Andrew Plum

Cs 121

3/28/22

Week 11 Notes

* Binary trees
  + Expression trees
  + Search trees
* Error codes
  + Can be implemented to reduce segmentation faults
* Debugging
  + When was the last time it worked?
    - Ask yourself this question
* Stub-functioning
  + Making sure the pieces of code fit together and not necessarily function as intended
* Binary Search Tree:
  + For any node, all items in the left subtree are smaller, and , all items in the right subtree are larger
  + Code
    - Bool node::binary\_search(entry\_type goal){

If(data == goal){

return true;

}

If(data > goal){

If(left == NULL){

return false;

}

return left->binary\_search(goal);

}

If(data < goal){

If(right == NULL){

return false;

}

return right->binary\_search(goal);

}

}

* + - Bool node::binary\_search(entry\_type goal){

Bool result = false;

If(data == goal){

Result == true;

}

If(data > goal){

If(left == NULL){

Result == false;

} else {

Result == left->binary\_search(goal);

}

}

If(data < goal){

If(right == NULL){

Result == false;

} else {

Result == right->binary\_search(goal);

}

}

Return result;

}

* Binary Search Tree
  + Binary trees are like a link list
  + Use randomized order for data when inserting data in the binary tree
    - If not random, could binary tree will only use one side which will make it basically like a link list
  + Use insertion sort binary
    - Code
      * Errorcode node::binary\_insert(node n){

If(n.data == data){

Return duplicate; // errorcode

}

If(n.data < data){ // NULL

If(left == NULL){

Left == n;

Return success;

} else {

Return left->binary\_insert(n);

}

}

If(n.data > data){ // NULL

If(right == NULL){

Right == n;

Return success;

} else {

Return right->binary\_insert(n);

}

}

Return false;

}

* Binary Search Tree
* Code:
  + Errocode tree::insert( node n, node \*p){

If(n.data == data){

Return duplicate; // errorcode

}

* If(n.data < data){ // NULL
* If(left == NULL){
* Left == n;
* Return success;
* } else {
* Return left->binary\_insert(n);
* }
* }
* If(n.data > data){ // NULL
* If(right == NULL){
* Right == n;
* Return success;
* } else {
* Return right->binary\_insert(n);
* }
* }
* Return false;

}