ĐẠI HỌC UEH

TRƯỜNG CÔNG NGHỆ VÀ THIẾT KẾ

KHOA CÔNG NGHỆ THÔNG TIN KINH DOANH

ĐỒ ÁN MÔN HỌC

ĐỀ TÀI:

CÂY TÌM KIẾM NHỊ PHÂN VÀ ỨNG DỤNG TRONG QUẢN LÝ SẢN PHẨM

Học Phần: Cấu Trúc Dữ Liệu & Giải Thuật

Giảng viên: TS. Đặng Ngọc Hoàng Thành

Danh Sách Nhóm:

Nguyễn Đỗ Đức Hào	31181024020
Nguyễn Lê Nguyên	31181024110
Hoàng Thị Đan Phương	31181023510
Nguyễn Quỳnh Khánh Hà	31201020239
Trần Phạm Minh Việt	31181023058

TP. Hồ Chí Minh, ngày 19 tháng 12 năm 2021

MỤC LỤC

MỤC LỤC HÌNH ẢNH

CHƯƠNG 1. CÂY NHỊ PHÂN TÌM KIẾM

1.1. Các khái niệm liên quan

Cây (tree) là một dạng cấu trúc dữ liệu bao gồm: các đỉnh (Node) và cạnh (Edge). Các đỉnh chứa thông tin và được kết nối bởi các cạnh.

Tập thứ tự các cạnh sẽ tạo thành một đường đi (path).

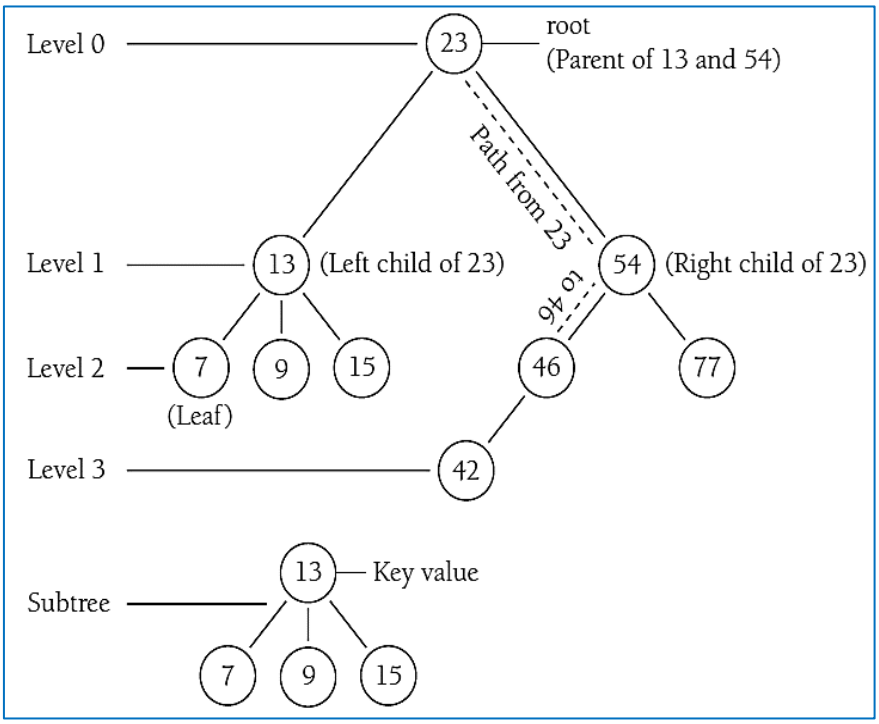
Đỉnh ở trên cùng được gọi là gốc (root).

Các đỉnh còn được gọi là nút con (child).

Đỉnh không có bất kì nút con nào được gọi là lá (leaf).

Các đỉnh được phân thành các mức khác nhau (level 0, level 1, v.v.)

Một cây mà mỗi đỉnh có không quá 2 nút con được gọi là cây nhị phân (Binary tree).



1. Ví dụ mô tả cây nhị phân

Cây nhị phân

- Mỗi một đỉnh có không quá hai nút con: nút trái (left) và nút phải (right).

Cây tìm kiếm nhị phân (BST-Binary Search Tree)

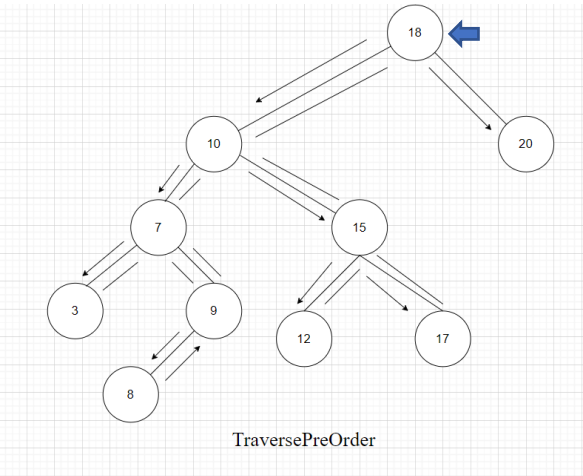
- Giá trị của tất cả các nút ở cây con bên trái <= giá trị của nút gốc.
- Giá trị của tất cả các nút ở cây con bên phải > giá trị của nút gốc.
- Tất cả các cây con (bao gồm bên trái và phải) đều phải đảm bảo 2 tính chất trên.

Duyệt cây

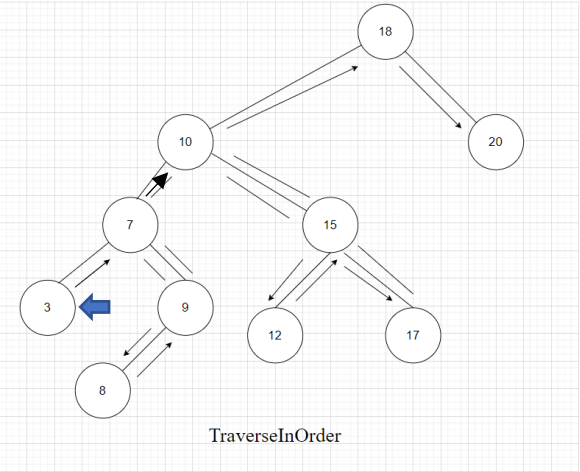
Duyệt cây là một tiến trình để truy cập tất cả các nút của một cây và cũng có thể in các giá trị của các nút này. Bởi vì tất cả các nút được kết nối thông qua các cạnh (hoặc các link), nên chúng ta luôn luôn bắt đầu truy cập từ nút gốc. Do đó, chúng ta không thể truy cập ngẫu nhiên bất kỳ nút nào trong cây. Có ba phương thức mà chúng ta có thể sử dụng để duyệt một cây:

- ✓ Duyệt tiền thứ tự (TraversePreOrder)
- ✓ Duyệt trung thứ tự (TraverseInOrder)
- ✓ Duyệt hậu thứ tự (TraversePostOrder)

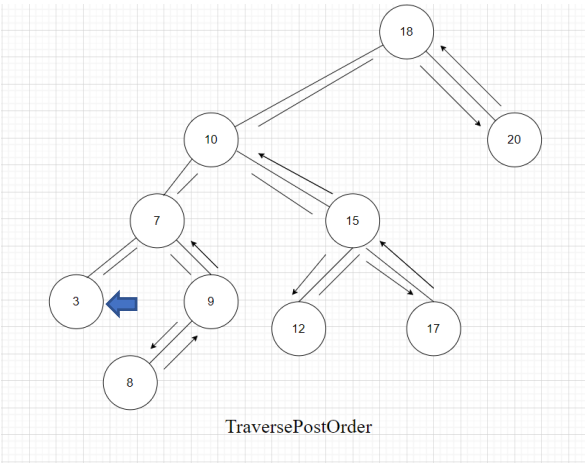
Ví dụ:



TraversePreOrder



TraverseInOrder



2. Ví dụ duyệt cây nhị phân tìm kiếm

1.2. Cấu trúc và cài đặt cây nhị phân

- Mỗi nút gồm có dữ liệu (data), nút trái (left) và nút phải (right).

- Mỗi BST có một nút gốc (root).

```
public class Node
    {
                public Product product;
                public Node left;
                public Node right;

                public  Node(Product product)
                {
                        this.product = product;
                        this.left = null;
                        this.right = null;
                //Tiếp tục các hàm
```

```
public class BSTree
        {
                private int size;
                public Node root;
  //Tiếp tục cho các hàm……
```

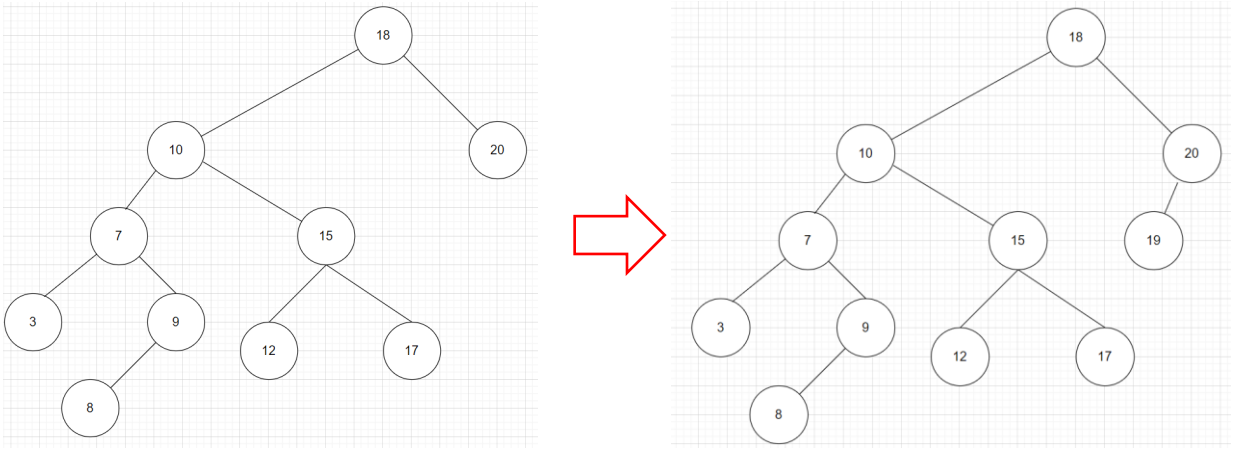
3. Cấu trúc node

1.3. Các thuật toán trên cây BST

a) Thêm nút vào BST

```
public  void insert(Node newNode)
        {
                root = insert(root, newNode);
        }

        private Node insert(Node node, Node newNode)
        {
                if (node == null)
                {
                        size++;
                        return newNode;
                }

                if (newNode.lessThanOrEqual(node.product))
                {
                        node.left = insert(node.left, newNode);
                }
                else
                {
                        node.right = insert(node.right, newNode);
                }
                return node;
        }
```

