BỘ GIÁO DỤC VÀ ĐÀO TẠO

**TRƯỜNG ĐẠI HỌC NHA TRANG**

KHOA CÔNG NGHỆ THÔNG TIN



**THỰC TẬP CƠ SỞ CÔNG NGHỆ THÔNG TIN**

**THAO TÁC TRÊN CÂY NHỊ PHÂN TÌM KIẾM**

**Giảng viên hướng dẫn: ThS. Nguyễn Thị Hương Lý**

**Sinh viên thực hiện: Trần Quang Vinh**

**Lớp: 64.CNTT-1**

Khánh Hòa – 2024

BỘ GIÁO DỤC VÀ ĐÀO TẠO

**TRƯỜNG ĐẠI HỌC NHA TRANG**

KHOA CÔNG NGHỆ THÔNG TIN



**THỰC TẬP CƠ SỞ CÔNG NGHỆ THÔNG TIN**

**THAO TÁC TRÊN CÂY NHỊ PHÂN TÌM KIẾM**

GVHD: ThS. Nguyễn Thị Hương Lý

Sinh viên thực hiện: Trần Quang Vinh

Lớp: 64.CNTT-1

Khánh Hòa – 2024

**MỤC LỤC**

LỜI NÓI ĐẦU 9

CHƯƠNG 1: TỔNG QUAN VỀ ĐỀ TÀI 10

1.1 Lý do chọn đề tài 10

1.2 Mục tiêu của đề tài 10

1.3 Phạm vi nghiên cứu 11

1.4 Phương pháp nghiên cứu 11

CHƯƠNG 2: CƠ SỞ LÝ THUYẾT 12

2.1 Lý thuyết về cây nhị phân tìm kiếm 12

2.1.1 Định nghĩa về cây nhị phân tìm kiếm: 12

2.1.2 Ưu và nhược điểm: 13

2.1.2 So sánh độ phức tạp với các cấu trúc dữ liệu khác: 13

2.2 Tổng quan về thuật toán 14

2.2.1 Cấu trúc một nút: 14

2.2.2 Chèn một nút vào BST: 14

2.2.3 Xóa một nút trong BST: 15

[2.2.4 Thể hiện trực quan logic BST: 17](#_Toc93268356)

CHƯƠNG 3: XÂY DỰNG CHƯƠNG TRÌNH 20

3.1 Công cụ và ngôn ngữ được sử dụng 20

3.1.1 Visual Studio Code (VS Code) 20

3.1.2 Ngôn ngữ HTML, CSS 20

3.1.3 Ngôn ngữ lập trình JavaScript 21

3.2 Đặc tả bài toán 21

3.2.1 Yêu cầu đề tài 22

3.2.2 Hướng dẫn xử lý bài toán 22

3.3 Cài đặt chương trình 23

3.3.1 Mã nguồn chương trình và kết quả chương trình 24

3.3.1.1 Tạo cây nhị phân 24

3.3.1.2 Thêm một nút 25

3.3.1.3 Xóa một nút 26

KẾT LUẬN 27

1.Các kết quả đạt được. 27

2.Ưu và nhược điểm của đề tài 27

TÀI LIỆU THAM KHẢO 28

**TRƯỜNG ĐẠI HỌC NHA TRANG**

KHOA CÔNG NGHỆ THÔNG TIN

**PHIẾU THEO DÕI TIẾN ĐỘ VÀ ĐÁNH GIÁ  
THỰC TẬP CƠ SỞ**   
**(Dùng cho CBHD và nộp cùng báo cáo thực tập cơ sở của sinh viên)**

**Tên đề tài:** : Thể hiện trực quan cây nhị phân tìm kiếm

**Giáo viên hướng dẫn:** Nguyễn Thị Hương Lý

**Sinh viên được hướng dẫn:** Trần Quang Vinh

**MSSV:** 64132999

**Khóa:** 64.CNTT-1 **Ngành:** Công Nghệ Thông Tin

|  |  |  |  |
| --- | --- | --- | --- |
| **Lần** | **Ngày** | **Nội dung** | **Nhận xét của GVHD** |
| 1 |  |  |  |
| 2 |  |  |  |
| 3 |  |  |  |
| 4 |  |  |  |
| 5 |  |  |  |
| 6 |  | . |  |

**Nhận xét chung** (sau khi sinh viên hoàn thành thực tập cơ sở):

………………..…………………………………………………………………….……

…………………………………………………………………..………….……………

…………………………………………..………………………………………….……

Điểm hình thức:……/10 Điểm nội dung:......./10 **Điểm tổng kết**: ………/10

Kết luận sinh viên: Được bảo vệ:  Không được bảo vệ: 

*Khánh Hòa, ngày…….tháng…….năm……*

**Cán bộ hướng dẫn**

*(Ký và ghi rõ họ tên)*

**TRƯỜNG ĐẠI HỌC NHA TRANG**

KHOA CÔNG NGHỆ THÔNG TIN

**PHIẾU CHẤM ĐIỂM THỰC TẬP CƠ SỞ**   
**(Dành cho cán bộ chấm phản biện)**

**1. Họ tên người chấm**: …………………………………………………….

**2. Sinh viên thực hiện:** …………………………….…………… MSSV: …………….  
**3. Tên đề tài:** ………………………………………………………………....................

…………………………………………………………………………………………...

**4. Nhận xét**

a) Kết quả thực hiện đề tài

...........................................................................................................................................

...........................................................................................................................................

...........................................................................................................................................

b) Báo cáo thực tập

- Hình thức: ......................................................................................................................

...........................................................................................................................................

...........................................................................................................................................

- Nội dung: ……………………………………………………………………………...

…………………………………………………………………………………………...

…………………………………………………………………………………….…….

…………………………………………………………………………………………..

…………………………………………………………………………………………..

…………………………………………………………………………………………..

…………………………………………………………………………………….…….

