Data Structures and Algorithms

HUS HKI, 23 - 24

Assignment 5

Lecturer: Nguyễn Thị Hồng Minh
Trần Bá Tuấn - Đặng Trung Du

§ Tree, Binary Tree and Application §

Phần 1: Mục tiêu

- Nắm được các khái niệm liên quan đến cây tổng quát, cây nhị phân.
- Giải thích được các cách biểu diễn cây và có khả năng vẽ cây từ theo một yêu cầu cụ thể.
- Giải thích và triển khai được thuật toán chuyển đổi cây tổng quát thành cây nhị phân và ngược lại.
- Giải thích và triển khai được cách biểu diễn cây nhị phân trên máy tính.
- Tìm hiểu một số thuật toán và triển khai được thuật toán cho cây tổng quát và cây nhị phân.

Phần 2: Tài liệu đọc thêm

- Cây - https://www.geeksforgeeks.org/introduction-to-tree-data-structure-and-algorithm-tutorials/
- Cây nhị phân - https://www.geeksforgeeks.org/binary-tree-data-structure/?ref=ghm

Phần 3: Thực hành

(1) Bài tập

CHÚ Ý

1. Lựa chọn các gói bài tập để thực hiện:
   - Combo 1: Bài 1, Bài 2 (cơ bản)
   - Combo 2: Bài 2, Bài 3 (nâng cao)
   - Combo 3: Bài 4 (nâng cao)
   - Combo 4: Bài 5 (nâng cao)
2. Trong bài nộp có file .doc hoặc .txt thuyết minh về gói bài tập thực hiện và những nội dung cần giải thích về bài làm: thuật toán được chọn, tài liệu tham khảo, định dạng input, yêu cầu hệ thống...

Bài tập 1. ***Tạo giao diện BinaryTreeInterface như sau:***

```
public interface BinaryTreeInterface<T> {
    T root();
    int size(); // number of node in tree
    boolean isEmpty();
    int numChildren(T p); // nmber of children of element p;
```

```
    T parent(T p);  //return parent of p
    T left(T p);  //return left child of p
    T right(T p);   //return right child of p
    T sibling(T p); //return sibling of p
}
```

a) ***Xây dựng kiểu dữ liệu BinaryTree sử dụng mảng, cài đặt giao diện BinaryTreeInterface đã xây dựng ở trên với gợi ý như sau:***

```
public class ArrayBinaryTree<E,T> implements BinaryTreeInterface<T> {
    private E [] array;
    private int n = 0;
    private int defaultsize = 100;

    public ArrayBinaryTree(){
        array = (E[]) new Object[defaultsize];
    }
    //update methods
    public void setRoot(E element) {
        // Set element to root of an empty tree (at index 1)
    }
    public void setLeft(int p, E element) {
        // Set left child of p (at index 2p)
    }
    public void setRight(int p, E element) {
        // Set right child of p (at index 2p+1)
    }
}
```

b) ***Xây dựng cấu trúc dữ liệu BinaryTree bằng cách sử dụng danh sách liên kết, cài đặt giao diện BinaryTreeInterface đã xây dựng ở trên với gợi ý như sau:***

```
public class LinkedBinaryTree<E,T> implements BinaryTreeInterface<T> {

    protected static class Node<E> {
        private E element;    // an element stored at this node
        private Node<E> parent;   // a reference to the parent node (if any)
        private Node<E> left;   // a reference to the left child
        private Node<E> right;  // a reference to the right child

        // Constructs a node with the given element and neighbors.
        public Node(E e, Node<E> above, Node<E> leftChild, Node<E> rightChild){
            // Todo
        }
    }
    //update methods
    public Node<E> addRoot(E element) {
        // Add element to root of an empty tree
    }

    public Node<E> addLeft(Node p, E element) {
        // Add element to left child node of p if empty
    }

    public Node<E> addRight(Node p, E element) {
        // Add element to right child node of p if empty
    }


```

```
    public void set(Node p, E element) {
        // set element to node p
    }
}
```

Lưu ý: T là position trong cây, nếu cài đặt bằng Array thì T là kiểu int, nếu cài đặt bằng Linked thì T là kiểu Node<E>, E là kiểu generic giá trị của phần tử (data) trong cây.

c) ***Tạo một cây nhị phân sử dụng hai cấu trúc dữ liệu trên. Viết hàm in và in ra màn hình và file output.txt cây nhị phân vừa tạo theo chiều ngang. Ví dụ:***

```
        7
    3
        2
1
        6
    5
        8
```