Ví dụ: Chèn số 19 vào



4. Ví dụ chèn nút vào BST

b) Thăm nút BST

TraversePreOrder (Tiền thứ tự)

```
public void TraversePreOrder()
{
	Console.Write("\n\n----------------------------------------
TraversePreOrder-----------------------");
	TraversePreOrder(root);
}

private void TraversePreOrder(Node node)
{
	if (node == null) return;
	node.Data();
	TraversePreOrder(node.left);
	TraversePreOrder(node.right);
}
```

TraverseInOrder (Trung thứ tự)

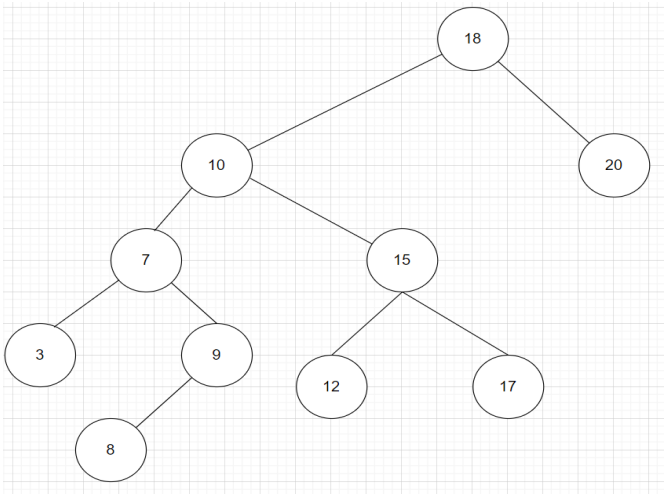
```
public void TraversetInOrder()
{
	Console.Write("\n\n----------------------------------------
TraverseInOrder-------------------------------------");
	TraverseInOrder(root);
}

private void TraverseInOrder(Node node)
{
	if (node == null) return;
	TraverseInOrder(node.left);
	node.Data();
	TraverseInOrder(node.right);
}
```

TraversePostOrder (Hậu thứ tự)

```
public void TraversePostOrder()
{
	Console.Write("\n\n----------------------------------------
TraversePostOrder-----------------------------------");
	TraversePostOrder(root);
}

private void TraversePostOrder(Node node)
{
	if (node == null)
	{
		return;
	}
	TraversePostOrder(node.left);
	TraversePostOrder(node.right);
	node.Data();
```

```
}
```

Ví dụ:



5. Ví dụ thăm nút

TraversePreOrder: 18, 10, 7, 3, 8, 9, 15, 12, 17, 20

TraverseInOrder: 3, 7, 8, 9, 10, 12, 15, 17, 18, 20

TraversePostOrder: 3, 8, 9, 7, 12, 17, 15, 10, 20, 18

c) Tìm nút BST

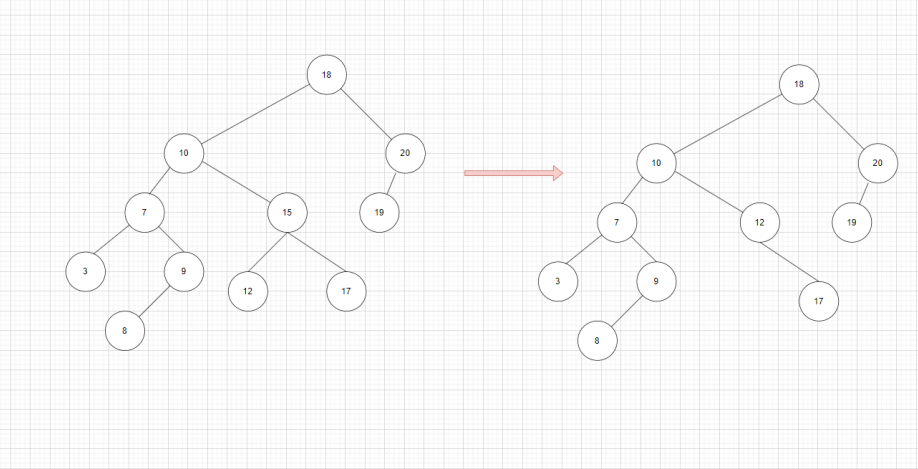
```
public Node search(int productId)
		{
			return search(this.root, productId);
		}

		private Node search(Node node, int productId)
		{
			if (node == null)    return null;


			if (node.hasproduct(productId))
			{
				return node;
			}
			if (node.lessThan(productId))
			{
				return search(node.right, productId);
			}
			else
			{
				return search(node.left, productId);
```

```
                }
```

d) Xóa nút BST

```
public void delete(int productId)
            {
                root = delete(root, productId);
            }

            public Node delete(Node node, int productId)
            {
                if (node == null) return null;


                if (node.hasproduct(productId))
                {
                    size--;
                    if (node.left == null)
                    {
                        return node.right;
                    }
                    Node maxNode = node.left;
                    Node preNode = maxNode;
                    while (maxNode.right != null)
                    {
                        preNode = maxNode;
                        maxNode = maxNode.right;
                    }

                    maxNode.right = node.right;
                    if (maxNode != node.left)
                    {
                        preNode.right = maxNode.left;
                        maxNode.left = node.left;

                    }
                    return maxNode;
                }
                if (node.lessThan(productId))
                {
                    node.right = delete(node.right, productId);
                }
                else
                {
                    node.left = delete(node.left, productId);
                }
                return node;
            }
```

Ví dụ: Xóa nốt 15



6. Ví dụ xóa nút

e) Program-Main

```
static void Main(string[] args)
        {

            Console.OutputEncoding = System.Text.Encoding.UTF8;
            Console.Clear();


            Console.Write("Cây nhị phân tìm kiếm với sản phẩm \n");
            BSTree tree = new BSTree();
            tree.insert(new Node(new Product(18, "Hub 4.0", 180, 1, "Chậm")));
            tree.insert(new Node(new Product(20, "Ram DDR4", 980, 5, "tốc độ xử lí
nhanh")));
            tree.insert(new Node(new Product(10, "SSD 256gb", 1000, 5, "Nhanh, rẻ")));
            tree.insert(new Node(new Product(15, "HDD 1T", 560, 4, "Ổn định")));
            tree.insert(new Node(new Product(12, "VGA NVIDIA", 3800, 4, "Ngon")));
            tree.insert(new Node(new Product(17, "USB 16gb", 250, 3, "Xịn xò")));
            tree.insert(new Node(new Product(7, "Laptop Dell Vostro 5490", 18000, 4, "Máy
đẹp")));
            tree.insert(new Node(new Product(8, "Surface Pro 3", 15000, 3, "máy yếu")));
            tree.insert(new Node(new Product(3, "Logitech Mouse", 480, 3, "Giá ổn")));

            Console.WriteLine("| 1 - Insert   |\n| 2 - Traverse |\n| 3 - Find     |\n| 4 -
Remove   |\n| 0 - Exit     |");
            for (; ; )

//Tiếp tục các hàm
```

CHƯƠNG 2. PHÂN TÍCH VÀ THIẾT KẾ LỚP

2.1. Phân tích bài toán tìm kiếm sản phẩm bằng cây BST

Mục đích bài toán:

Lập trình được cây nhị phân tìm kiếm với class sản phẩm: Id, Tên sản phẩm, Đánh giá, Mô tả

Nội dung bài toán:

- Chương trình thực hiện đầy đủ các chức năng của cây nhị phân tìm kiếm của sản phẩm: Chèn, Duyệt, Tìm kiếm, Xóa

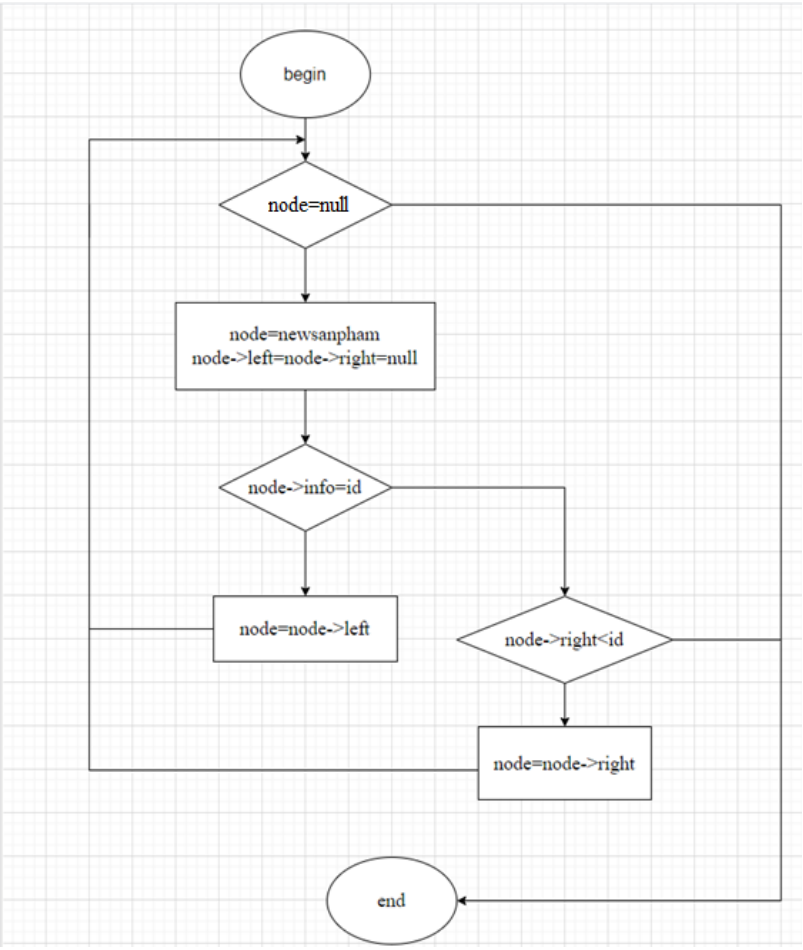
- Sản phẩm phải đầy đủ các thông tin: Id, Tên sản phẩm, Giá, Đánh giá, Mô tả

