Báo cáo Assignment 04.01

Võ Lê Ngọc Thịnh

Ngày 11 tháng 4 năm 2025

Mục lục

1 Hàm chuyển cây BST sang Doubly Linked List tăng

```
class Node{
    public:
        int data;
        Node *left;     // Trong BST: left-node, trong DLL: prev-node
        Node *right;    // Trong BST: right-node, trong DLL: next-node

        Node (int _data = 0){
            this->data = _data;
            this->left = nullptr;
            this->right = nullptr;
        }
};

template<class T>
class BinarySearchTree{
    private:
        Node *root;

    public:
        BinarySearchTree (Node *_root = nullptr){
            this->root = _root;
        }

        Node* convertBSTtoDLL(Node* root){
            if (root == nullptr)    return nullptr;

            static Node* pHead = nullptr;
            static Node* pTail = nullptr;

            convertBSTtoDLL(root->left);

            if (pTail == nullptr){
                pHead = root;
            }
            else{
                pTail->right = root;
                root->left = pTail;
```

```
            }
            pTail = root;

            convertBSTtoDLL(root->right);

            return pHead;
        }
};
```

2 Hàm chuyển Doubly Linked List tăng sang cây BST

```
class Node{
    public:
        int data;
        Node *left;     // BST: left-node,  DLL: prev-node
        Node *right;    // BST: right-node, DLL: next-node

        Node (int _data = 0){
            this->data = _data;
            this->left = nullptr;
            this->right = nullptr;
        }
};

template<class T>
class BinarySearchTree{
    private:
        Node *root;

        Node *findMid(Node *pHead, Node *pTail){
            if (pHead == nullptr || pTail == nullptr)
                return nullptr;
            Node *slow = pHead;
            Node *fast = pHead;
            while (fast != pTail && fast->right != pTail){
                slow = slow->right;
                fast = fast->right->right;
            }
            return slow;
        }
```

```
30
31      public:
32          BinarySearchTree (Node *_root = nullptr){
33              this->root = _root;
34          }
35
36          Node *convertDLLToBST(Node *pHead, Node *pTail){
37              if (pHead == nullptr || pHead == pTail)
38                  return nullptr;
39
40              Node *mid = findMid(pHead, pTail);
41
42              mid->left = convertDLLToBST(pHead, mid);
43
44              mid->right = convertDLLToBST(mid->right, pTail);
45
46              return mid;
47          }
48
49          Node *convertDLLToBST(Node *pHead){
50              return convertDLLToBST(pHead, nullptr);
51          }
52  };
```