Advanced Programming

Cicchini, Taroni

December 27, 2021

1 BST

BST is a class that implement the concept of binary search tree, it's a template class that receive three templates: key, value, and the operation that define the relation between nodes

1.1 Attributes

The BST class has two attributes:

- root which is unique pointer to node needed as a starting point in the heap to the first node of the tree
- op that is an operation type of the template previously described

1.2 Methods

1.2.1 Constructors

Most of the constructor used are defaults, on the other hand a custom constructor is implemented in order to use a custom operator as an op attribute. The copy constructor is implemented in order to offer a deep copy of the tree: it copies the root node that invoke the function **copy_tree** which is a recursive function that copies every node of the tree.

1.2.2 Useful Methods

- **Move on**: it takes a pointer to a node and a direction, and it returns the pointer to a node based on the direction given.
- **Compare**: take two different keys and uses the comparing operation to return a direction in which one key should be located respect to the other.
- **Compare and move**: compare obtaining a direction and moves with move_on into that direction, it returns the pointer to the node and the direction we came from.

- **_insert a.k.a Insertino**: it's a pre insert function that uses move and compare through the tree and then insert a node in the empty place.
- **Balancing**: this function takes the keys and return a std::vector with a specific order that allows to obtain a balanced tree if insert is used

1.2.3 Begin/End

Begin and **end** are implemented to find the first node(the most left one in the BST) in terms of OP and the next to last which is set to nullptr.

1.2.4 Methods

- **Insert** it inserts a node into the tree, in case a node with the same key is already present it return a boolean false and an iterator to that node.
- **Emplace** it takes values and create a pair from that, then invoke insert for that pair
- **Clear** delete the whole tree
- **Find** this function uses compare and move to go through the tree and find the input key, if it doesn't find the key it returns the iterator end()
- **Balance** this function creates a new tree with the same nodes that is balanced. Balancing is invoked, and the tree is filled in the specific order defined by balancing.
- **Erase** this function delete a node than attaches the right branch to the parent, if it is present, then it attaches the left branch of the erased node through right branch.

1.2.5 Operator

- [ ]: given a key it returns the value, if the key is not in the tree it inset a node with the given key and a null value
- <<: this prints the tree keys and value, overloading the standard output stream

2 Node

It's a template class that receive an element of the template type and creates connection with other nodes

2.1 Attributes

- **Element**: it contains the data of the template type
- **Parent** It is a raw pointer to the parent of the node
- **L_next** It is a smart pointer(unique pointer) to the node on the left
- **R_next** It is a smart pointer(unique pointer) to the node on the right

2.2 Methods

2.2.1 Constructors

For the class node three custom constructors are implemented, one that receive an object of the type of element, and other two that receive not only this object but a pointer to node that initialize parent attribute.

3 Iterator

It is a template class used to cycle through the elements of the BST. It takes two templates, one is referred to the type of the object the iterator is pointing at, the other one to the element contained into that object.

3.1 Attributes

- **Current** is a pointer to that node of the tree.

3.2 Methods

3.2.1 Constructors

There is only one custom constructor that creates an iterator from a pointer of the node type.

3.2.2 Operator

- ***** it returns the element contained in the attribute current.
- $->$ it returns the pointer at the element contained in current.
- $++$ the increment operator go through the tree looking for the next element that is in the BST respect the order given by the operation op.
- $==$ **!=** those two operators compare the equality or the inequality of the attribute current.