Điểm hình thức:……/10 Điểm nội dung:......./10 **Điểm tổng kết**:………/10

*Khánh Hòa, ngày…….tháng…….năm……*

**Cán bộ chấm phản biện**

*(Ký và ghi rõ họ tên)*

**LỜI NÓI ĐẦU**

Công nghệ thông tin và lập trình đã trở thành nền tảng quan trọng trong thời đại hiện nay, với ứng dụng rộng rãi trong mọi lĩnh vực từ giáo dục, y tế, tài chính cho đến sản xuất và quản lý. Trong đó, các cấu trúc dữ liệu và thuật toán đóng vai trò cốt lõi, giúp tối ưu hóa quá trình xử lý thông tin và giải quyết các bài toán một cách hiệu quả.

Cây nhị phân tìm kiếm (Binary Search Tree) là một trong những cấu trúc dữ liệu cơ bản, phổ biến và hữu ích trong lập trình. Với đặc tính sắp xếp tự nhiên và khả năng truy xuất dữ liệu nhanh chóng, cây nhị phân tìm kiếm thường được sử dụng trong các hệ thống cơ sở dữ liệu, trình biên dịch, và nhiều ứng dụng khác. Tuy nhiên, việc hiểu rõ nguyên lý hoạt động, cách cài đặt và trực quan hóa cây nhị phân tìm kiếm là một thách thức đối với nhiều lập trình viên mới bắt đầu.

Dự án “Mô phỏng trực quan cây nhị phân tìm kiếm” ra đời với mục tiêu cung cấp một công cụ hữu ích để học tập và nghiên cứu. Thông qua ứng dụng, người dùng có thể tạo ra cây nhị phân tìm kiếm, thêm hoặc xóa các node, và quan sát trực quan cách thức cây thay đổi theo từng thao tác. Đây không chỉ là một bài toán lập trình mà còn là một bước tiến trong việc kết nối lý thuyết với thực hành.

Hy vọng rằng, với dự án này, người đọc sẽ nắm vững hơn về cấu trúc cây nhị phân tìm kiếm, đồng thời có thêm cảm hứng khám phá các thuật toán khác trong lĩnh vực khoa học máy tính. Dự án cũng là một bước đệm quan trọng để tiến xa hơn trong các nghiên cứu và ứng dụng thực tiễn sau này.

***Xin chân thành cảm ơn !***

**CHƯƠNG 1: TỔNG QUAN VỀ ĐỀ TÀI**

**1.1 Lý do chọn đề tài:**

Cây nhị phân tìm kiếm (BST) là một trong những cấu trúc dữ liệu cơ bản và quan trọng trong khoa học máy tính, đặc biệt là trong các bài toán yêu cầu tìm kiếm và quản lý dữ liệu một cách hiệu quả. BST có khả năng duy trì một cấu trúc phân loại tự nhiên, giúp các thao tác như tìm kiếm, thêm, xóa các phần tử trong cây trở nên nhanh chóng và hiệu quả. Chính vì vậy, BST trở thành một công cụ không thể thiếu trong việc xây dựng các ứng dụng đòi hỏi xử lý dữ liệu lớn và có cấu trúc phân cấp như trong các hệ thống cơ sở dữ liệu, công cụ tìm kiếm, hệ thống giao dịch và nhiều lĩnh vực khác.

Mặc dù cây nhị phân tìm kiếm mang lại nhiều lợi ích về hiệu suất, nhưng việc hiểu rõ nguyên lý hoạt động và các thao tác của cây này lại không phải là điều đơn giản. Đặc biệt là đối với những người mới bắt đầu học lập trình hoặc đang tìm hiểu về cấu trúc dữ liệu, việc hình dung và nắm vững các thao tác trên cây nhị phân tìm kiếm có thể gặp nhiều khó khăn. Các thao tác như tìm kiếm, thêm, xóa hay cân bằng cây đều có thể làm người học cảm thấy phức tạp nếu không có sự hỗ trợ trực quan.

Để giải quyết vấn đề này, việc phát triển một công cụ mô phỏng trực quan cây nhị phân tìm kiếm là rất quan trọng. Một công cụ như vậy không chỉ giúp người học dễ dàng theo dõi sự thay đổi của cây qua từng thao tác, mà còn giúp họ hiểu rõ hơn về cách các thuật toán hoạt động trong thực tế. Thông qua việc xây dựng và sử dụng công cụ này, người học có thể thực hành trực tiếp và quan sát được hiệu quả của các thuật toán tìm kiếm, thêm, xóa trên cây, qua đó củng cố được kiến thức lý thuyết và kỹ năng lập trình.

**1.2 Mục tiêu của đề tài:**

Mục tiêu của đề tài là xây dựng một công cụ trực quan để mô phỏng và thao tác trên cây nhị phân tìm kiếm (BST). Công cụ này giúp người dùng dễ dàng thêm, xóa và tìm kiếm trong cây nhị phân tìm kiếm một cách trực quan, từ đó hỗ trợ việc học tập và nghiên cứu về cấu trúc cây nhị phân.

**1.3 Phạm vi nghiên cứu:**

Phạm vi nghiên cứu của đề tài bao gồm:

* Nghiên cứu lý thuyết về cây nhị phân tìm kiếm.
* Xây dựng công cụ trực quan hóa BST với các thao tác cơ bản như chèn, xóa và tìm kiếm.
* Sử dụng các công cụ phát triển web như HTML, CSS, và JavaScript để tạo giao diện và chức năng cho ứng dụng.

