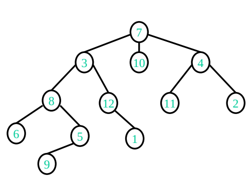
Implementação de Árvores

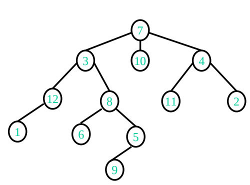




Árvore Ordenada

- Árvore enraizada em que os filhos de cada nó estão ordenados.

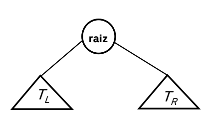




Árvores enraizadas iguais.
Árvores ordenadas diferentes.

Árvore Binária

- Estrutura de nós que é definida recursivamente através de um conjunto de nós:
  - Não contém nenhum nó, ou;
  - Formada por 3 conjuntos disjuntos: um nó raiz, duas sub-árvores binárias (direita e esquerda).

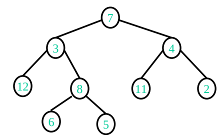


Árvore Binária – Conceitos Importantes

- **Árvore vazia ou nula:** não contém nenhum nó.
- **Filho da esquerda:** raiz da sub-árvore da esquerda (quando houver).
- **Filho da direita:** raiz da sub-árvore da direita (quando houver).
- **Filho ausente:** quando a sub-árvore dé a árvore nula.

Árvore Binária Cheia

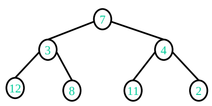
- Cada nó ou é uma folha ou tem grau exatamente 2.



O número de nós internos de uma árvore binária cheia é $f - 1$, onde f é o número de folhas.

Árvore Binária Completa

- Árvore binária em que todas as folhas estão em uma mesma profundidade e todos os nós internos têm grau 2.



O número de nós internos de uma árvore binária completa é $2^h - 1$, onde h é a altura da árvore.

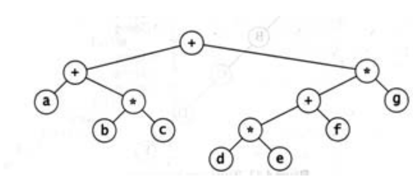
Árvore k-ária Completa

- Em uma árvore posicional, os filhos de um nó são rotulados como inteiros distintos.
- Árvore k-ária é uma árvore posicional onde os filhos com rótulos maiores do que k são ausentes.
- Árvore k-ária completa é uma árvore k-ária onde todas as folhas têm a mesma profundidade e todos os nós internos têm grau k.
- Uma árvore binária é uma árvore k-ária com k = 2.

O número de nós internos de uma árvore k-ária completa é $k^h - 1/ k - 1$, onde h é a altura da árvore.

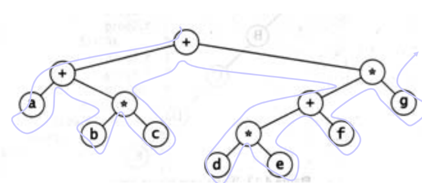
Aplicação: Árvores de Expressões

•Seja a expressão (a+b*c)+((d*e+f)*g):
- –Folhas são operandos.
- –Nós internos são operadores.



Árvores de Expressões

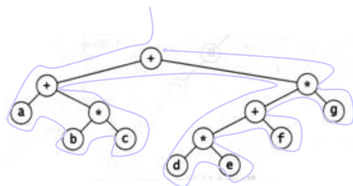
- Caminhamento em ordem:
  - – produz expressão na notação infixa.



((a+(b*c))+(((d*e)+f)*g))

Árvores de Expressões

- Caminhamento em pós-ordem:
  - produz expressão na notação pósfixa.



Expressão pósfixa:
abc*+de*f+g*+

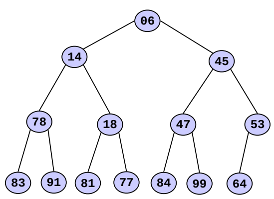
Árvore Binária de Pesquisa - ABP

- Árvore binária em que cada nó tem uma chave que não é menor do que as chaves dos nós de sua sub-árvore esquerda e não é maior do quas chaves dos nós da sub-árvore direita.

ABP **Não é ABP**

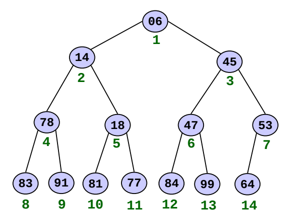
Heap Binário

- Árvore binária com duas propriedades:
  - Estrutura: árvore binária quase completa. O último nível pode não ser completado.
  - Ordem: todo filho é maior (ou igual) do que o pai.



