

**Lab report**

|  |  |
| --- | --- |
| **Course**: | Class Libraries and Data Structures |
| **Semester**: | 1st semester of the academic year **2020-2021** |
| **Major**: | Software Engineering |
| **Class**: | 2019 |
| **Student Name**: | Fu Ruoxuan |
| **Student ID:** | 222019321062060 |
| **Teacher:** | ZHAO, Hengjun (赵恒军) |

**School of Computer and Information Science**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Name | | Binary Search Tree | | | |
| Date | | Jan，2021 | Type | | □Confirmatory  √ Design  □Comprehensive |
| 1. **Objective & Requirements**    1. Understand the concept and property of binary search tree    2. Get familiar with the insert, delete and find operations on binary search tree    3. Grasp the design of recursive or iterative algorithms about binary search tree | | | | | |
| 1. **Experimental environment (**platform and software**)**   Windows 7 (or higher versions) + Visual Studio 2010 (or higher versions) | | | | | |
| 1. **Experimental content and design** (Main Content, Procedure, Codes and Results)   Task 1  Generate a series of integers (of size n) randomly and insert them into an empty binary search tree, and compute the height of the tree.  Task1 Code   1. **template**<**typename** T> 2. **int** BinSearchTree<T>::height() **const** 3. { 4. **return** heightRec(root); 5. //please implement this 6. } 8. **template**<**typename** T> 9. **int** BinSearchTree<T>::heightRec(Node\* curr) **const** 10. { 11. **if** (curr == NULL) 12. **return** 0; 13. **else** 14. { 15. **int** leftHeight = 0, rightHeight = 0; 16. leftHeight = heightRec(curr->left); 17. rightHeight = heightRec(curr->right); 18. **return** (leftHeight > rightHeight) ? (leftHeight + 1) : (rightHeight + 1); 19. } 20. }   Task1 Result  文本  描述已自动生成 | | | | | |
| 1. **Result analysis and discussion**（Analysis of experimental results and summing up the harvest and the existing problems）   Analysis  The easiest way to get the tree height of a binary lookup tree is to recursively find the tree heights of the left and right subtrees separately, compare the heights of the left and right subtrees, and finally add one to the larger tree height to get the height of the whole binary lookup tree. Therefore, we need to define a new recursive function to find the tree height, and use the defined height() function to return the final result of this recursive function by passing in the address of the root node.  Harvest  A binary search tree is either an empty tree or a binary tree with the following properties: if its left subtree is not empty, the values of all nodes in the left subtree are smaller than the value of its root node; if its right subtree is not empty, the values of all nodes in the right subtree are larger than the value of its root node; its left and right subtrees are also binary sorted trees, respectively. Binary search tree as a classical data structure, which has the characteristics of fast insertion and deletion operations of a chain table and the advantages of fast lookup of arrays.  When performing an insert operation, the newly inserted operation node must be a newly added leaf node. When deleting, you should consider whether the node to be deleted is a leaf node or a node with one child node or a node with two children  Problem  None. | | | | | |
| Comments & Evaluation | Content & Design (A-E) | | |  | |
| Procedure & Codes (A-E) | | |  | |
| Results (A-E) | | |  | |
| Analysis & Discussion (A-E) | | |  | |
| Score (A-E):  Feedback comments: | | | | |