**1.4 Phương pháp nghiên cứu:**

Phương pháp nghiên cứu được áp dụng trong đề tài bao gồm:

* Phương pháp lý thuyết: Tìm hiểu và phân tích các thuật toán cơ bản của cây nhị phân tìm kiếm, các ưu điểm và nhược điểm của cây nhị phân tìm kiếm so với các cấu trúc dữ liệu khác.
* Phương pháp mô phỏng: Xây dựng và triển khai mô phỏng các thao tác trên cây nhị phân tìm kiếm thông qua việc lập trình các thuật toán chèn, xóa, duyệt và vẽ cây.
* Phương pháp thực nghiệm: Kiểm tra và đánh giá hiệu quả của công cụ trực quan qua việc thao tác trên cây nhị phân tìm kiếm, đánh giá tính trực quan và dễ sử dụng của giao diện.

**CHƯƠNG 2: CƠ SỞ LÝ THUYẾT**

**2.1 Lý thuyết về cây nhị phân tìm kiếm**

**2.1.1 Định nghĩa về cây nhị phân tìm kiếm:**

Cây (Tree) là một cấu trúc dữ liệu tương tự như Danh sách liên kếtở việc mỗi nút chứa dữ liệu và có thể liên kết với các nút khác. Tuy nhiên, cây có một đặc điểm khác với các cấu trúc dữ liệu khác như Mảng, Danh sách liên kết, Ngăn xếp và Hàng đợi là một phần tử đơn lẻ có thể chứa nhiều phần tử ‘tiếp theo’ thay vì một phần tử. Cây nhị phân (Binary Tree) là một loại cấu trúc dữ liệu cây nhưng mỗi nút trong đó chứa tối đa hai nút con ‘trái – phải’, nhờ cấu trúc này mà cây nhị phân mang lại nhiều lợi ích như có các thuật toán duyệt, tìm, chèn và xóa dễ triển khai và chạy nhanh hơn.

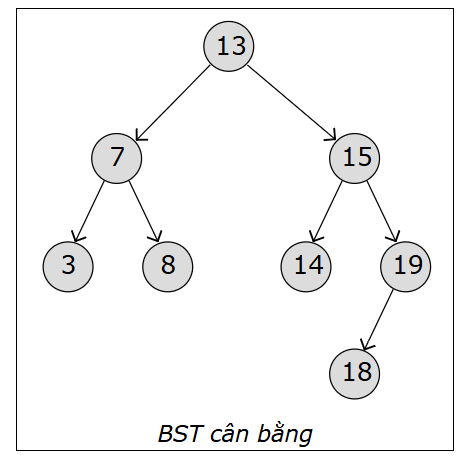
Cây nhị phân tìm kiếm (Binary Search Tree) là một cấu trúc dữ liệu dạng cây nhị phân, trong đó các thuộc tính sau phải đúng với bất kỳ nút ‘X’ nào trong cây:

* Giá trị của nút con bên trái của nút ‘X’ và các con của nó luôn nhỏ hơn giá trị của ‘X’.
* Giá trị của nút con bên phải của nút ‘X’ và các con của nó luôn lớn hơn giá trị của ‘X’.
* Cây con bên trái và phải cũng là Cây nhị phân tìm kiếm.

**2.1.2 Ưu và nhược điểm:**

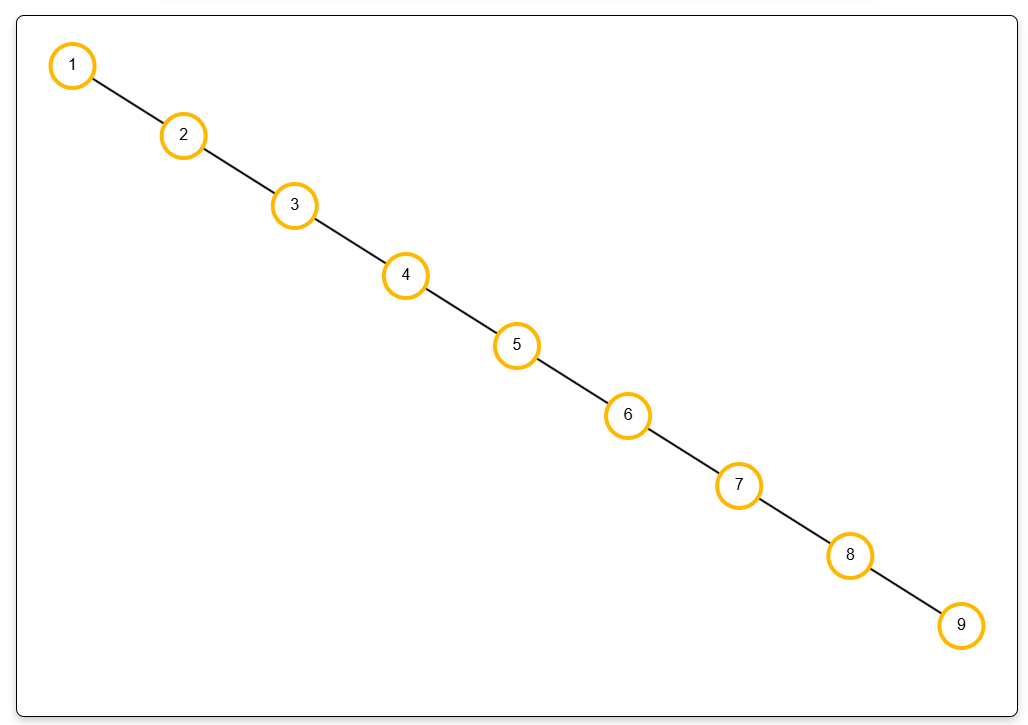
**Ưu điểm:**

* Hiệu quả: Các thao tác tìm kiểm, chèn, xóa có độ phức tạp trung bình O(log n) nếu cây được cân bằng.