Bài tập 2. *Biểu thức số học (Arithmetic Expression) có thể biểu diễn theo một dạng khác nhau tùy thuộc vào vị trí tương đối của toán tử so với các toán hạng. Phổ biến bao gồm các dạng biểu thức: trung tố (infix), hậu tốt (postfix), tiền tố (prefix). Ví dụ:*

- *Biểu thức trung tố: (6/3 + 2) * (7 - 4) (toán hạng đứng hai bên toán tử).*
- *Biểu thức hậu tố: 6 3 / 2 + 7 4 - * (toán hạng đứng sau toán tử)*
- *Biểu thức tiền tố: * + / 6 3 2 - 7 4 (toán hạng đứng trước toán tử)*

a) ***Xây dựng lớp ExpressionTree mở rộng lớp LinkedBinaryTree với việc bổ sung các phương thức duyệt cây theo gợi ý như sau:***

```
public class ExpressionTree<E> extends LinkedBinaryTree{

    public void preorderPrint(Node<E> p) {
        //pre-order traversal of tree with root p
    }

    public void postorderPrint(Node<E> p) {
        //post-order traversal of tree with root p
    }

    public void inorderPrint(Node<E> p) {
        //in-order traversal of tree with root p
    }
}
```

b) ***Sử dụng các phương thức để in ra các dạng biểu diễn của biểu thức (tiền tố, trung tố, hậu tố) được biểu diễn bằng một cây nhị phân biểu thức cho trước.***

- *Input: Cây nhị phân biểu thức*
- *Output: Các dạng biểu diễn của biểu thức*

c) ***Sử dụng các phương thức để tính toán giá trị của biểu thức biểu diễn bởi một cây nhị phân cho trước.***

- *Input: Cây nhị phân biểu thức*
- *Output: Giá trị của biểu thức*

Bài tập 3. ***Viết chương trình dựng cây nhị phân biểu diễn biểu thức số học được cho dưới dạng chuỗi.***

Giải thích thêm về các dạng biểu diễn biểu thức số học

Biểu thức số học (Arithmetic Expression) có thể biểu diễn theo một dạng khác nhau tùy thuộc vào vị trí tương đối của toán tử so với các toán hạng. Phổ biến bao gồm các dạng biểu thức: trung tố (infix), hậu tốt (postfix), tiền tố (prefix). Ví dụ:

- *Biểu thức trung tố: (6/2 + 3) * (7 - 4) (toán hạng đứng hai bên toán tử)*
- *Biểu thức hậu tố: 6 2 / 3 + 7 4 - * (toán hạng đứng sau toán tử)*
- *Biểu thức tiền tố: * + / 6 2 3 - 7 4 (toán hạng đứng trước toán tử)*

Độ ưu tiên của các toán tử như sau:

- *Toán tử nhân (*) = toán tử chia (/)*
- *Toán tử (+) = toán tử trừ (-)*
- *Nhân, chia ưu tiên cao hơn cộng, trừ*

Bài toán dựng cây nhị phân biểu diễn biểu thức số học được cho dưới dạng chuỗi được phân tích như sau:

Input: *biểu thức dạng trung tố đã được tách thành các tokens. Mỗi token là thành phần của biểu thức gồm dấu ngoặc, toán hạng, toán tử.*

*Ví dụ input cho biểu thức (6/3 + 2) * (7 - 4) là*

```
String[] tokens = {"(","6", "/", "3","+","2",")", "*", "(", "7", "-", "4",")"}
```

Ouput: *cây nhị phân biểu diễn biểu thức (sử dụng cấu trúc liên kết).*

Algorithm: *(1 trong 2 cách)*

- *Dựng cây trực tiếp từ các token input.*
- *Chuyển biểu thức từ dạng trung tố sang hậu tố hoặc tiền tố sử dụng Stack, rồi dựng cây.*

MỘT SỐ THUẬT TOÁN HỖ TRỢ

a) ***Thuật toán chuyển từ dạng trung tố sang dạng hậu tố***

Đọc từng token trong biểu thức infix từ ***trái qua phải****, với mỗi token ta thực hiện các bước sau:*

- Nếu là toán hạng: cho ra output.

- Nếu là dấu mở ngoặc "(": cho vào stack.

- Nếu là dấu đóng ngoặc ")": lấy các toán tử trong stack ra và cho vào output cho đến khi gặp dấu mở ngoặc "(". Dấu mở ngoặc cũng phải được đưa ra khỏi stack.

