BÀI TẬP TUẦN 7

Môn học: Cấu Trúc Dữ Liệu và Giải Thuật.
GV hướng dẫn thực hành: Nguyễn Khánh Toàn (ktoan271199@gmail.com).
Nội dung chính:

- Cây nhị phân tìm kiếm.

Thời gian thực hiện: **1** tuần.
Bài tập được biên soạn lại và tham khảo từ tài liệu thực hành môn Cấu Trúc Dữ Liệu và Giải Thuật của quý Thầy Cô Trợ Giảng khóa trước của bộ môn Khoa Học Máy Tính và quý Thầy Cô của bộ môn Công Nghệ Tri Thức. Trân trọng cảm ơn quý Thầy Cô của các bộ môn.

1 Cây nhị phân tìm kiếm

Chúng ta định nghĩa một nút trong cây BST như sau (hình 1):

```
struct Node{
    int key;
    Node* left;
    Node* right;
};
```

Hình 1: Định nghĩa một nút trong cây tìm kiếm nhị phân (Binary Search Tree).

Các bạn cũng có thể sử dụng *class* để cài đặt cây tìm kiếm nhị phân nếu đã có kiến thức về lập trình hướng đối tượng (OOP) (khuyến khích).

Từ những kiến thức đã học ở phần lý thuyết, các bạn thực hiện cài đặt các thao tác sau của cây tìm kiếm nhị phân (lưu ý rằng các chữ ký hàm dưới đây chỉ là gợi ý, các bạn hoàn toàn có thể thay đổi, miễn hoàn thành đúng yêu cầu là được):

1. Khởi tạo một nút từ một giá trị cho trước.
    - **NODE*** createNode(**int** data);

2. Thêm một nút mới vào cây tìm kiếm nhị phân.
   - **void** Insert(**NODE*** &root, **int** x);
3. Duyệt cây theo thứ tự trước. Thứ tự duyệt này thường dùng để copy cây nhị phân tìm kiếm hoặc dùng trong biểu diễn tiền tố với biểu thức toán tử Ba Lan.
   - **void** NLR(**NODE*** root);
4. Duyệt cây theo thứ tự giữa. Trong cây tìm kiếm nhị phân, nếu ta duyệt theo thứ tự giữa, giá trị trả về sẽ là một mảng không giảm.
   - **void** LNR(**NODE*** root);
5. Duyệt cây theo thứ tự sau. Thứ tự duyệt này thường dùng để xóa cây nhị phân tìm kiếm hoặc dùng trong biểu diễn hậu tố với biểu thức toán tử Ba Lan.
   - **void** LRN(**NODE*** root);
6. (*) Duyệt cây theo tầng (level), ví dụ tầng 1 là nút gốc, tầng 2 là hai nút con của nút gốc, v.v. Kết quả trả về giá trị của cây theo từng tầng từ thấp đến cao, ở mỗi tầng in từ trái sang phải.
   - **void** levelOrder(**NODE*** root);
7. Tính chiều cao của cây.
   - **int** height(**NODE*** root);
8. Đếm số lượng nút của cây.
   - **int** countNode(**NODE*** root);
9. Tìm và trả về một nút với giá trị cho trước từ cây nhị phân tìm kiếm.
   - **NODE*** search(**NODE*** root, **int** x);
10. Xóa một nút với giá trị cho trước trong cây nhị phân tìm kiếm.
    - **void** remove(**NODE*** &root, **int** x);
11. Xóa toàn bộ cây nhị phân tìm kiếm.
    - **void** removeTree(**NODE*** &root);
12. Tính chiều cao tại một nút với giá trị cho trước trong cây tìm kiếm. Chẳng hạn nút với giá trị số 3 có chiều cao là 2. Ở nút lá các bạn sẽ cho ra chiều cao là 0. Trong trường hợp có nhiều nút cùng giá trị, xem xét phần tử xuất hiện đầu tiên; trường hợp không có trả về -1.
    - **int** heightNode(**NODE*** root, int x);

Các bạn cần lưu ý phân biệt giữa chiều cao của một nút và độ sâu của một nút:

Độ sâu của một nút: Chiều dài đường đi từ nút gốc tới nút đó.

Chiều cao của một nút: Chiều dài đường đi từ nút đó tới nút lá sâu nhất của nó.

Chiều cao của một cây: Chiều cao của nút gốc.

Tham khảo

13. Kiểm tra xem một cây nhị phân cho trước có phải là cây nhị phân tìm kiếm hay không?

    - **bool** isBST(**NODE*** root);

Hết