|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 节点p | 先序的后继 | 中序的后继 | 后序的后继 | 先序的前驱 | 中序的前驱 | 后序的前驱 |
| 先序线索树 | 若p->ltag==TRUE，  则p->lchild为后继；  若p->ltag==FALSE，则p->rchild为后继 | 若p->rtag==TRUE且p的右子树的ltag==TRUE  那么找到右子树的最左端点即为后继  如果p的右子树无左孩子，则p->rchild为后继  若p->rtag==FALSE  则需要遍历找到后继 | 需要遍历 | 若p->latg==FALSE时，则p->Lchild为前驱；  当p->ltag==TRUE时，需遍历才能求得前驱 | 若p->ltag==TRUE  则p的左子树的最右孩子为前驱  若p->ltag==FALSE  则需要遍历找到前驱 | 若p->rtag==TRUE  p->rchild为前驱  若p->rtag==FALSE  但是p->ltag==TRUE  则p->lchild为前驱  否则需要遍历找到前驱 |
| 中序线索树 | p\* | P$ | 若p->rtag==FALSE且p->rchild->rtag==FALSE  则p->rchild为后继  若p->rtag==FALSE && p  ->rchild->rtag==TRUE  则需要遍历找到后继  若p->rtag==TRUE  则p的右子树的最右节点为后继 | 若p->ltag==FALSE  则p->lchild->rchild是否为p，若是则找p->lchild的左子树的最右端点，即为前驱，若不是，则遍历p->rchild，每轮节点设为m，当m->lchild==p时，m为前驱  若p->ltag==TRUE,则找到p的左子树的最左节点然后重复上述过程 | $P | 若p->rtag==TRUE  则p->rchild为前驱  若p->rtag==FALSE  但p->ltag==TRUE  则p->lchild为前驱  否则令m=p->lchild  如果m是head节点，则p无前驱  如果不是head节点，m->lchild为前驱 |
| 后续线索树 | 若p->ltag==TRUE，  则p->lchild为后继；  若p->ltag==FALSE，则p->rchild为后继 | 若p->rtag==TRUE  则p的右子树的最左节点为后继  若p->rtag==FALSE  则需要遍历找到后继 | 若p->rtag==FALSE，则p->rchild为后继；  若p->rtag==TRUE，需遍历才能求得后继 | 需要遍历 | 若p->ltag==TRUE  则p的左子树的最右节点为前驱  若p->ltag==FALSE  则需要遍历找到前驱 | 若p->rtag==TRUE  则p->rchild为前驱  若p->rtag==TRUE  则p->lchild为前驱 |