- Chạy được chương trình trên nền dữ liệu chung

- Dễ sử dụng, không phức tạp

2.2. Sơ đồ lớp

a) Chèn

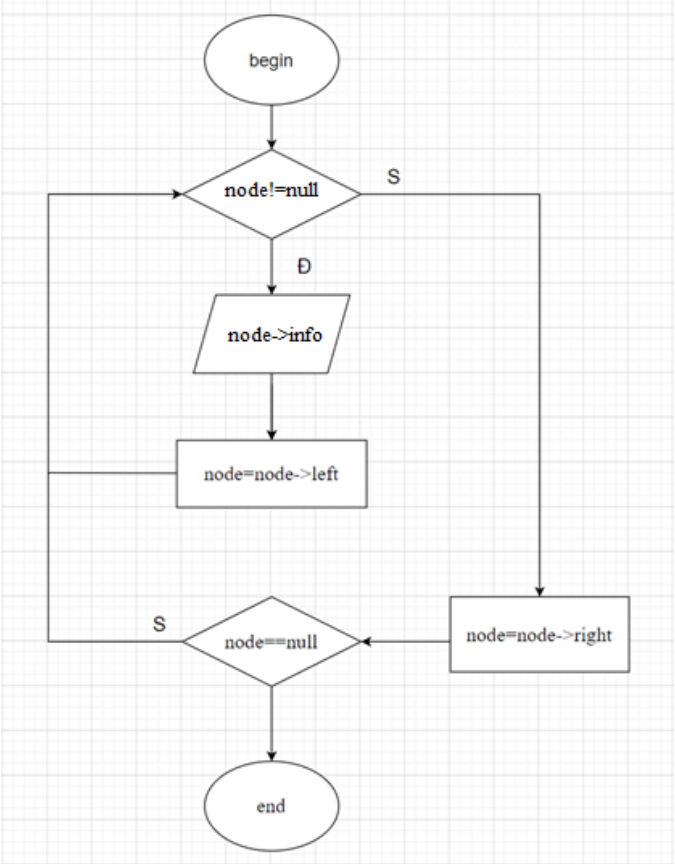


7. Sơ đồ khối chèn nút

- **B1:** Nếu cây rỗng
  - B1.1: Thêm nốt vào
  - B1.2: Gán cây bằng nốt vừa tạo
- **B2:** Nếu Id của sản phẩm mới nhỏ hơn hoặc trùng Id của sản phẩm trong cây thì thêm vào bên trái của cây
- **B3:** Nếu Id của sản phẩm mới lớn hơn Id của sản phẩm trong cây thì thêm vào bên phải của cây
- **B4:** Kết thúc

b) Duyệt

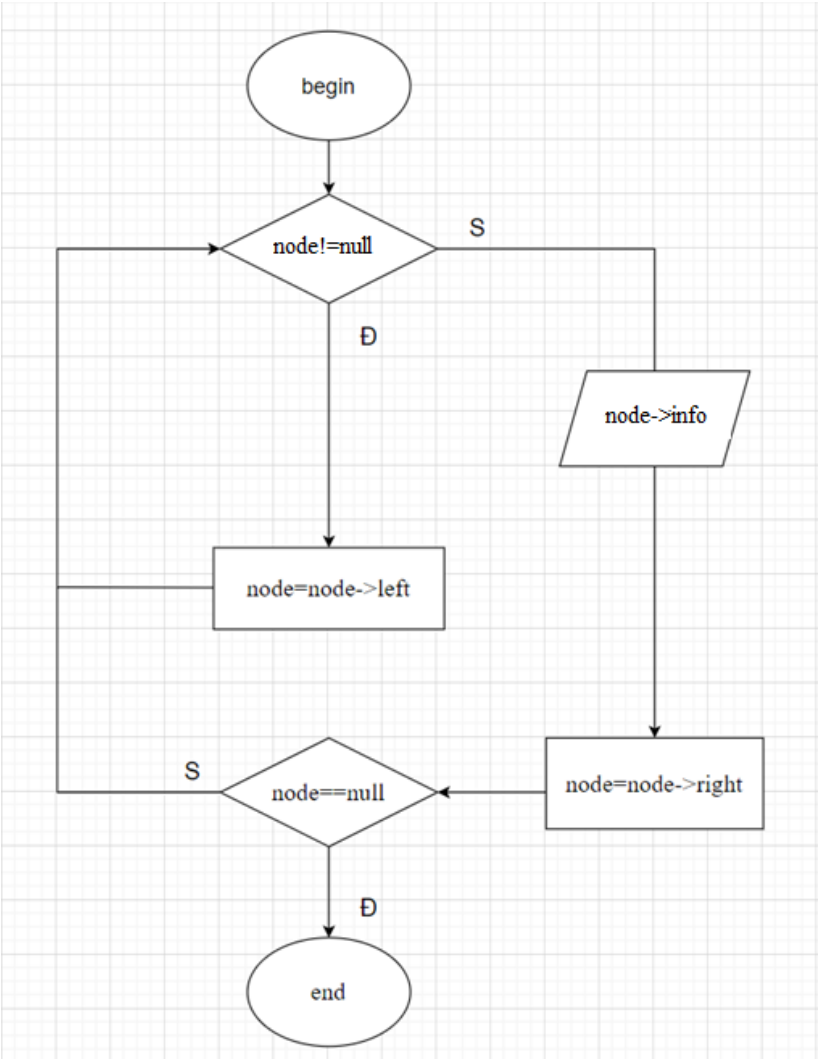
Duyệt tiền tự



8. Sơ đồ khối duyệt tiền tự

- **B1:** Nếu cây là rỗng
- **B2:** Xuất giá trị của cây
- **B3:** Duyệt tiền thứ tự cây con gốc trái
- **B4:** Duyệt tiền thứ tự cây con gốc phải

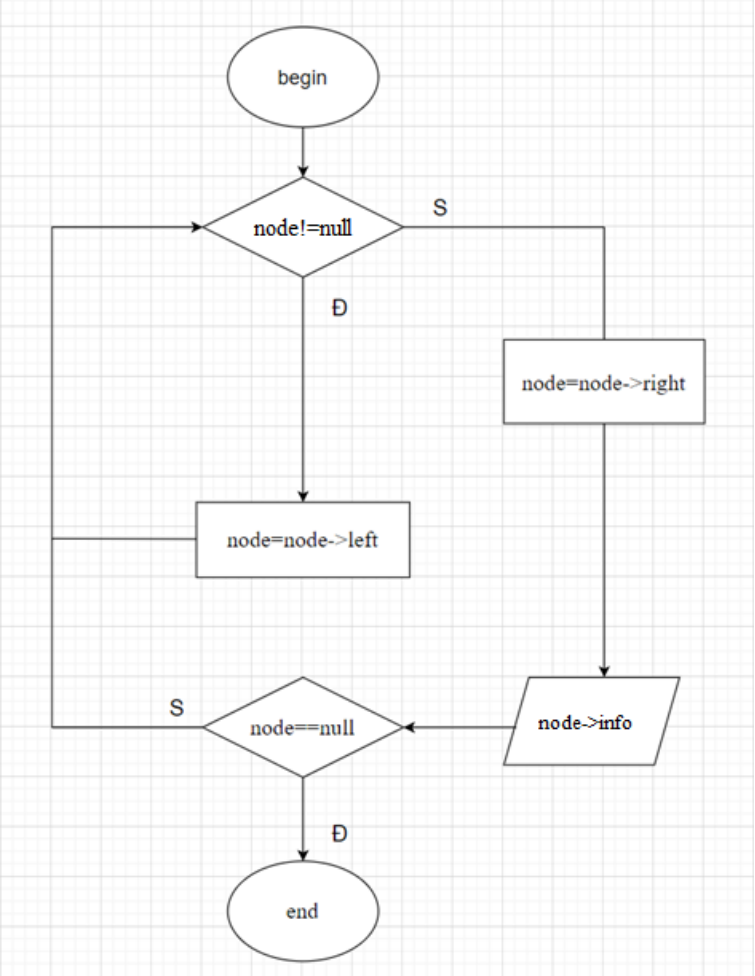
Duyệt trung tự



9. Sơ đồ khối duyệt trung tự

- **B1:** Nếu cây là rỗng
- **B2:** Duyệt trung thứ tự cây con gốc trái
- **B3:** Xuất giá trị của cây
- **B4:** Duyệt trung thứ tự cây con gốc phải

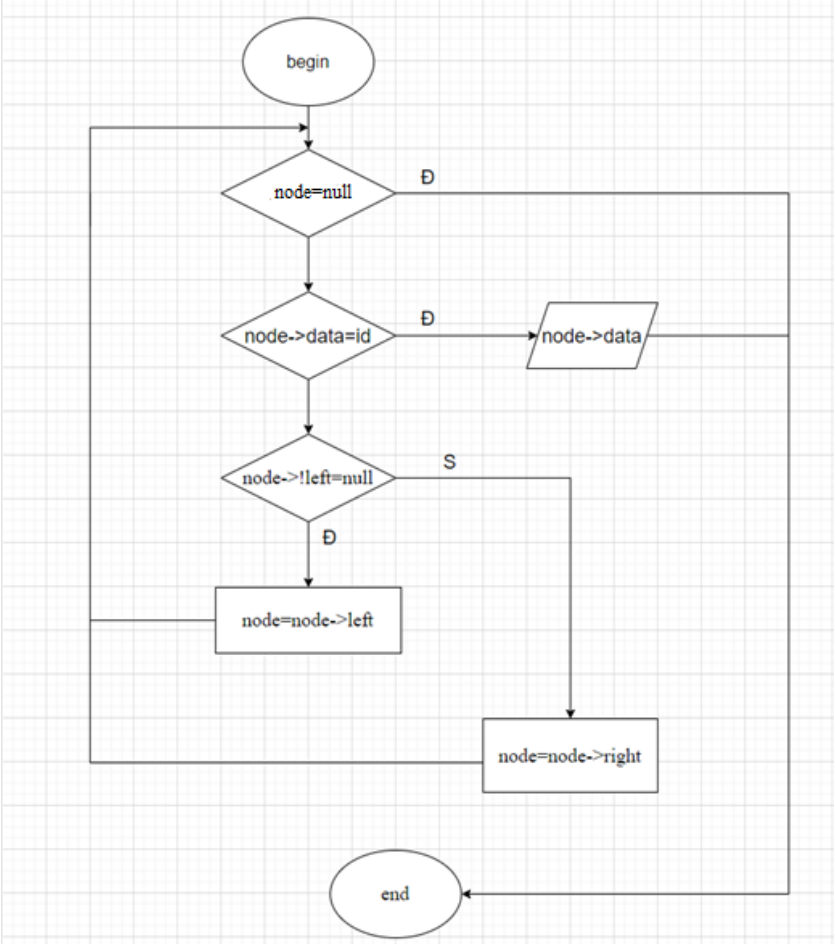
Duyệt hậu tự



10. Sơ đồ khối duyệt hậu tự

- **B1:** Nếu cây là rỗng
- **B2:** Duyệt hậu thứ tự cây con gốc trái
- **B3:** Duyệt hậu thứ tự cây con gốc phải
- **B4:** Xuất giá trị của cây

