```
1  // BST.h
2
3  #pragma once
4  #include <iostream>
5  #include <initializer_list>
6  using namespace std;
7
8
9
10 class BST {
11
12 private:
13     struct Node {
14         int val; // The value stored in the node
15         int size; // number of nodes in the subtree rooted here
16         Node *left; // points to the left subtree
17         Node *right; // points to the right subtree
18         Node(int v, Node *l = nullptr, Node *r = nullptr) : val(v), size(1),
19             left(l), right(r) {};
20     };
21
22     Node* root; //root node
23
24 public:
25     /*********************************************************************
26     BST()
27     Parameters: none
28     Complexity: O(1)
29
30     Default constructor. Makes an object of the BST class and sets the
31     root node to nullptr.
32     *********************************************************************/
33     BST() {
34         root = nullptr;
35     }
36
37     /*********************************************************************
38     BST()
39     Parameters: initializer_list<int>
40     Complexity: O(n)
41
42     User defined constructor. Makes an object of the BST class and inserts
43     every element in the list into the tree.
44     *********************************************************************/
45     BST(initializer_list<int> lst) {
46         for (auto i : lst)
47             insert(i);
48     }
49
50     /*********************************************************************
51     ~BST()
52     Parameters: none
```

```
53      Complexity: O(n)
54
55      Default destructor. Calls the clear function and passes it a pointer to
56      the root node. Deletes every node in the tree and removes dangling
57      pointers.
58      Complexity is O(n) because it has to delete every node in the tree.
59      **********************************************************************/
60      ~BST() {
61          clear(root);
62      }
63
64      /**********************************************************************
65      insert()
66      Parameters: int
67      Complexity: O(n)
68
69      Public insert method. Calls the private method and passes it a pointer to
70      the root node and the integer to be inserted.
71      Complexity is O(n) because the tree may not be balanced.
72      **********************************************************************/
73      void insert(int v) {
74          insertAux(root, v);
75      }
76
77      /**********************************************************************
78      remove()
79      Parameters: int
80      Complexity: O(n)
81
82      Public method to delete a node from the tree using recursion
83      It calls the private search function to make sure the node exists before
84      trying to delete it and therefore decrement all sizes along the path.
85      Then it call the private delete node method and passes it a pointer to a
86      node and the integer to be removed.
87      Complexity is O(n) because the tree may not be balanced.
88      **********************************************************************/
89      void remove(int v) {
90          if (searchAux(root, v) == true)
91              removeAux(root, v);
92      }
93
94      /**********************************************************************
95      inOrderPrint()
96      Parameters: ostream&
97      Complexity: O(n)
98
99      Public method to print the contents of a tree in order of ascending value.
100     Calls the private method and passes it a pointer to the root node
101     and the ostream type to use for output.
102     Complexity is O(n) because it has to go to every node in the tree and
103     print its value.
104     **********************************************************************/
```

```
105     void inOrderPrint(ostream& os)const {
106         if (root == nullptr)  // Tree has no data
107             os << "<empty>";
108         else
109             inOrderPrintAux(root, os);
110         os << endl;
111     }
112
113     /*****************************************************************
114     numNodes()
115     Parameters: none
116     Complexity: O(n)
117
118     Public method to return an int with the total number of nodes in the tree.
119     Calls the private method and passes it a pointer to the root node.
120     Complexity is O(n) because the method has to go to every node
121     in the tree.
122     *****************************************************************/
123     int numNodes() {
124         return numNodesAux(root);
125     }
126
127     /*****************************************************************
128     search()
129     Parameters: int
130     Complexity: O(n)
131
132     Public method to search for an int in a tree.
133     Returns a bool that says whether the int was found.
134     Calls the private method and passes it a pointer to the root node
135     and the int to search for.
136     Complexity is O(n) because the tree may not be balanced.
137     *****************************************************************/
138     bool search(int v) {
139         return searchAux(root, v);
140     }
141
142     /*****************************************************************
143     rank()
144     Parameters: int
145     Complexity: O(n)
146
147     Public method to return an integer's rank in the tree.
148     The rank is its position in the sorted tree. For example, the smallest
149     int in a tree would be rank 1. The largest would be equal to the number
150     of nodes in the tree. If the integer is not found in the tree, it
151     returns 0.
152     Calls the private method and passes it a pointer to the root node and
153     the int whose rank will be returned.
154     Complexity is O(n) because the method may have to go through every node
155     in the tree.
156     *****************************************************************/
```

```
157     int rank(int v) {
158         return rankAux(root, v);
159     }
160
161     /*****************************************************************
162     range()
163     Parameters: 2 ints
164     Complexity: O(n)
165
166     Public method to return an int with the range between two ints in a tree.
167     The range is the number of nodes in the tree with values that are
168     greater than or equal to the first argument (i),
169     and less than the second (j). If the arguments create an impossible range
170     or the tree is empty, it returns 0.
171     Calls the private method and passes it a pointer to the root node, and
172     two ints whose range will be returned.
173     Complexity is O(n) because the method may have to go through every node in
174     the tree.
175     *****************************************************************/
176     int range(int i, int j) {
177         return rangeAux(root, i, j);
178     }
179
180
181 private:
182     void clear(Node*& r);
183     void insertAux(Node*&, int);
184     void removeAux(Node*&, int);
185     void inOrderPrintAux(Node*, ostream&)const;
186     int numNodesAux(Node*&) const;
187     bool searchAux(Node*&, int)const;
188     int rankAux(Node*&, int, int = 1);
189     int rangeAux(Node*&, int, int);
190 };
```