Report

Mssv:24120360

Tên: Trần Vĩnh Kiệt

BINARY TREE

1. createNode(int key)

Tạo một node mới có giá trị key, với con trái và phải là nullptr.

2. NLR(NODE* pRoot)

Duyệt tiền tự (Pre-order):
Node - Left - Right

3. LNR(NODE* pRoot)

Duyệt trung tự (In-order):
Left - Node - Right

4. LRN(NODE* pRoot)

Duyệt hậu tự (Post-order):
Left - Right - Node

5. LevelOrder(NODE* pRoot)

Duyệt theo từng tầng (Level-order/BFS). Trả về vector chứa các node theo từng tầng.

6. countNode(NODE* pRoot)

Đếm tổng số node trong cây.

7. sumNode(NODE* pRoot)

Tính tổng giá trị của tất cả các node.

8. heightNode(NODE* pRoot, int value)

Tìm độ cao của node có giá trị value, tính từ node đó đến lá xa nhất.
Nếu không tìm thấy, trả về -1.

9. Level(NODE* pRoot, NODE* p)

Tìm tầng (level) của node p trong cây, tính từ gốc là tầng 0.
Trả về -1 nếu không tìm thấy.

10. countLeaf(NODE* pRoot)

Đếm số lượng lá (node không có con trái và phải).

BST

1. Search(NODE* pRoot, int x)

Tìm node có giá trị x trong cây theo cách duyệt tuần tự cả trái và phải (không tận dụng đặc điểm BST).
Trả về con trỏ đến node nếu tìm thấy, hoặc nullptr nếu không.

2. Insert(NODE* &pRoot, int x)

Thêm giá trị x vào cây theo quy tắc BST:

- Nếu x < pRoot->key thì chèn vào cây con trái, ngược lại vào cây con phải.
- Nếu node rỗng thì tạo node mới.

3. Remove(NODE* &pRoot, int x)

Xoá node có giá trị x trong cây BST:

- Nếu node có 0 hoặc 1 con thì xoá trực tiếp.
- Nếu node có 2 con, tìm node nhỏ nhất trong cây con phải để thay thế và xoá node đó.

4. createTree(int a[], int n)

Tạo cây BST từ mảng a[] gồm n phần tử, chèn từng phần tử một.

5. removeTree(NODE* &pRoot)

Giải phóng bộ nhớ cây (xoá toàn bộ các node).

6. Height(NODE* pRoot)

Tính chiều cao cây:

- Nếu cây rỗng, chiều cao là 0.
- Chiều cao bằng 1 + chiều cao lớn nhất của cây con trái và phải.

7. countLess(NODE* pRoot, int x)

Đếm số node có giá trị nhỏ hơn x trong cây.

8. countGreater(NODE* pRoot, int x)

Đếm số node có giá trị lớn hơn x trong cây.

9. isBST(NODE* pRoot)

Kiểm tra cây có phải BST hợp lệ không (mọi node bên trái nhỏ hơn node cha, bên phải lớn hơn node cha).

10. isFullBST(NODE* pRoot)

Kiểm tra cây có phải là Full Binary Tree hay không (mọi node hoặc có 2 con hoặc không có con nào).

AVL

1. struct NODE

- Đại diện cho một node trong cây.
- key: giá trị của node.
- p_left, p_right: con trái, phải.
- height: chiều cao của node, dùng để kiểm tra cân bằng AVL.

2. createNode(int key)

Tạo một node mới với key và height = 1 (lá).

3. Insert(NODE* &pRoot, int x)

Chèn phần tử x vào cây:

- Chèn bình thường như BST.
- Sau khi chèn, gọi updateHeight() để cập nhật chiều cao.

4. Remove(NODE* &pRoot, int x)

Xoá node có giá trị x khỏi cây:

- Thực hiện xoá node trong BST như thường.
- Nếu node có 2 con, tìm node nhỏ nhất bên phải để thay thế.
- Cập nhật chiều cao sau khi xoá.

5. isAVL(NODE* pRoot)

Kiểm tra cây có phải AVL Tree không:

- Với mỗi node, kiểm tra chênh lệch chiều cao giữa trái và phải ≤ 1.
- Đệ quy kiểm tra tiếp các node con.