1. insertion(s):

1)找到要insert的位置: 設一個pointer p，從root開始用while迴圈找，如 果目標比p大往 右，比p小則往左，順便用previous\_p紀錄目標的parent 2)num++

3) p = new node(s);

4)如果root!=NULL

if(s > previous\_p->number，即s的parent) { //insert s as right child

p -> right = previous\_p ->right;

p -> left = previous\_p;

previous\_p ->right = p;

}

else{ // insert as left child

p -> left = previous\_p -> left;

p -> right = previous\_p;

previous\_p ->left = p;

}

5)如果root==NULL(empty tree) : initialization，讓root=p

(左小孩為head，is\_threadl設為-1; 右小孩為tail，is\_threadr設為-1;)

2. deletion(s):

1)找出s的位置紀錄為p，順便找出其parent

2)num--

3)分為以下情況考慮

if(p是root)

else if(p的left是head)

else if(p的right是tail)

else if(p是parent的左小孩)

else if(p是parent的右小孩)

再分為以下情況考慮: p有兩個小孩、沒小孩、一左小孩、一右小孩。

以下以(p是parent的左小孩)為例說明:

if(兩個小孩) 從p的左樹中找到最大的node(biggest)來取代p

else if(沒小孩) parent->left=p->left;

else if(一右小孩) parent->left=p->right;

找出p的right subtree中最小的node (tmp)

tmp->left = p->left; //修改thread

else if(一左小孩) parent->left=p->left;

找出p的left subtree中最大的node (tmp)

tmp->right = p->right; //修改thread

4)delete p

3. inorder\_run():從head開始traverse直到tail

while(1)

if(p->is\_threadr==1) 直接移往right

else 移到right subtree中最小的node(is\_threadl=1)

if(p==tail) break

4. reverseorder\_run(): 從tail開始往回traverse直到head

while(1)

if(p->is\_threadl==1) 直接移往left

else 移到left subtree中最大的node(is\_threadr=1)

if(p==head) break