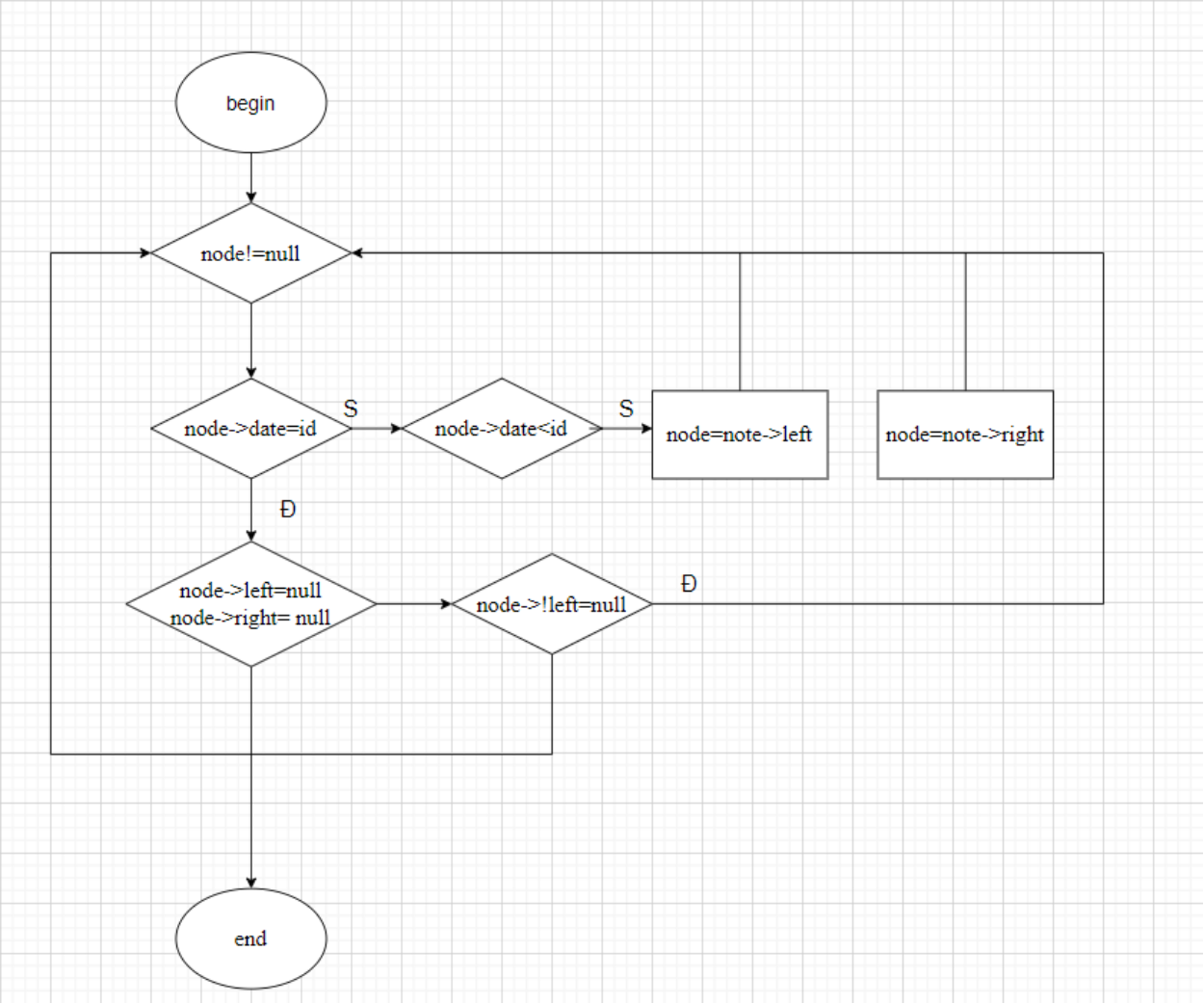
c) Tìm kiếm



11. Sơ đồ khối tìm kiếm nút

- **B1:** Nếu cây rỗng báo không tồn tại
- **B2:** Nếu Id của sản phẩm trùng với Id trong cây thì sẽ hiển thị thông tin sản phẩm đó ra màn hình
- **B3:** Nếu Id của sản phẩm không trùng với Id trong cây thì sẽ hiển thị không có kết quả tìm kiếm, xin vui lòng thử lại

d) Xóa



12. Sơ đồ khối xóa nút

- **B1:** Nếu cây rỗng thì sẽ không xóa được
- **B2:** Ngược lại

  - Nếu Id có trong cây thì xóa sản phẩm đó.

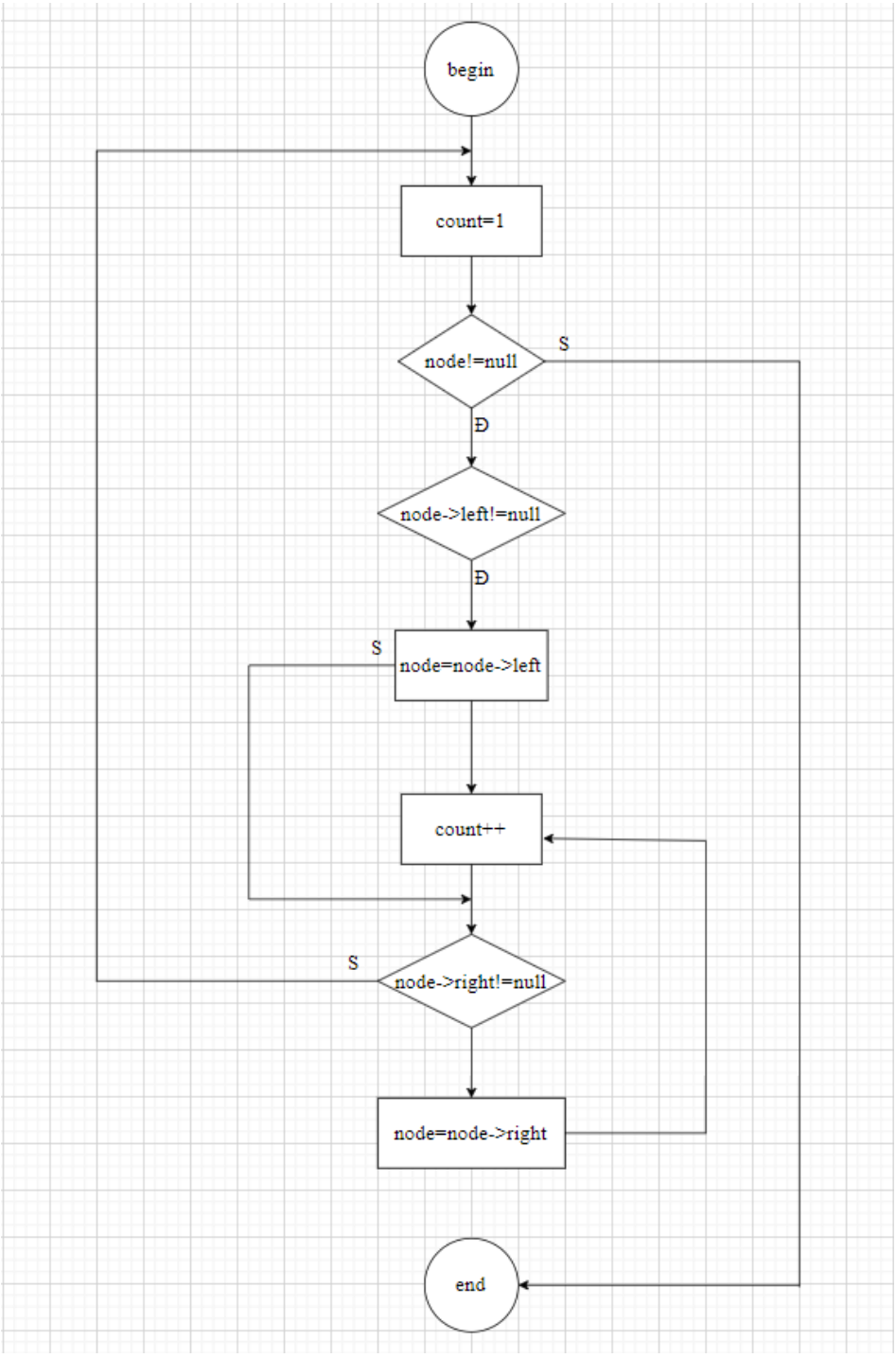
  - Nếu cây chỉ có con trái thì gán cây bằng con trái.

  - Nếu cây chỉ có con phải thì gán cây bằng con phải.

- **B3:** Ngược lại nếu Id cần xoá lớn hơn Id thì chuyển sang xoá bên phải của cây
- **B4:** Ngược lại nếu Id cần xoá nhỏ hơn Id thì chuyển sang xoá bên trái của cây

e) Tổng số nút, cạnh

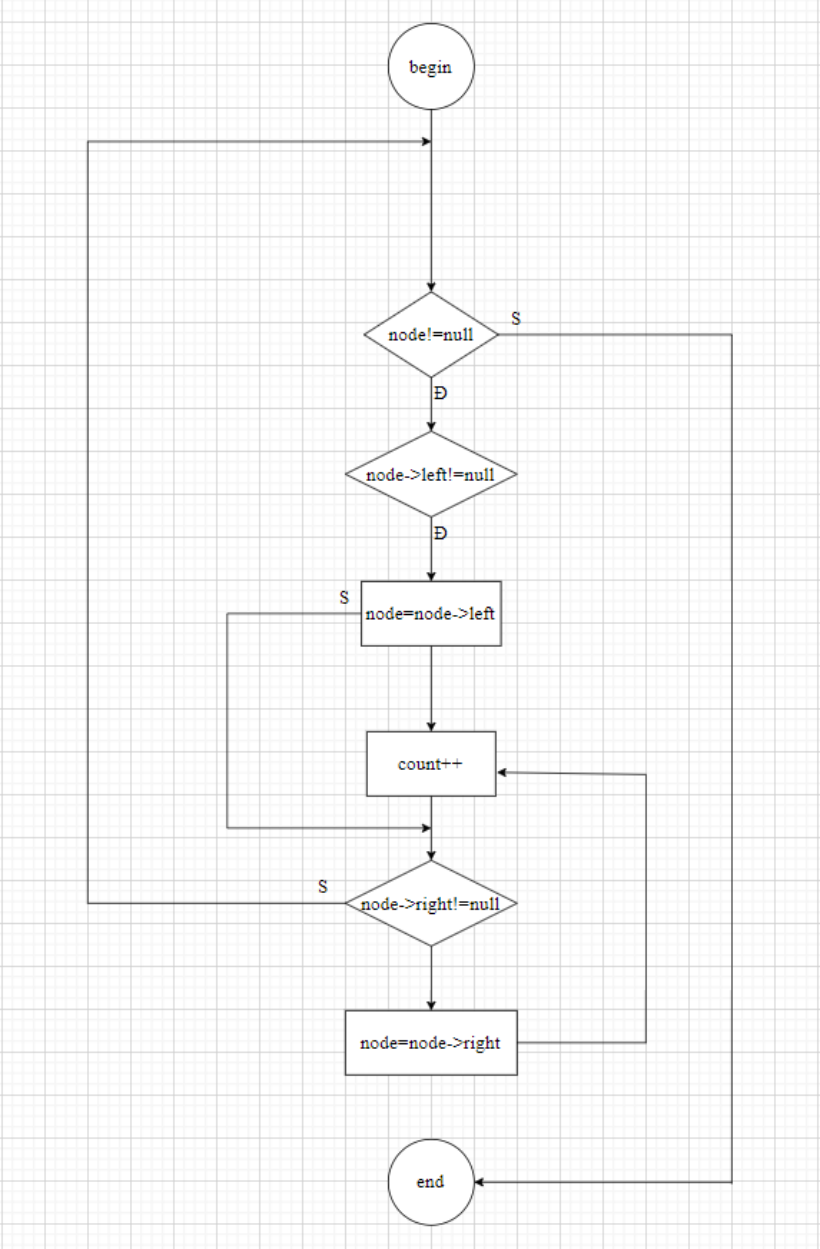
Tổng số nút



134. Sơ đồ tổng số nút

- **B1**: Nếu cây rỗng trả về 0
- **B2**: Ngược lại trả về 1 cộng với hàm đếm số sản phẩm cây con trái cộng với hàm đếm số cây con phải