* Tính mở rộng: Có thể dễ dàng mở rộng thành các cấu trúc cây nâng cao như cây AVL hoặc cây đỏ-đen để cân bằng tự động.
* Đơn giản: Cách triển khai dễ dàng và dễ hiểu

**Nhược điểm:**

* Hiệu năng không đảm bảo: Nếu cây mất cân bằng, có thể biến thành danh sách liên kết, dẫn đến độ phức tạp O(n).
* 
* Khó khăn trong cân bằng: Với dữ liệu không ngẫu nhiên hoặc bị lệch, việc duy trì cây cân bằng yêu cầu các thuật toán bổ sung phức tạp.

**2.1.3 So sánh độ phức tạp với các cấu trúc dữ liệu khác:**

Cây nhị phân tìm kiếm sử dụng những ưu điểm của hai cấu trúc dữ liệu khác: Mảng và Danh sách liên kết.

|  |  |  |
| --- | --- | --- |
| Data Structure | Searching for a value | Delete / Insert leads to shifting in memory |
| Sorted Array | O(log n) | Yes |
| Linked List | O(n) | No |
| Binary Search Tree | O(log n) | No |

Tìm kiếm BST cũng nhanh như Tìm kiếm nhị phân trên một Array, với cùng độ phức tạp

về thời gian O(log n). Và việc xóa và chèn các giá trị mới có thể được thực hiện mà không cần dịch chuyển các phần tử trong bộ nhớ, giống như với Danh sách liên kết.

**2.2 Tổng quan về thuật toán**

**2.2.1 Cấu trúc một nút:**

Mỗi nút trong cây nhị phân có thể được biểu diễn như sau trong JavaScript:

|  |
| --- |
| class Node {  constructor(data) {  this.data = data; // Giá trị tại nút  this.left = null; // Con trỏ đến con trái  this.right = null; // Con trỏ đến con phải  }  } |

**2.2.2 Chèn một nút vào BST:**

Chèn một nút mới vào cây nhị phân tìm kiếm theo quy tắc:

* Nếu nhỏ hơn hoặc bằng nút hiện tại, chèn vào cây con trái.
* Nếu lớn hơn, chèn vào cây con phải.

|  |
| --- |
| insert(value) {  const newNode = new Node(value);  if (this.root == null) {  this.root = newNode;  } else {  this.#insertNode(this.root, newNode);  }  }  #insertNode(node, newNode) {  if (newNode.value <= node.value) { //Nếu nhỏ hơn hoặc bàng nút hiện tại  if (node.left === null) { //Thêm vào con bên trái nếu chưa có  node.left = newNode;  } else { //Nếu con bên trái đã có thì duyệt tiếp.  this.#insertNode(node.left, newNode);  }  } else { // Nếu lớn hơn nút hiện tại  if (node.right === null) { //Thêm vào con bên phải nếu chưa có  node.right = newNode;  } else { //Nếu con bên trái đã có thì duyệt tiếp.  this.#insertNode(node.right, newNode);  }  }  } |

**2.2.3 Xóa một nút trong BST:**

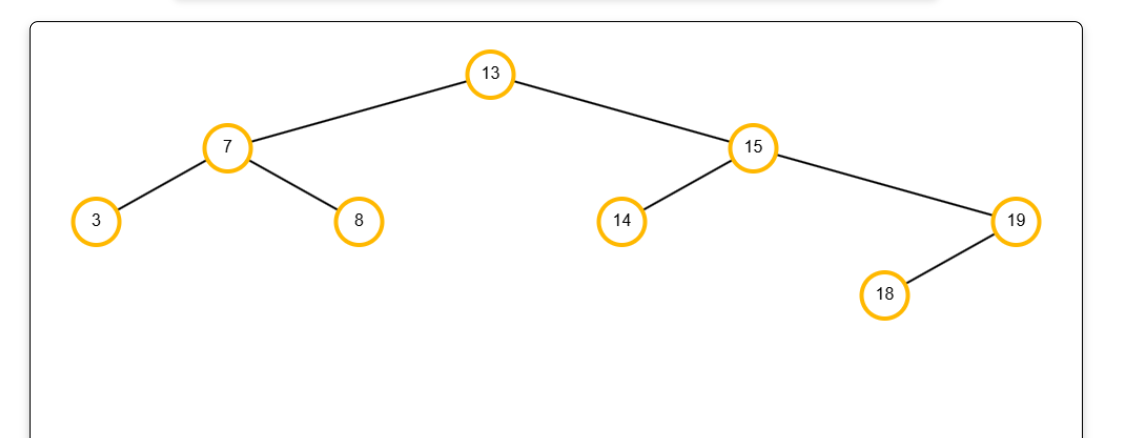
Xóa một nút trong BST bao gồm ba trường hợp:

* Nút là lá (không có con): Xóa trực tiếp.
* Nút có một con: Gán con của nút đó thay thế vị trí của nó.
* Nút có hai con:Tìm nút nhỏ nhất trong cây con phải (hoặc lớn nhất trong cây con trái). Thay giá trị của nút cần xóa bằng giá trị của nút này. Xóa nút vừa được thay thế.

|  |
| --- |
| delete(value) {  this.root = this.#deleteNode(this.root, value);  }  #deleteNode(node, value) {  if (node === null) return null; // Node không tồn tại  if (value < node.value) {  node.left = this.#deleteNode(node.left, value); // Xóa ở nhánh trái  return node;  } else if (value > node.value) {  node.right = this.#deleteNode(node.right, value); // Xóa ở nhánh phải  return node;  } else { // Node cần xóa đã được tìm thấy  if (node.left === null && node.right === null) { // Trường hợp node không có con  return null;  } else if (node.left === null) { // Trường hợp node chỉ có con phải  return node.right;  } else if (node.right === null) { // Trường hợp node chỉ có con trái  return node.left;  } else {  const minRightValue = this.#findMinNode(node.right).value;  node.value = minRightValue;  node.right = this.#deleteNode(node.right, minRightValue);  return node;  }  }  }  #findMinNode(node) {  while (node.left !== null) {  node = node.left;  }  return node;  } |

**2.2.4 Thể hiện trực quan logic BST:**

Thể hiện các nút lên tọa độ ngang (x):



Để ánh xạ các nút lên tọa độ ngang thì ta phải lấy ra các nút có giá trị từ nhỏ đến lớn (hàm inOrder), sau đó ta chia chiều rộng của canvas cho số lượng nút trong cây để có khoảng cách các nút. Và tọa độ của các nút sẽ là index (vị trí phần tử sau khi duyệt) \* khoảng cách các nút + khoảng cách nút/2 (dùng để tạo khoảng cách nhỏ so với viền canvas).