- Nếu là toán tử:

- *Chừng nào ở đỉnh stack là toán tử và toán tử đó có độ ưu tiên* ***lớn hơn hoặc bằng*** *toán tử hiện tại thì lấy toán tử đó ra khỏi stack và cho ra output.*
- *Đưa toán tử hiện tại vào stack.*

Sau khi duyệt hết biểu thức infix, nếu trong stack còn phần tử thì lấy các token trong đó ra và cho lần lượt vào output. Cuối cùng là đảo ngược biểu thức một lần nữa và ta sẽ thu được kết quả.

b) ***Thuật toán chuyển từ dạng trung tố sang dạng tiền tố***

Đọc từng token trong biểu thức infix từ ***phải qua trái****, với mỗi token ta thực hiện các bước sau:*

- Nếu là toán hạng: cho ra output.

- Nếu là dấu đóng ngoặc ")": cho vào stack.

- Nếu là dấu mở ngoặc "(": lấy các toán tử trong stack ra và cho vào output cho đến khi gặp dấu đóng ngoặc ")". Dấu đóng ngoặc cũng phải được đưa ra khỏi stack.

- Nếu là toán tử:

- *Chừng nào ở đỉnh stack là toán tử và toán tử đó có độ ưu tiên* ***lớn hơn hoặc bằng*** *toán tử hiện tại thì lấy toán tử đó ra khỏi stack và cho ra output.*

- *Đưa toán tử hiện tại vào stack.*

Sau khi duyệt hết biểu thức infix, nếu trong stack còn phần tử thì lấy các token trong đó ra và cho lần lượt vào output. Cuối cùng là đảo ngược biểu thức một lần nữa và ta sẽ thu được kết quả.

c) ***Thuật toán dựng cây nhị phân biểu thức***

Từ biểu thức xâu kí tự đầu vào, dựng cây nhị phân biểu diễn biểu thức theo các bước sau:

1. *Chuyển biểu thức từ dạng trung tố sang hậu tố.*
2. *Xây dựng cây nhị phân từ dạng hậu tố:*
   Lặp qua từng token trong chuỗi hậu tố
   - Tạo một đối tượng BinaryTreeNode nhãn của node là token này
   - Nếu là toán hạng: Push node vào stack
   - Nếu là toán tử:
   - *Pop một toán hạng ra khỏi stack và đặt làm RightChild của node.*
   - *Pop toán hạng kế tiếp ra khỏi stack và đặt làm LeftChild của node.*
   - *Push node vào stack.*

   Sau khi vòng lặp kết thúc, phần tử cuối cùng còn lại trong stack là node gốc của cây biểu thức.

Bài tập 4. *Xem yêu cầu project P-8.68 sách M. Goodrich, trang 358 và thực hiện project.*

Bài tập 5. *Xem yêu cầu project P-8.69 sách M. Goodrich, trang 358 và thực hiện project.*

(2) Bài tập thêm

MỤC TIÊU

- Giúp sinh viên có thêm nhiều bài tập để cải thiện kỹ năng lập trình của bản thân.
- Nắm rõ kiến thức, thành thạo cài đặt về cây tổng quát, cây nhị phân.

Luyện tập 1. *Sinh viên tự chọn 2 bài trong Bài tập từ 52 đến 69.*

https://codelearn.io/learning/data-structure-and-algorithms

A. MỘT SỐ BÀI TẬP VỀ TÍNH CHẤT CỦA CÂY NHỊ PHÂN

Luyện tập 2. *Tính chiều cao của cây nhị phân - Height of Binary Tree*

- *Nguồn: https://www.geeksforgeeks.org/height-and-depth-of-a-node-in-a-binary-tree/*

Luyện tập 3. *Đếm số lá trên cây nhị phân - Count Leaves in Binary Tree*

- *Nguồn: https://www.geeksforgeeks.org/write-a-c-program-to-get-count-of-leaf-nodes-in-a-binary-tree/*
- *Gợi ý: Duyệt, kiểm tra 1 nút có là lá và đưa vào kết quả.*

Luyện tập 4. *Tổng các nút lá - https://leetcode.com/problems/sum-of-left-leaves/*

- *Gợi ý: Duyệt, kiểm tra 1 nút có là lá và đưa vào kết quả.*

Luyện tập 5. *Kiểm tra các lá có cùng 1 giá trị - https://leetcode.com/problems/univalued-binary-tree/*

- *Gợi ý: Duyệt, kiểm tra 1 nút có là lá và đưa vào kết quả.*

Luyện tập 6. *Số phần tử của cây nhị phân - Size of Binary Tree*

- *Nguồn: https://www.geeksforgeeks.org/write-a-c-program-to-calculate-size-of-a-tree/*

Luyện tập 7. *Đếm số phần tử không phải lá của cây nhị phân - Count Non-Leaf Nodes in Tree*

