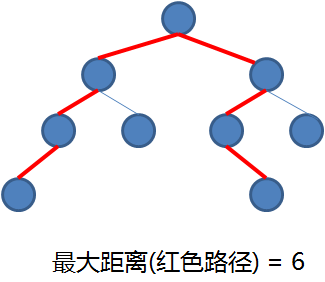
**问题定义**

如果我们把二叉树看成一个图，父子节点之间的连线看成是双向的，我们姑且定义"距离"为两节点之间边的个数。写一个程序求一棵二叉树中相距最远的两个节点之间的距离。



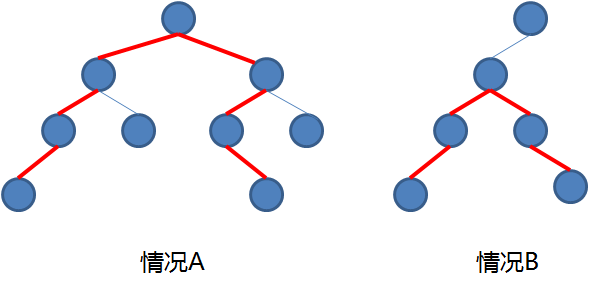
**书上的解法**

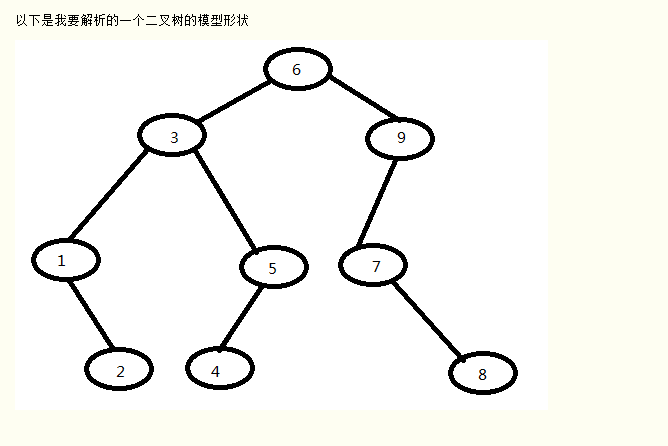
书中对这个问题的分析是很清楚的，我尝试用自己的方式简短覆述。

计算一个二叉树的最大距离有两个情况:

* 情况A: 路径经过左子树的最深节点，通过根节点，再到右子树的最深节点。
* 情况B: 路径不穿过根节点，而是左子树或右子树的最大距离路径，取其大者。

只需要计算这两个情况的路径距离，并取其大者，就是该二叉树的最大距离。





**package** facehandjava.tree;  
  
**public class** MaxDistance {  
 **public static** Node init() {*//注意必须逆序建立，先建立子节点，再逆序往上建立，因为非叶子结点会使用到下面的节点，而初始化是按顺序初始化的，不逆序建立会报错* Node J = **new** Node(8, **null**, **null**);  
 Node H = **new** Node(4, **null**, **null**);  
 Node G = **new** Node(2, **null**, **null**);  
 Node F = **new** Node(7, **null**, J);  
 Node E = **new** Node(5, H, **null**);  
 Node D = **new** Node(1, **null**, G);  
 Node C = **new** Node(9, F, **null**);  
 Node B = **new** Node(3, D, E);  
 Node A = **new** Node(6, B, C);  
 **return** A; *//返回根节点* }  
  
 **public static void** main(String[] args) {  
 Node root = MaxDistance.*init*();  
 System.***out***.println(**"树叶子节点的最大距离"**);  
 **int** L = *MaxDistance*(root);  
 System.***out***.println(L);  
 }  
  
 **private static** Integer *max* = 0;  
 **public static int** MaxDistance(Node node) {  
 **if** (node == **null**) {  
 **return** *max*;  
 }  
 **int** l = *DepthTree*(node.getLeftNode());  
 **int** r = *DepthTree*(node.getRightNode());  
 **int** d = l + r + 1;  
 *max* = *max* > d ? *max* : d;  
 *MaxDistance*(node.getLeftNode());  
 *MaxDistance*(node.getRightNode());  
 **return** *max*;  
 }  
  
 **public static int** DepthTree(Node node) {  
 **if** (node == **null**) {  
 **return** 0;  
 }  
 **int** l = *DepthTree*(node.getLeftNode());  
 **int** r = *DepthTree*(node.getRightNode());  
 **int** d = l > r ? l : r;  
 **return** d+1;  
 }  
  
}