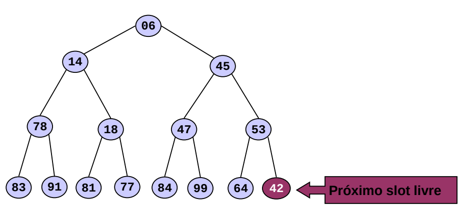
Implementação de Heap Binário

- Usar um array:
  - Parent(i) = ⌊i/2⌋
  - Left(i) = 2i
  - Right(i) = 2i + 1

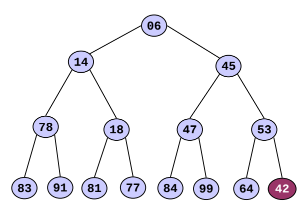


Inserção em Heap Binário

- Manter propriedades de ordem e estrutura.
- Inserir no *slot* livre e depois procurar lugar correto.

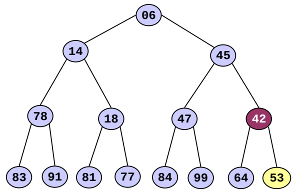


Inserção em Heap Binário

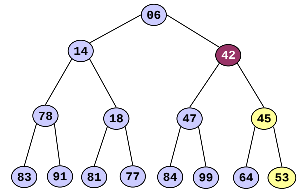


Trocar com o pai, se necessário

Inserção em Heap Binário

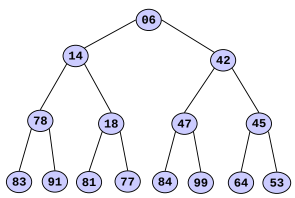


Inserção em Heap Binário



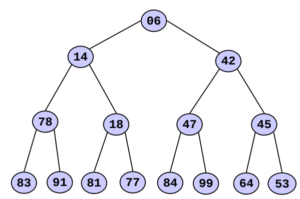
Propriedade de ordem OK!!

Inserção em Heap Binário



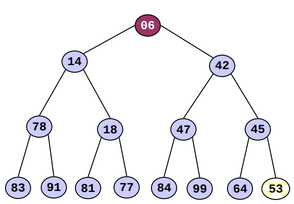
Remoção em Heap Binário

- Manter propriedades de ordem e estrutura.
- Sempre remove o menor elemento.



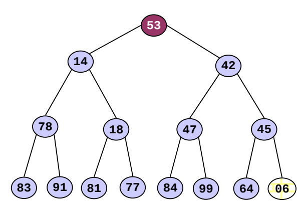
Remoção em Heap Binário

- Manter propriedades de ordem e estrutura.
- Sempre remove o menor elemento.



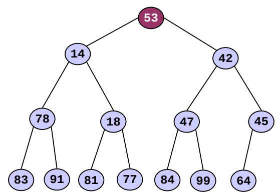
Remoção em Heap Binário

- Manter propriedades de ordem e estrutura.
- Sempre remove o menor elemento.



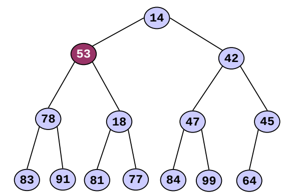
Remoção em Heap Binário

- Manter propriedades de ordem e estrutura.
- Sempre remove o menor elemento.



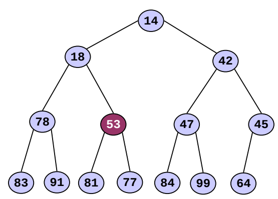
Remoção em Heap Binário

- Manter propriedades de ordem e estrutura.
- Sempre remove o menor elemento.



Remoção em Heap Binário

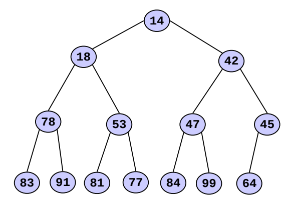
- Manter propriedades de ordem e estrutura.
- Sempre remove o menor elemento.



Propriedade de ordem OK!!

Remoção em Heap Binário

- Manter propriedades de ordem e estrutura.
- Sempre remove o menor elemento.



Aplicação em Ordenação: HeapSort

- Inserir N itens no heap.
- executar N operações de remoção.
- O(N log N).
- Não é necessário armazenamento extra.