**list.sort(cmp= list\_cmp)**

**def list\_cmp(node\_a,node\_b):**

**if node\_a 的len 与 node\_b 的len不等:**

**len小的放前面**

**else: 若相等**

**id大的放前面**

**for 节点A in （list）:**

**for 节点B in （节点A后面一个节点 到 list末）:**

**if A的左端点 ≥ B的左端点 and A的右端点 ≤ B的右端点:**

**则 A是B的孩子**

**Break；**

**算法证明：**

**证明树的正确性，即证明每个 父亲节点 与其 孩子节点 的对应完全正确；**

**即证明该算法中 两层循环及If条件内部 A一定是B 的孩子节点 成立。**

**反证法：**

**假设 A、B节点之间存在一个C节点，C节点 为 B节点 的孩子节点且为 A节点 的父亲节点。**

**则有关系 ：**

**C.L ≥ B.L && C.R ≤ B.R && C.id ≥ B.id ==> C.len ≤ B.len && C.id ≥ B.id**

**C.L ≤ A.L && C.R ≥ A.R && C.id ≤ A.id ==> C.len ≥ A.len && C.id ≤ A.id**

**列表经过排序后 其中所有元素均已保证 ：**

1. **len升序排列；② 若len 相等，则 id降序排列**

**若假设成立，则在该排列中存在相对位置为 A. . . . . . C . . . . . . B的相对排列 ，如果C存在的话，那么for循环从 A+1节点 扫描应该会先碰到 C节点，与命题相悖，因此不存在 C节点，即 A 一定是 B 的孩子节点。**