4.2 二叉树

二叉树(binary tree)是一棵树，其中每个节点都不能有多于两个的儿子。

图 4-11 显示一棵由一个根和两棵子树组成的二叉树，子树 T_L 和 T_R 均可能为空。

二叉树的一个性质是一棵平均二叉树的深度要比节点个数 N 小得多，这个性质有时很重要。分析表明，其平均深度为 $O(\sqrt{N})$，而对于特殊类型的二叉树，即**二叉查找树**(binary search tree)，其深度的平均值是 $O(\log N)$。不幸的是，正如图 4-12 中的例子所示，这个深度是可以大到 $N-1$ 的。

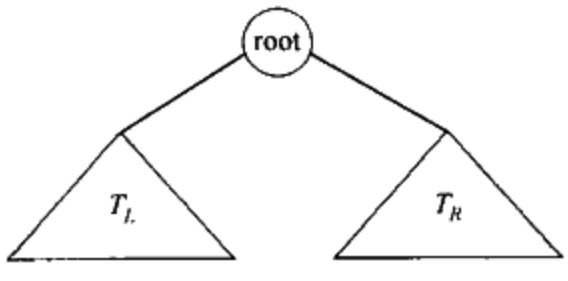

图 4-11 一般二叉树

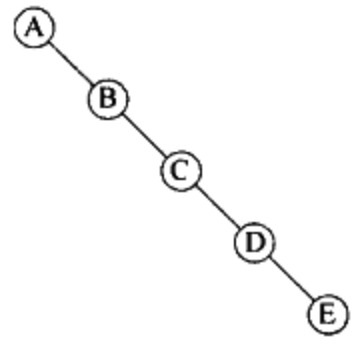

图 4-12 最坏情形的二叉树

4.2.1 实现

因为一个二叉树节点最多有两个子节点，所以我们可以保存直接链接到它们的链。树节点的声明在结构上类似于双链表的声明，在声明中，节点就是由 **element**(元素)的信息加上两个到其他节点的引用(left 和 right)组成的结构(见图 4-13)。

```
class BinaryNode
{
      // Friendly data; accessible by other package routines
  Object     element;      // The data in the node
  BinaryNode left;         // Left child
  BinaryNode right;        // Right child
}
```

图 4-13 二叉树节点类

我们习惯上在画链表时使用矩形框画出二叉树，但是，树一般画成圆圈并用一些直线连接起来，因为它们实际上就是图(graph)。当涉及到树时，我们也不明显地画出 null 链，因为具有 N 个节点的每一棵二叉树都将需要 $N+1$ 个 null 链。

二叉树有许多与搜索无关的重要应用。二叉树的主要用处之一是在编译器的设计领域，我们现在就来探索这个问题。

4.2.2 例子：表达式树

图 4-14 显示一个**表达式树**(expression tree)的例子。表达式树的树叶是**操作数**(operand)，如常数或变量名，而其他的节点为**操作符**(operator)。由于这里所有的操作都是二元的，因此这棵特定的树正好是二叉树，虽然这是最简单的情况，但是节点还是有可能含有多于两个的儿子。一个节点也有可能只有一个儿子，如具有**一目减算符**(unary minus operator)的情形。我们可以将通过递归计算左子树和右子树所得到的值应用在根处的运算符上而算出表达式树 T 的值。在本例中，左子树的值是 a+(b*c)，右子树的值是((d*e)+f)*g，因此整个树表示(a+(b*c))+(((d*e)+f)*g)。