Ví dụ cây trên: Ta có 8 nút, và độ rộng của canvas là 800px, vậy khoảng cách giữa các node là: 800 / 8 = 100px. Nên ta có tọa độ (x) từng nút là:

* Nút có giá trị là 3 = index \* chiều rộng mỗi nút + chiều rộng mỗi nút/2

= 0 \* 100 + 100/2 = 50px => tọa độ x = 50

* Nút có giá trị là 7 = index \* chiều rộng mỗi nút + chiều rộng mỗi nút/2

= 1 \* 100 + 100/2 = 150px => tọa độ x = 150

* Và tương tự các nút khác

Thể hiện các nút lên tọa độ dọc (y)

Để ánh xạ các nút lên tọa độ dọc thì ta phải tìm được mức của các node sau đó lấy mức đó \* khoảng cách ta muốn (ví dụ 70px) + 50px.

Ví dụ cây trên: Mức 0 – nút 13, mức 1 – nút 7 và 15, ….

* Nút có giá trị là 13 = mức \* khoảng cách + 50

= 0 \* 70 + 50 = 50 => tọa độ y mức 0 = 50px

* Nút có giá trị là 7 và 15 = 1 \* 70 + 50 = 120 => tọa độ y mức 1 = 120px

|  |
| --- |
| inOrder(node, result = []) {          if (node !== null) {              this.inOrder(node.left, result);              result.push(node);              this.inOrder(node.right, result);          }          return result;      }      levelOrder() {          const levels = [];          if (this.root === null) return levels;          const queue = [{ node: this.root, level: 0 }];          while (queue.length > 0) {              const { node, level } = queue.shift();              if (!levels[level]) levels[level] = [];              levels[level].push(node);              if (node.left) queue.push({ node: node.left, level: level + 1 });              if (node.right) queue.push({ node: node.right, level: level + 1 });          }          return levels;      }  }  function drawTree(bst) {      const canvas = document.getElementById("treeCanvas");      const ctx = canvas.getContext("2d");      const canvasWidth = canvas.width;      // Ánh xạ vị trí ngang      const inOrderNodes = bst.inOrder(bst.root);      const horizontalSpacing = canvasWidth / inOrderNodes.length;      inOrderNodes.forEach((node, index) => {          node.x = index \* horizontalSpacing + horizontalSpacing / 2;      });      // Ánh xạ vị trí dọc      const levels = bst.levelOrder();      const verticalSpacing = 70;      levels.forEach((level, levelIndex) => {          level.forEach((node) => {              node.y = levelIndex \* verticalSpacing + 50;          });      });  } |

**CHƯƠNG 3: XÂY DỰNG CHƯƠNG TRÌNH**

**3.1 Công cụ và ngôn ngữ được sử dụng**

**3.1.1 Visual Studio Code (VS Code)**

Visual Studio Code là một mã nguồn mở và miễn phí, được phát triển bởi Microsoft. Đây là một trong những công cụ phổ biến nhất cho lập trình viên nhờ vào giao diện người dùng thân thiện và khả năng hỗ trợ nhiều ngôn ngữ lập trình.

**Tính năng nổi bật:**

* Hỗ trợ đa ngôn ngữ: VS Code hỗ trợ nhiều ngôn ngữ lập trình như Python, JavaScript, C++, Java, và nhiều ngôn ngữ khác thông qua các extension.
* Tính năng IntelliSense: Cung cấp các gợi ý thông minh về mã nguồn giúp lập trình viên viết mã nhanh chóng và chính xác hơn.
* Debugging: VS Code cung cấp các công cụ debugging mạnh mẽ giúp kiểm tra và sửa lỗi trong mã nguồn.
* Tích hợp Git: VS Code cho phép tích hợp với Git, giúp quản lý mã nguồn và phối hợp làm việc nhóm hiệu quả.
* Extension phong phú: Cung cấp một hệ sinh thái phong phú các extension giúp mở rộng tính năng của công cụ, đặc biệt là hỗ trợ các công cụ phân tích, kiểm thử, và phát triển dự án.

**3.1.2 Ngôn ngữ HTML, CSS**

HTML (HyperText Markup Language) là ngôn ngữ đánh dấu được sử dụng để xây dựng cấu trúc của các trang web, còn CSS (Cascading Style Sheets) được dùng để định dạng và trình bày giao diện của các trang web. Cả hai đều là nền tảng cơ bản cho việc phát triển các trang web hiện đại.

**HTML:**

* Cấu trúc trang web: HTML cung cấp các phần tử (elements) như tiêu đề, đoạn văn, hình ảnh, bảng, danh sách, v.v., để xây dựng bố cục trang web.
* Định nghĩa nội dung: HTML giúp phân loại và tổ chức nội dung trong trang web, như tiêu đề, đoạn văn, liên kết và các bảng.

**CSS:**

* Định dạng giao diện: CSS dùng để thay đổi màu sắc, phông chữ, kích thước, khoảng cách và các đặc điểm khác của các phần tử HTML.
* Tạo giao diện phản hồi: CSS hỗ trợ thiết kế trang web có thể thay đổi linh hoạt khi người dùng truy cập từ các thiết bị khác nhau (responsive design).
* Tăng tính thẩm mỹ: CSS giúp tạo ra các hiệu ứng hình ảnh đẹp mắt như chuyển động, bóng đổ, và các hiệu ứng hover.

