Aula 12 - Estrutura de dados 1 – Uniube

Prof. Marcos Lopes

34 9 9878 0925

Estrutura de dados do tipo árvore binária (parte 2):

Varredura D-R-E: direita - raiz - esquerda

A varredura DRE é também uma varredura interessante no contexto de árvores binárias.

Exercício 1) Com base no código do exercício da aula passada criar uma função que implementa a varredura DRE e verifique o resultado.

Altura e profundidade:

A altura de um nó x em uma árvore binária é a distância entre x e o seu descendente mais afastado.

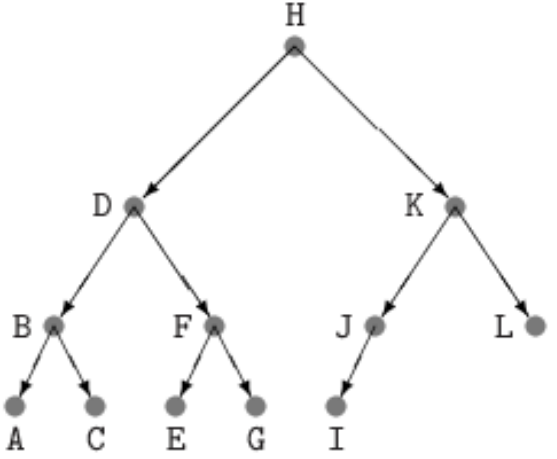
Mais exatamente, a altura de x é o número de passos no mais longo caminho que leva de x até uma folha.

Os caminhos a que essa definição se refere são os obtido pela iteração das instruções x = x->esq e x = x->dir, em qualquer ordem.

A altura (= height) de uma árvore é a altura da raiz da árvore.

Uma árvore com um único nó tem altura 0.

A árvore da figura tem altura 3.



Veja como a altura de uma árvore com raiz r pode ser calculada com uma função:

```
// Devolve a altura da árvore binária
// cuja raiz é r.

int altura (arvore r) {
   if (r == NULL)
      return -1; // altura da árvore vazia
   else {
      int he = altura (r->esq);
      int hd = altura (r->dir);
      if (he < hd) return hd + 1;
      else return he + 1;
   }
}
```

Exercício 2) Teste o código acima no programa do exercício 1.

Qual a relação entre a altura, digamos h, e o número de nós de uma árvore binária?

Se a árvore tem n nós então

$\lg(n) \leq h \leq n-1$,

onde $\lg(n)$ denota $\lfloor \log n \rfloor$. Uma árvore binária de altura $n-1$ é um tronco sem galhos: cada nó tem no máximo um filho.

No outro extremo, uma árvore de altura $\lg(n)$ é quase completa: todos os níveis estão lotados exceto talvez o último.

n	lg(n)
4	2
5	2
6	2
10	3
64	6
100	6
128	7
1000	9
1024	10
1000000	19

Uma árvore binária é balanceada (ou equilibrada) se, em cada um de seus nós, as subárvores esquerda e direita tiverem aproximadamente a mesma altura.

Uma árvore binária balanceada com n nós tem altura próxima de log n. (A árvore do exemplo anterior é balanceada).

Sempre que possível, convém trabalhar com árvores que são balanceadas. Mas isso não é fácil se a árvore aumenta e diminui ao longo da execução do seu programa.

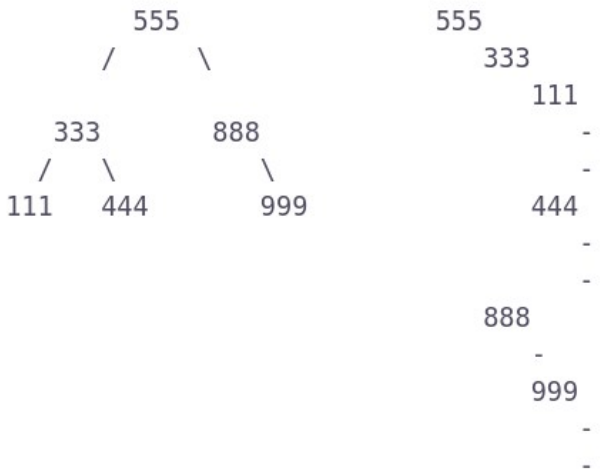
Profundidade de um nó:

A profundidade (= depth) de um nó s em uma árvore binária com raiz r é a distância de r a s. Mais exatamente, a profundidade de s é o comprimento do único caminho que vai de r até s. Por exemplo, a profundidade de r é 0 e a profundidade de r->esq é 1.

Uma árvore é balanceada se todas as suas folhas têm aproximadamente a mesma profundidade.

Exercícios 3)

a) Escreva uma função que imprima o conteúdo de cada nó de uma árvore binária. Faça uma indentação (recuo de margem) proporcional à profundidade do nó. Segue um exemplo de árvore e sua representação (os caracteres '-' representam NULL):




```
555
    333
        111
            -
            -
        444
            -
            -
    888
        -
        999
            -
            -
```

Nós com campo pai:

Em algumas aplicações é conveniente ter acesso direto ao pai de cada nó.

Para isso, é preciso acrescentar um campo pai a cada nó:

```
typedef struct reg {
   int         conteudo;
   struct reg *pai;
   struct reg *esq, *dir;
} noh;
```

Como a raiz da árvore não tem pai, é preciso tomar uma decisão de projeto a respeito do valor do campo pai nesse caso.

Se r é a raiz da árvore, poderíamos dizer

r->pai == NULL.

Exercícios 4)

a) Com base no código do exercício 2, adicione a propriedade “pai” ao “noh” e ajuste o código para que o campo “pai” seja sempre preenchido adequadamente.

b) Adicione ao programa acima uma função que recebe um nó “n” e mostra os nós (conteudos) do caminho de “n” até na raiz (inclusive).