Tổng số cạnh

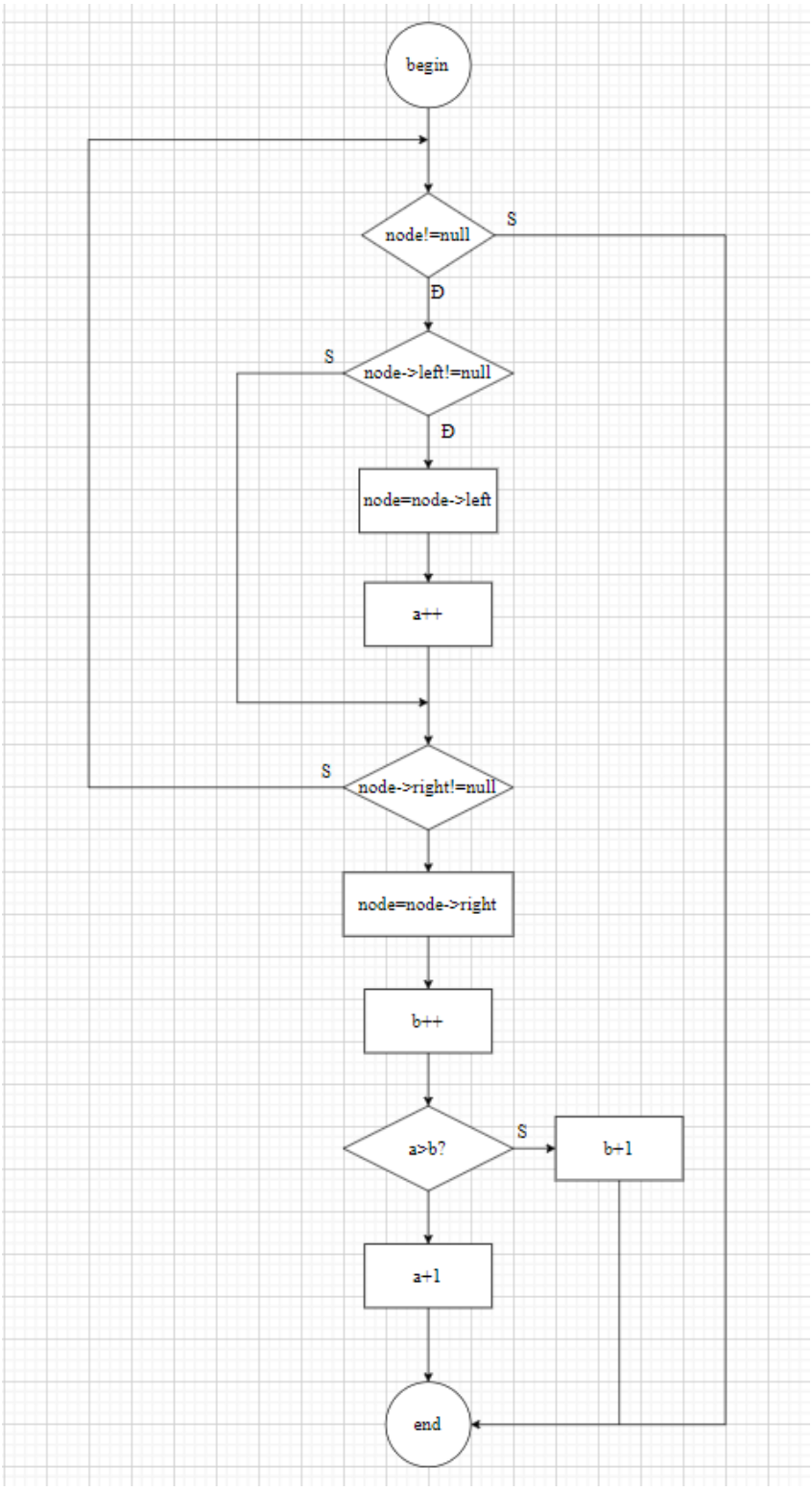


145. Sơ đồ tổng số cạnh

- **B1**: Nếu cây rỗng trả về 0
- **B2**: Ngược lại trả về hàm đếm số sản phẩm cây con trái cộng với hàm đếm số cây con phải

f) Độ sâu max, độ sâu min

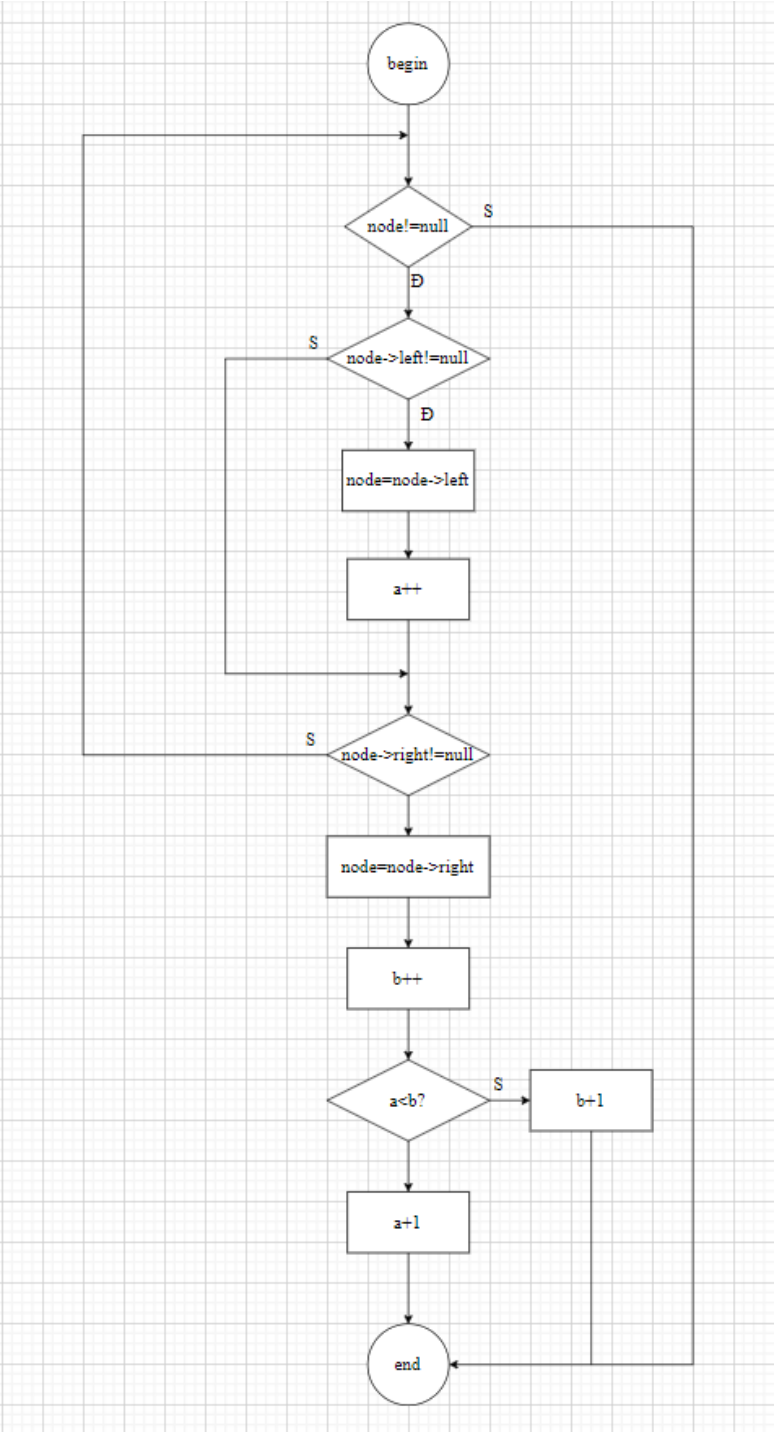
Độ sâu max



156. Sơ đồ độ sâu max

- **B1**: Nếu cây rỗng trả về -1
- **B2**: Ngược lại, tìm độ sâu lớn nhất của cây con bên trái và cây con bên phải
- **B3**: Độ sâu lớn nhất bằng max của cây con bên trái và cây con bên phải cộng 1

Độ sâu min



167. Sơ đồ độ sâu min

- **B1**: Nếu cây rỗng trả về -1
- **B2**: Ngược lại, tìm độ sâu nhỏ nhất của cây con bên trái và cây con bên phải
- **B3**: Độ sâu nhỏ nhất bằng min của cây con bên trái và cây con bên phải cộng 1

2.3. Cài đặt lớp

a) Chèn

BSTree

```
public  void insert(Node newNode)
            {
                  root = insert(root, newNode);
            }

            private Node insert(Node node, Node newNode)
            {
                  if (node == null)
                  {
                        size++;
                        return newNode;
                  }

                  if (newNode.lessThanOrEqual(node.product))
                  {
                        node.left = insert(node.left, newNode);
                  }
                  else
                  {
                        node.right = insert(node.right, newNode);
                  }
                  return node;
            }
```