**3.1.3 Ngôn ngữ lập trình JavaScript**

JavaScript là một ngôn ngữ lập trình phổ biến, động và thông dịch được sử dụng chủ yếu để tạo ra các trang web động và tương tác. Đây là ngôn ngữ quan trọng không thể thiếu trong việc phát triển các ứng dụng web hiện đại.

**Tính năng nổi bật:**

* Tương tác người dùng: JavaScript cho phép tạo ra các tính năng tương tác trong trang web như xử lý sự kiện (click, hover, nhập liệu), tạo các biểu mẫu động và thông báo.
* Xử lý dữ liệu: JavaScript có thể xử lý và thay đổi dữ liệu trên trang web mà không cần phải tải lại toàn bộ trang (AJAX, Fetch API).
* Chạy trên trình duyệt: JavaScript có thể chạy trực tiếp trên trình duyệt của người dùng, giúp giảm thiểu độ trễ và cải thiện trải nghiệm người dùng.
* Khả năng mở rộng: Với các thư viện và framework như React, Angular, và Vue.js, JavaScript trở thành công cụ mạnh mẽ cho phát triển ứng dụng web phức tạp.

**3.2 Đặc tả bài toán**

**3.2.1 Yêu cầu đề bài:**

Vận dụng kiến thức lập trình xử lý yêu cầu bài toán, hiển thị cây nhị phân tìm kiếm một cách trực quan.

* Tạo một cây nhị phân tìm kiếm từ một mảng giá trị được nhập từ bàn phím
* Thao tác với cây nhị phân tìm kiếm như thêm, xóa.
* Hiển thị trực quan cây nhị phân tìm kiếm

**3.2.2 Hướng dẫn xử lý:**

Thuật toán tạo và thao tác trên cây nhị phân tìm kiếm tương tự mục Tổng quan về thuật toán (2.2) đã trình bày. Sau đây là xử lý phần hiển thị cây một cách trực quan bằng Canvas.

Canvas là một phần tử trong HTML5 cho phép bạn vẽ đồ họa, tạo hoạt hình, hiển thị hình ảnh, hoặc thực hiện các tính toán đồ họa trực tiếp trên trang web. Nó không phải là một ngôn ngữ lập trình mà là một thẻ HTML (thẻ <canvas>) được sử dụng kết hợp với JavaScript để thao tác với đồ họa.

Thẻ <canvas> định nghĩa một khu vực mà bạn có thể vẽ các hình ảnh hoặc đồ họa, và mọi thao tác vẽ đều được xử lý bằng JavaScript thông qua API của Canvas.

**Tính năng và cách hoạt động của Canvas:**

* Vẽ đồ họa 2D: Canvas có thể được sử dụng để vẽ các hình học cơ bản như hình vuông, hình tròn, đường thẳng, hoặc hình ảnh phức tạp hơn.
* Hiển thị hình ảnh: Bạn có thể sử dụng canvas để hiển thị hình ảnh, áp dụng bộ lọc, thay đổi màu sắc, hoặc thực hiện các thao tác hình ảnh khác.
* Hoạt hình và chuyển động: Canvas hỗ trợ vẽ các đối tượng động, giúp bạn tạo ra các hiệu ứng hoạt hình mượt mà trên trang web mà không cần sử dụng Flash hay các plugin khác.
* Tạo game 2D: Canvas là một công cụ phổ biến để phát triển game 2D trực tiếp trong trình duyệt, vì nó hỗ trợ hiệu suất cao và tính linh hoạt trong việc vẽ đồ họa.

    // Vẽ các đường nối

    function drawLine(node1, node2) {

        ctx.beginPath();

        ctx.moveTo(node1.x, node1.y);

        ctx.lineTo(node2.x, node2.y);

        ctx.lineWidth = 2;

        ctx.strokeStyle = "black";

        ctx.stroke();

    }

    levels.forEach((level) => {

        level.forEach((node) => {

            if (node.left) drawLine(node, node.left);

            if (node.right) drawLine(node, node.right);

        });

    });

    // Vẽ các node

    function drawNode(node) {

        const radius = 22;

        ctx.beginPath();

        ctx.arc(node.x, node.y, radius, 0, 2 \* Math.PI);

        ctx.fillStyle = "white";

        ctx.fill();

        ctx.lineWidth = 4;

        ctx.strokeStyle = "#FFB800 ";

        ctx.stroke();

         // Hiệu ứng bóng

        ctx.shadowColor = "rgba(0, 0, 0, 0.3)";

        ctx.shadowBlur = 10;

        ctx.shadowOffsetX = 0;

        ctx.shadowOffsetY = 4;

        // Vẽ giá trị node

        ctx.shadowColor = "transparent";

        ctx.fillStyle = "#000000";

        ctx.font = "16px Arial";

        ctx.textAlign = "center";

        ctx.textBaseline = "middle";

        ctx.fillText(node.value, node.x, node.y);

    }

    inOrderNodes.forEach((node) => drawNode(node));

