**private void** redistributeLeafNode(Node correctLeafNode, Node correctLeafNode,Node parent)

**int** bestBranch = -1;  
**int** largestDis = -1;

*//*最大未满值**int** left = branchPos - 1;  
**int** right = branchPos + 1;

//搜索起始位置，中心点两侧

**int** leftDPCount = 0;  
**int** rightDPCount = 0;

*//记录*左右重分布区间的数据点数量

**while**( right < branchCount || left >= 0){  
 Node leftSibling = **null**;  
 Node rightSibling = **null**;  
 **/……./**跟踪更新leftDPCount和rightDPCount  
 **if**(bestBranch != -1){  
 **break**;*//尽快结束搜索* }  
 left--;  
 right++;  
}

//锁定最好分支的循环

**if**(bestBranch !=-1){  
 **int** redisBranchCount = (**int**)(Math.abs(branchPos - bestBranch) + 1);

//计算区间的分支距离  
 /……../最好分支在左侧，各分支数据点依次向左移动，元数据的更新 /……../最好分支在右侧，分支数据点依次向右移动，元数据的更新    
 parent.childrenBounds[2\*i] = currentSibling.distances[0];  
 parent.childrenBounds[2\*(i - 1) + 1] = nextSibling.distances[0];

//更新parent节点的界标数组  
 }  
}