Program

```
case "1":
                        Console.Write("Nhập lần lượt thông tin sản phẩm cần thêm: ");
                        Console.Write("\nId = ");
                        _id = Console.ReadLine();
                        _Id = int.Parse(_id);
                        Console.Write("Product Name = ");
                        _Productname = Console.ReadLine();
                        Console.Write("Product Price = ");
                        _price = Console.ReadLine();
                        _Price = int.Parse(_price);
                        Console.Write("Product Rating =");
                        _rating = Console.ReadLine();
                        _Rating = int.Parse(_rating);
                        Console.Write("Product Description =");
                        _Des = Console.ReadLine();
```

```
                        tree.insert(new Node(new Product(_Id, _Productname, _Price, _Rating,
_Des)));
                        continue;
```

b) Duyệt

BSTree

```
//TraverseInOrder
              public void TraversetInOrder()
              {
                     Console.Write("\n\n----------------------------------TraverseInOrder----
-----------------------------------");
                     TraverseInOrder(root);
              }

              private void TraverseInOrder(Node node)
              {
                     if (node == null) return;
                     TraverseInOrder(node.left);
                     node.Data();
                     TraverseInOrder(node.right);
              }
//TraversePostOrder
              public void TraversePostOrder()
              {
                     Console.Write("\n\n---------------------------------------
TraversePostOrder----------------------------------------");
                     TraversePostOrder(root);
              }

              private void TraversePostOrder(Node node)
              {
                     if (node == null)
                     {
                            return;
                     }
                     TraversePostOrder(node.left);
                     TraversePostOrder(node.right);
                     node.Data();
              }
```

Program

```
case "2":
                        try
                        {
                            Console.Write("1 -- Inorder\n2 -- Preorder\n3 -- Postorder\n");
                            Console.Write("Chọn Số: ");
```

```
                sel = Console.ReadLine();
                switch (sel)
                {
                    case "1":
                        tree.TraversetInOrder();
                        break;
                    case "2":
                        tree.TraversePreOrder();
                        break;
                    case "3":
                        tree.TraversePostOrder();
                        break;
                    default:
                        Console.WriteLine("Sai số, hãy thử lại");
                        continue;
                }
                continue;
            }
            catch
            {
                Console.WriteLine("Không tồn tại cây");
                continue;
            }
```

c) Tìm kiếm

BSTree

```
public Node search(int productId)
        {
            return search(this.root, productId);
        }

        private Node search(Node node, int productId)
        {
            if (node == null)    return null;


            if (node.hasproduct(productId))
            {
                return node;
            }
            if (node.lessThan(productId))
            {
                return search(node.right, productId);
            }
            else
            {
                return search(node.left, productId);
            }
        }
```

Program

```
case "3":
                        try
                        {
                            Console.Write("Id sản phẩm cần tìm : ");
                            _id = Console.ReadLine();
                            _Id = int.Parse(_id);
                            Node node = tree.search(_Id);
                            if (node == null)
                            {
                                Console.WriteLine("\n\nKhông có kết quả tìm kiếm. Xin vui
lòng thử lại! \n");
                            }
                            else
                            {
                                Console.WriteLine("\n\n---Kết quả tìm kiếm sản phẩm có Id =
" + _Id);
                                node.Data();
                            }
                            continue;
                        }
                        catch
                        {
                            Console.WriteLine("\n\nKhông có kết quả tìm kiếm. Xin vui lòng
thử lại! \n");
                            continue;
                        }
```

d) Xóa

BSTree

```
public void delete(int productId)
              {
                     root = delete(root, productId);
              }

              public Node delete(Node node, int productId)
              {
                     if (node == null) return null;


                     if (node.hasproduct(productId))
                     {
                            size--;
                            if (node.left == null)
                            {
                                   return node.right;
                            }
                            Node maxNode = node.left;
                            Node preNode = maxNode;
```

```
                    while (maxNode.right != null)
                    {
                        preNode = maxNode;
                        maxNode = maxNode.right;
                    }

                    maxNode.right = node.right;
                    if (maxNode != node.left)
                    {
                        preNode.right = maxNode.left;
                        maxNode.left = node.left;

                    }
                    return maxNode;
                }
                if (node.lessThan(productId))
                {
                    node.right = delete(node.right, productId);
                }
                else
                {
                    node.left = delete(node.left, productId);
                }
                return node;

            }
```

Program

```
case "4":
                    try
                    {
                        Console.Write("Id sản phẩm cần xóa: ");
                        _id = Console.ReadLine();
                        _Id = int.Parse(_id);
                        Node node = tree.search(_Id);
                        if (node == null)
                        {
                            Console.WriteLine("\n\nKhông có kết quả. Xin vui lòng thử
lại! \n");
                        }
                        else
                        {
                            tree.delete(_Id);
                            Console.WriteLine("\n\n---Đã xóa sản phẩm có Id = " + _Id);
                            tree.TraversePreOrder();
                        }
                        continue;
                    }
                    catch
                    {
                        Console.WriteLine("\n\nKhông có kết quả tìm kiếm. Xin vui lòng
thử lại! \n");
```

```
                    continue;
                }
```

e) Tổng số nút, cạnh

BSTree

//Tổng nút

```
public int numNodesIn(Node node)
        {
            if (node == null) return 0;
            return 1 + numNodesIn(node.left) + numNodesIn(node.right);
        }
        public int numNodesIn()
        {
            return numNodesIn(this.root);
        }
```

//Tổng cạnh

```
public int numEdgesIn(Node node)
        {
            return node == null ? 0 : numNodesIn(node) - 1;
        }
        public int numEdgesIn()
        {
            return numEdgesIn(this.root);
        }
```

Program

```
case "2":
                    try
                    {
                        Console.WriteLine("Tổng số nút có trên cây là: " +
tree.numNodesIn());
                        Console.WriteLine("Tổng số cạnh có trên cây là: " +
tree.numEdgesIn());
```

f) Độ sâu max, min

//Max Depth

```
        public int maxDepth(Node node)
        {
            if (node == null)
                return -1;
            else
            {
                int leftDep = maxDepth(node.left);
                int rightDep = maxDepth(node.right);

                if (leftDep > rightDep)
                {
                    return (leftDep + 1);
                }
                else
                {
```

```
                        return (rightDep + 1);
                    }
                }
            }
            public int maxDepth()
            {
                return maxDepth(this.root);
            }
```

//Min Depth

```
            public int minDepth(Node node)
            {
                if (node == null)
                    return -1;
                else
                {
                    int leftDep = minDepth(node.left);
                    int rightDep = minDepth(node.right);

                    if (leftDep < rightDep)
                    {
                        return (leftDep + 1);
                    }
                    else
                    {
                        return (rightDep + 1);
                    }
                }
            }
            public int minDepth()
            {
                return minDepth(this.root);
            }
```

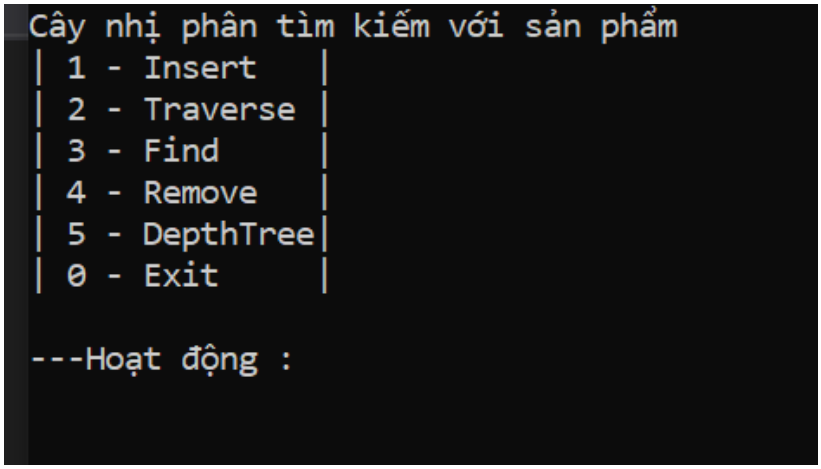
Program

```
case "5":
                    {
                        Console.Write("Độ sâu lớn nhất của cây là: " + tree.maxDepth());
                        Console.Write("\nĐộ sâu nhỏ nhất của cây là: " +
tree.minDepth());
                        break;
                    }
```

CHƯƠNG 3: THIẾT KẾ GIAO DIỆN

3.1. Giao Diện Menu Chính

Dưới đây là giao diện Menu chính của chương trình khi bắt đầu khởi chạy:



178. Giao diện Menu chính

Hướng dẫn thao tác:

Để lựa chọn các tác vụ trong Menu, người dùng chỉ cần nhập số từ 1 - 4 tùy vào nhu cầu sử dụng và sau đó nhấn Enter:

Số 1 – Insert: Chèn sản phẩm

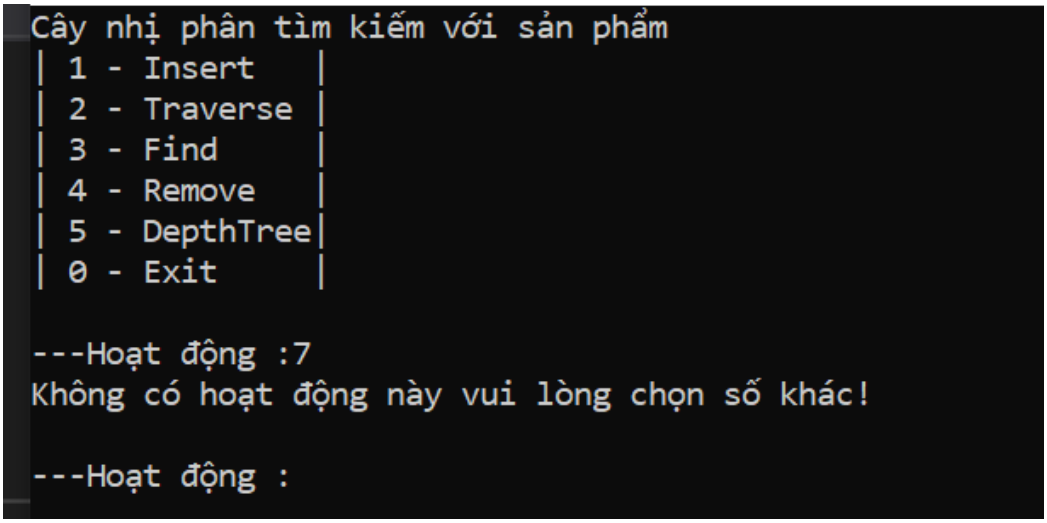
Số 2 – Traverse: Duyệt cây và đếm tổng số nút, số cạnh

Số 3 – Find: Tìm kiếm sản phẩm

Số 4 – Remove: Xóa sản phẩm

Số 5 – DepthTree: Tìm độ sâu lớn nhất và nhỏ nhất của cây

- Nếu muốn dừng chương trình và thoát khỏi giao diện Console, người dùng nhập số 0.
- Nếu người dùng nhập số khác với 5 số trên, chương trình sẽ trả về thông báo "*Không có hoạt động này vui lòng chọn số khác!*"
- Đến khi nào người dùng nhập đúng số đã cho, chương trình sẽ bắt đầu thực hiện các chức năng trên cây nhị phân tìm kiếm.

