Wenlan Tian

COP 3530

Section 1087, MAEB 211

10/09/2014

Homework 3

"On my honor, I have neither given nor received unauthorized aid in doing this assignment.”

Summary of learning experience:

The easiest parts of the task is to figure out what the in-order and post-order traversals are, and the draw the figure.

The most difficult parts of the task: how to let the nodes being processed in two different orders with stack only (push and pop). The logic to achieve is really hard.

The assignment’s educational objectives: to let the students understand the algorithm of in-order and post-order traversal very well, and even how to implement using code.

How well I think I achieved them: I finished the pseudocode and prove the code is correct, but I do not have enough time to implement it in C++ because I have an exam. I think it’s very interesting to do this. I will do it later when I have time. From this homework, I finally understand the benefit of recursion, which is much easier to write.

Suggestions on how to improve the project: give more time and let us think more.

Wenlan Tian

COP 3530

Section 1087, MAEB 211

10/09/2014

Homework 3

Homework 3: iterative traversals

Figure out the algorithms for iterative inorder and postorder traversals, and prove they are correct by traversing with the same shape as in the two previous examples.

These are the pseudo-code for the algorithms for iterative in-order and post-order traversals. They also been proved by listing each step.

**1. In-order**

inorder(){

stack<Node\*> s;

Node\* n = root;

while(n!=NULL){

s.push(n);

n = n->left;

}

while(!s.empty()){

n = s.pop();

do\_something\_with(n);

if(n->right != NULL){

n = n -> right;

while(n!=NULL){

s.push(n);

n = n-> left;

}

}

}

}

Prof:

|  |  |
| --- | --- |
| Action | Stack contents |
| Push D | D |
| Push B | DB |
| Push A | DBA |
| Pop A | DB |
| Process A | DB |
| Pop B | D |
| Process B | D |
| Push C | DC |
| Pop C | D |
| Process C | D |
| Pop D |  |
| Process D |  |
| Push E | E |
| Pop E |  |
| Process E |  |
| Push F | F |
| Pop F |  |
| Process F |  |
| Done |  |

**2. Post-order**

postorder(){

stack<Node\*> s;

Node\* n = root;

while(!done){

while(n!=NULL){

if(n->right){

s.push(n->right);

}

s.push(n);

n = n -> left;

}

n = s.pop();

if (n->right && n->right == s.pop()){

s.pop();

s.push(n);

n = n-> right;

}else{

do\_something\_with(n);

n = NULL;

}

}

}

|  |  |
| --- | --- |
| Action | Stack contents |
| Push E | E |
| Push F | EF |
| Push B | EFB |
| Push C | EFBC |
| Push A | EFBCA |
| Pop A | EFBC |
| Process A | EFBC |
| Pop C | EFB |
| Pop B | EF |
| Push C | EFC |
| Push B | EFCB |
| Pop B | EFC |
| Process B | EFC |
| Pop C | EF |
| Process C | EF |
| Pop F | E |
| Pop E |  |
| Push F | F |
| Push D | FD |
| Push E | FDE |
| Pop E | FD |
| Pop D | F |
| Push E | FE |
| Push D | FED |
| Pop D | FE |
| Process D | FE |
| Pop E | F |
| Process E | F |
| Pop F |  |
| Done |  |