- *Nguồn: https://www.geeksforgeeks.org/print-all-internal-nodes-of-a-binary-tree/*

Luyện tập 8. *Độ sâu tối thiểu của cây nhị phân - Minimum Depth of a Binary Tree*

- *Mô tả: Là nút lá có mức thấp nhất.*
- *Gợi ý: Triển khai thuật toán duyệt cây và trong khi duyệt cây kiểm tra một nút có là nút lá và sau đó dựa vào mức của nút lá đó cập nhật kết quả.*

Luyện tập 9. *Liệt kê các phần tử mà không có họ hàng - Print all nodes that don't have sibling*

- *Nguồn: https://www.geeksforgeeks.org/print-nodes-dont-sibling-binary-tree/*

Luyện tập 10. *Kiểm tra hai cây có trùng nhau - Determine if Two Trees are Identical*

B. MỘT SỐ BÀI TẬP VỀ CÁCH DUYỆT CÂY

Luyện tập 11. *Duyệt cây theo thứ tự giữa - Inorder Traversal*

- *Nguồn: https://www.geeksforgeeks.org/tree-traversals-inorder-preorder-and-postorder/*

Luyện tập 12. *Duyệt cây theo thứ tự trước - Preorder Traversal*

Luyện tập 13. *Duyệt cây theo thứ tự sau - Postorder Traversal*

Luyện tập 14. *Duyệt cây theo mức của cây - Level Order Traversal*

- *Nguồn: https://www.geeksforgeeks.org/level-order-tree-traversal/*
- *Trao đổi thêm: Nếu duyệt bằng cách sử dụng hàng đợi thì độ phức tạp là $O(n)$. Còn nếu duyệt bằng đệ quy thì độ phức tạp $O(n^2)$.*

Luyện tập 15. *Duyệt cây theo kiểu zigzag - ZigZag Tree Traversal*

Luyện tập 16. *Duyệt theo đường chéo - Diagonal Traversal of Binary Tree*

- *Nguồn: https://www.geeksforgeeks.org/diagonal-traversal-of-binary-tree/*

Luyện tập 17. *Duyệt theo biên của cây - Boundary Traversal of binary tree*

- *Nguồn: https://www.geeksforgeeks.org/boundary-traversal-of-binary-tree/*

Luyện tập 18. *Duyệt cây sử dụng cách nhìn từ trái - Left View of Binary Tree*

Luyện tập 19. *Duyệt cây sử dụng cách nhìn từ trên xuống -Top View of Binary Tree*

MỘT SỐ BÀI TẬP VỀ CÁCH TẠO (XÂY DỰNG) CÂY

Luyện tập 20. *Tạo (Xây dựng) cây nhị phân từ danh sách liên kết - Make Binary Tree From Linked List*

- *Nguồn: https://www.geeksforgeeks.org/given-linked-list-representation-of-complete-tree-convert-it-to-linked-representation/*

Luyện tập 21. *Tạo cây nhị phân từ kết quả duyệt cây theo mức - https://www.geeksforgeeks.org/construct-complete-binary-tree-given-array/*

- *Nguồn: https://www.geeksforgeeks.org/construct-complete-binary-tree-given-array/*

(3) Quy cách nộp bài

- Mỗi sinh viên hoàn thành bài tập trong package có tên Hw5_<idSinhvien>_<Hovaten>
- Trong đó, <idSinhvien> là mã sinh viên.
- Sinh viên nộp bài làm trên tài khoản của mình bao gồm:
  1. File nén .zip của thư mục chứa package (Hw5_<idSinhvien>_<Hovaten>.zip)
  2. Tất cả các file nguồn *.java.
- Khi hết hạn nộp bài, các bài làm sẽ công bố để thực hiện chấm chéo bài tập.

- Sinh viên không nộp bài sẽ nhận điểm 0 bài tập tuần.
- Sinh viên CÓ GIAN LẬN trong nộp bài tập sẽ bị ĐÌNH CHỈ môn học (điểm 0 cho tất cả các điểm thành phần).

Lưu ý:

- **Phần Bài tập, sinh viên tự lựa chọn gói bài tập để thực hiện. Dựa vào gói bài tập sẽ tiến hành đánh giá khả năng của sinh viên. Khuyến khích sinh viên chọn mức độ nâng cao nhưng cần phải hiểu rõ các bài tập ở gói cơ bản.**
- **Trong phần Bài tập thêm, sinh viên thực hiện yêu cầu của Luyện tập 1, tự chọn trong các mục A, B, C với mỗi mục ít nhất 1 bài luyện tập. Khuyến khích sinh viên luyện tập càng nhiều càng tốt để nâng cao khả năng lập trình.**