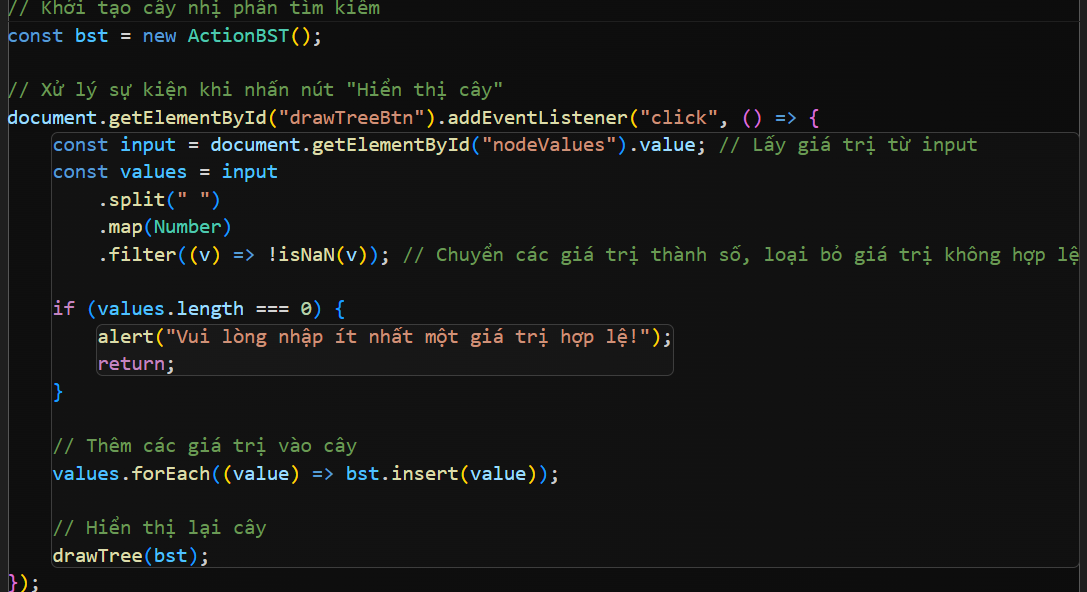
}

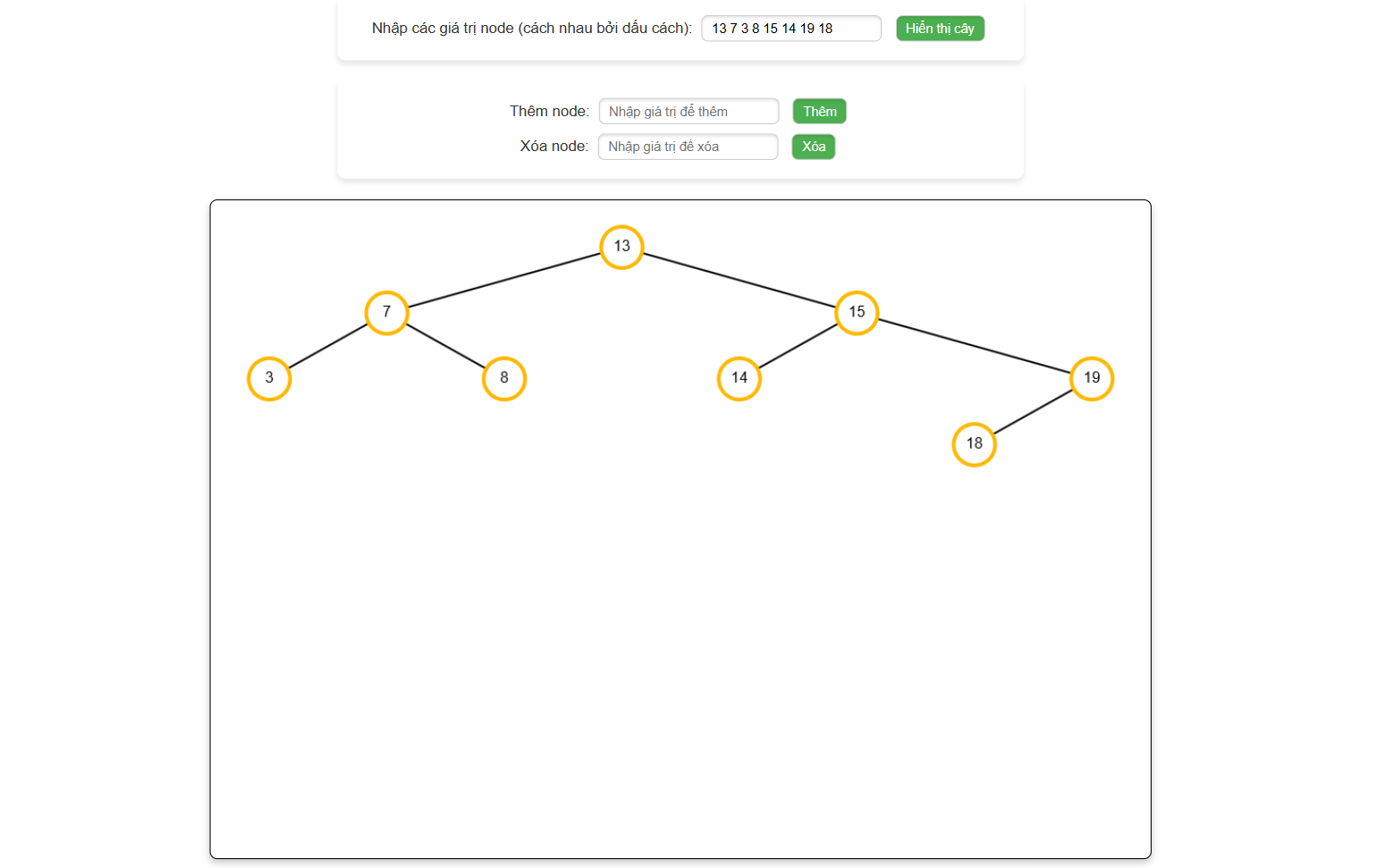
**3.3 Cài đặt chương trình**

**3.3.1 Mã nguồn chương trình và kết quả**

**3.3.1.1 Tạo cây nhị phân:**