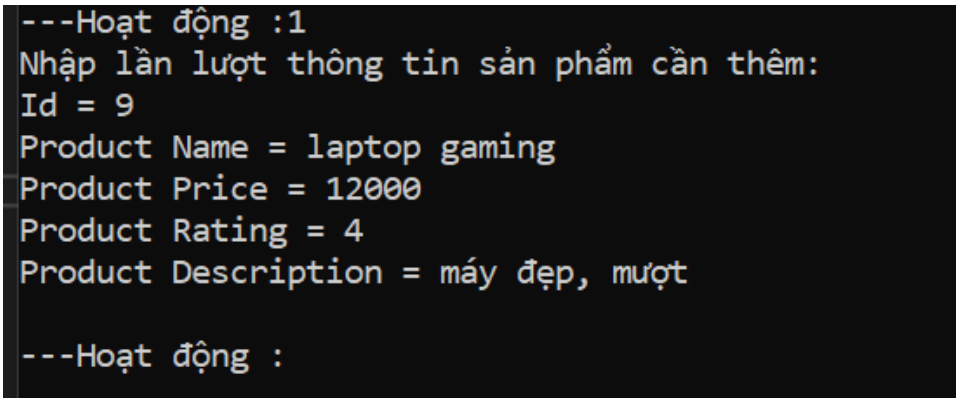


189. Nhập số sai trong Menu

3.2. Chi tiết chức năng

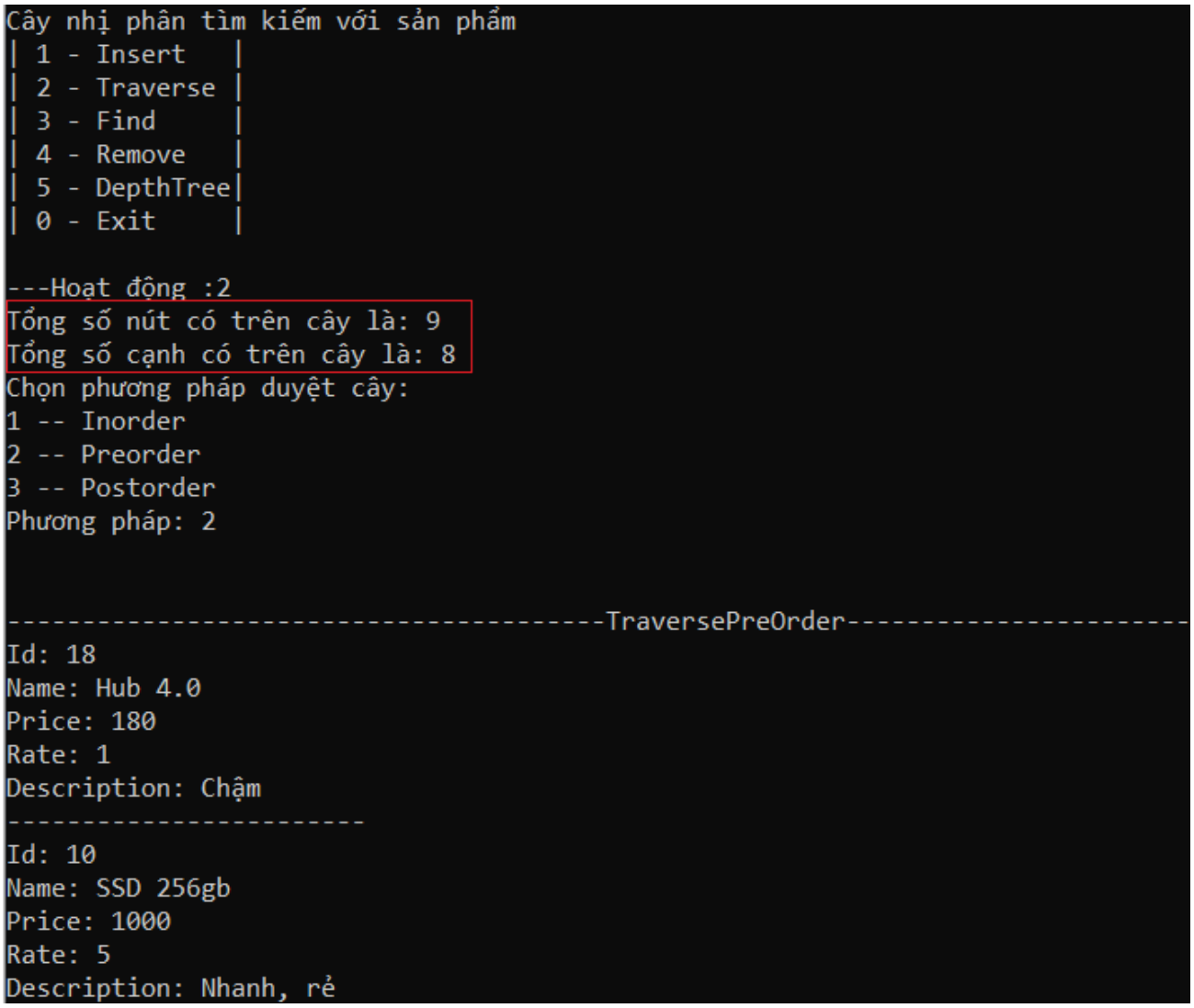
a) Chức năng 1: Chèn sản phẩm

Người dùng nhập lần lượt các thông tin về sản phẩm cần thêm vào cây, bao gồm: **Id** (Id sản phẩm), **Product Name** (tên sản phẩm). **Product Price** (giá sản phẩm), **Product Rating** (đánh giá về sản phẩm), **Product Description** (mô tả sản phẩm).



20. Demo chèn sản phẩm

b) Chức năng 2: Duyệt sản phẩm và đếm tổng số nút, tổng số cạnh



21. Demo duyệt sản phẩm

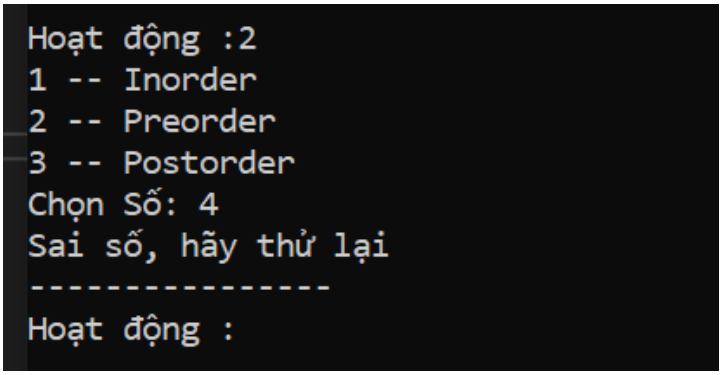
Trong *chức năng 2* bao gồm 3 cách duyệt sản phẩm khác nhau:

1 -- InOrder: Duyệt trung thứ tự

2 -- PreOrder: Duyệt tiền thứ tự

3 -- PostOrder: Duyệt hậu thứ tự

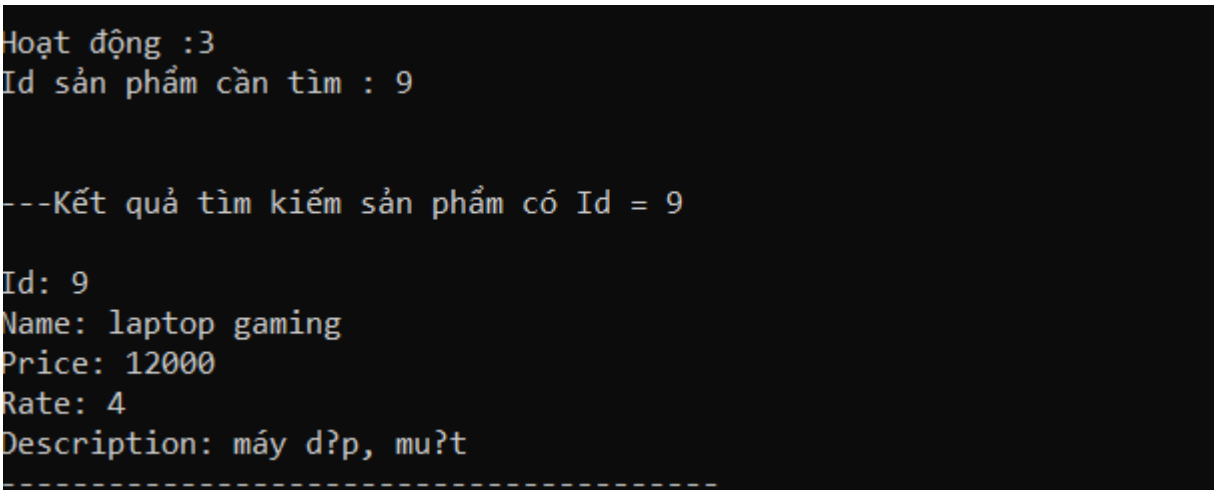
- Để lựa chọn tác vụ, người dùng chỉ cần nhập số tương ứng từ 1 – 3.
- **Note:** Ngoài ra, chức năng này còn cung cấp thêm thông tin về *tổng số nút* và *tổng số cạnh* ở trên cây.
- Nếu người dùng nhập số khác, chương trình trả về thông báo *"Sai số, hãy thử lại"*



22. Nhập số sai

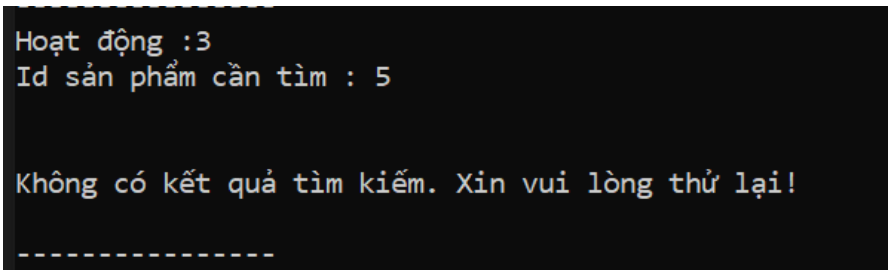
c) Chức năng 3: Tìm kiếm sản phẩm

Để tìm kiếm sản phẩm, người dùng chỉ cần nhập vào Id tương ứng, chương trình sẽ trả về Kết quả tìm kiếm gồm toàn bộ thông tin của sản phẩm có Id đó (*Id, Name, Price, Rate và Description*)



