CSCI 2270 Lecture Notes

2/18/19

* Node definition for BST
  + struct Node{

int key;

Node \*parent;

Node \*LC; // left child

Node \*RC; //right child

}

* BST ADT
  + private:

root

searchRecursive(node, value)

public:

init();

search(value)

insert(value)

disp() // ?

delete(value)

deleteTree() //destructor

* Insert Function
  + Given an existing tree, insert a node with a given key
    - Create new node and assign key value
    - Two pointers for traversal
      * temp = root
      * prev = null
    - Drill down to find next available empty spot
      * while(temp != nullptr){

prev = temp;

//check which way to traverse

if((n->key) < (temp)){

temp = temp->leftchild

}else{  
 temp = temp->rightchild

}

}

//established parent for our new node (prev)

* + - Add a new node to correct place
      * if prev == nullptrr
        + this would mean our tree was empty in the first place, and the while loop never ran
        + make the new node root
      * else if(n->key < prev->key)
        + this means new key is smaller than parent key, make the new node the left child

prev->leftChild = n;

n->parent = prev;

prev = nullptr;

* + - * else
        + key must be greater than or equal to parent key

prev->rightChild = n;

n->parent = prev;

prev = nullptr;

* Traversals
  + how do we decide about the order?
  + There are three conventions
    - Pre-Order
      * root, left, right
    - In-Order
      * left, root, right
    - Post-Order
      * left, right, root
  + How to implement
    - recursion
      * c++ allows a function to call another instance of itself
      * int foo(int x){

return foo(x); <-- this would result in stack overflow (BAD)

}

* + - * For any recursive algorithm, a base case needs to be defined
      * Once the base case is reached, no more recursive calls are made
        + the algorithm can end execution
* n! n factorial recursive function
  + int f(int n){

if(n <= 1){

return 1; //base case

}else{

return n\*f(n-1);

}

}

* + eg. f(4) = 4 \* f(3)

= 4 \* 3 \* f(2)

= 4 \* 3 \* 2 \* f(1)

= 4 \* 3 \* 2 \* 1

* In-Order Tree Traversal
  + function f(node)
  + drill down to left-most leaf (smallest value)
    - display this value
    - if(node.LC != nullptr){

f(node.LC)

}

* + Display key
    - cout << node->key << endl;
  + drill down to right most leaf.
    - if(node.RC !=nullptr){

f(node.RC)

}

* + Finish execution of current instance of f()