23. Demo tìm kiếm sản phẩm

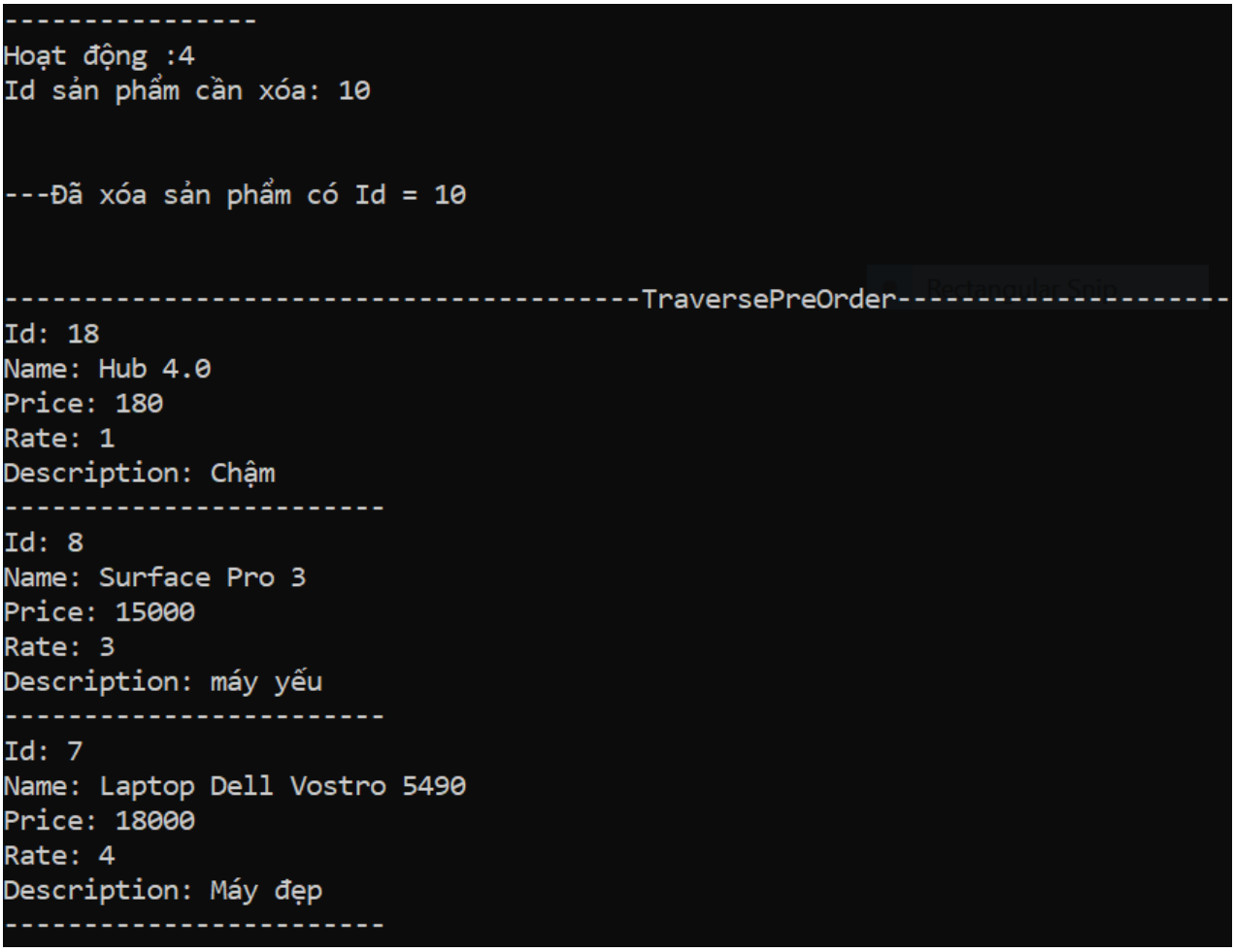
- Nếu Id nhập vào không tương thích với bất kỳ sản phẩm nào đang có, chương trình sẽ trả về thông báo "*Không có kết quả tìm kiếm. Xin vui lòng thử lại!*"



194. *Nhập sai Id*

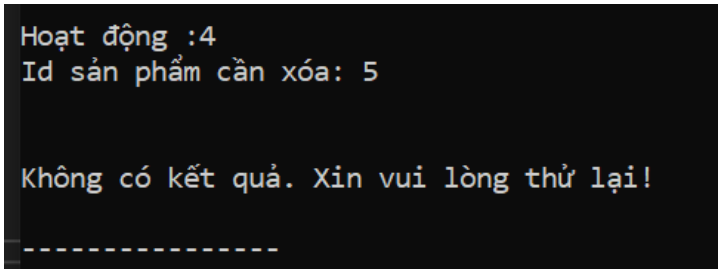
d) Chức năng 4: Xóa sản phẩm

Để xóa sản phẩm, người dùng chỉ cần nhập vào Id của sản phẩm đó. Sau khi xóa xong, chương trình sẽ trả về danh sách các sản phẩm còn lại của BSTree.



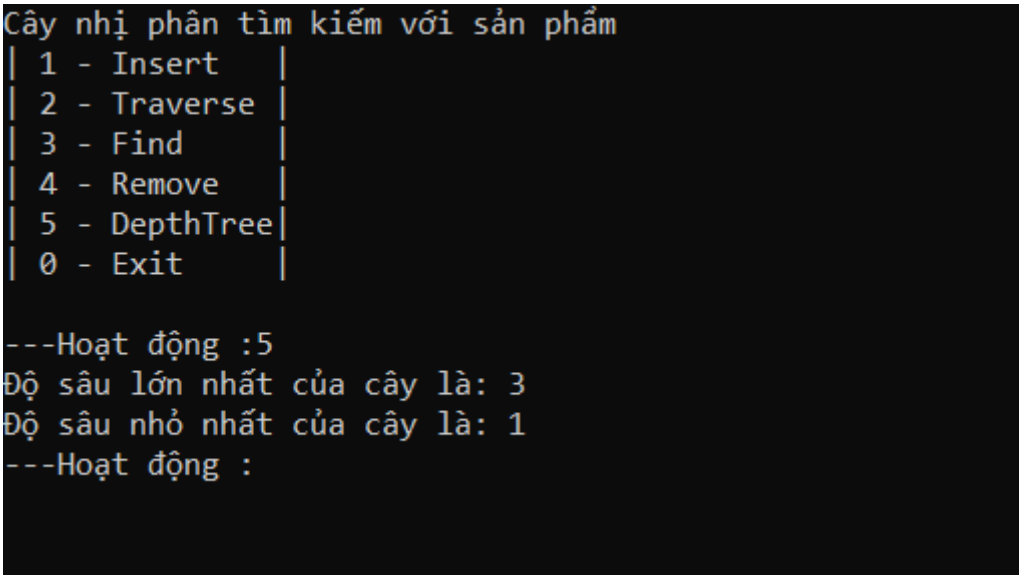
205. *Demo xóa sản phẩm*

- Nếu Id nhập vào không tương thích với bất kỳ sản phẩm nào đang có, chương trình sẽ trả về thông báo "*Không có kết quả tìm kiếm. Xin vui lòng thử lại!*"



216. *Nhập sai Id*

e) Chức năng 5: Tìm độ sâu lớn nhất và nhỏ nhất của cây



27. Tìm độ sâu lớn nhất và nhỏ nhất

Chức năng này sẽ trả về độ sâu lớn nhất và nhỏ nhất của cây nhị phân tìm kiếm.

CHƯƠNG 4: THẢO LUẬN & ĐÁNH GIÁ

5.1. Các Kết Quả Nhận Được

Qua quá trình thực hiện đồ án, nhóm đã vận dụng kiến thức đã học về cây nhị phân tìm kiếm để đáp ứng yêu cầu bài toán quản lý sản phẩm. Chương trình đã thực hiện được các chức năng quan trọng của Binary Search Tree như:

- Thêm sản phẩm mới
- Duyệt sản phẩm theo 3 cách (tiền tự, trung tự, hậu tự)
- Tìm kiếm sản phẩm
- Xóa sản phẩm
- Đếm tổng số sản phẩm
- Đếm tổng số cạnh
- Tìm độ sâu lớn nhất, nhỏ nhất

Chương trình cũng được chia bố cục rõ ràng:

- Class Program sẽ có vai trò xây dựng giao diện Menu giúp người sử dụng dễ dàng thao tác.
- Class Product thì sẽ gồm những thông tin sản phẩm như Id, tên, đánh giá và mô tả sản phẩm.
- Class Node với trường dữ liệu Product.
- Class BSTree với các phương thức, thuật toán trên cây nhị phân tìm kiếm.

5.2. Một Số Tồn Tại

Chương trình vẫn còn nhiều hạn chế như quy mô và cơ sở dữ liệu nhỏ, các chức năng chưa tối ưu, chưa phát triển được nhiều thuật toán có độ phức tạp cao hơn để ứng dụng vào thực tiễn. Do thời gian thực hiện có hạn, nên vẫn còn nhiều thiếu sót mong thầy có thể góp ý chi tiết để chúng em nhận ra ưu, nhược điểm của chương trình và phát triển chương trình tốt hơn trong tương lai.

5.3. Hướng Phát Triển

Điểm mạnh của cây BST là tính ứng dụng trong việc tìm kiếm. Vì vậy trong tương lai, chương trình sẽ tiếp tục được xây dựng hoàn thiện hơn, phát triển thêm các thuật toán phức tạp để cải tiến công tác quản lý sản phẩm. Đồng thời mở rộng thêm quy mô cơ sở dữ liệu lớn và phong phú hơn.

PHỤ LỤC I

BẢNG PHÂN CÔNG CÔNG VIỆC	
Nguyễn Đỗ Đức Hào	- Lập trình xây dựng cây nhị phân tìm kiếm với Class Product: id, name, price, rate, description. - Lập trình các chức năng: thêm, xóa, duyệt cây (theo tiền, trung, hậu)
Nguyễn Lê Nguyên	- Lập trình xây dựng bảng menu cho chương trình. - Lập trình chức năng đếm tổng số cạnh
Hoàng Thị Đan Phương	- Làm powerpoint để thuyết trình
Nguyễn Quỳnh Khánh Hà	- Làm và chỉnh sửa báo cáo chương 3-4 - Lập trình chức năng: đếm tổng số nút, tìm độ cao lớn nhất/ nhỏ nhất
Trần Phạm Minh Việt	- Làm và chỉnh sửa báo cáo chương 1-2

PHỤ LỤC II

Link github mã nguồn: https://github.com/haohaoo2012/DoAnCauTrucDuLieu_BStree

DANH MỤC THAM KHẢO

[1] Slide bài học chương 8

[2] https://vimentor.com/vi/lesson/cay-ti-m-kie-m-nhi-phan-1

----------Hết----------