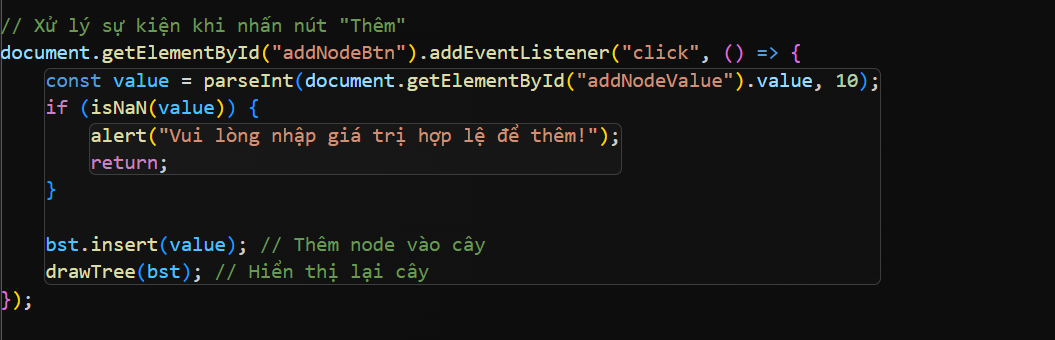
**Mã nguồn:**

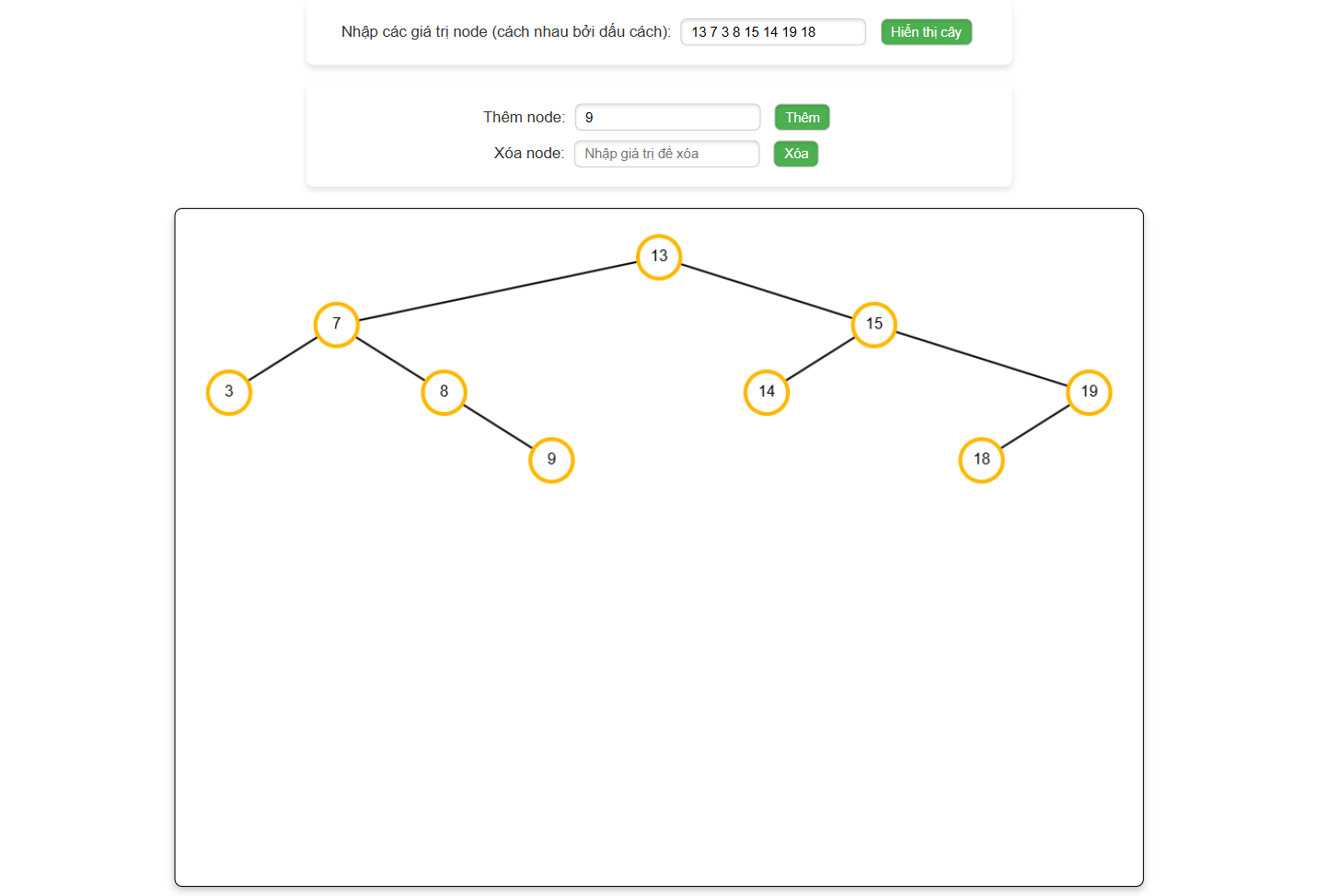
****

**Kết quả:  
**

**3.3.1.2 Thêm một nút:**

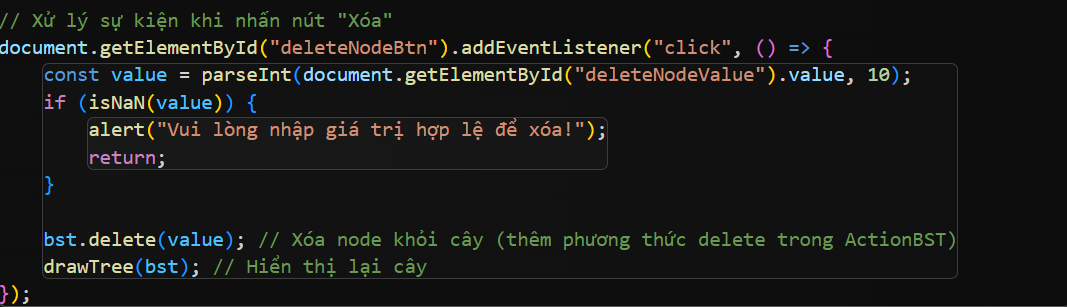
**Mã nguồn:**

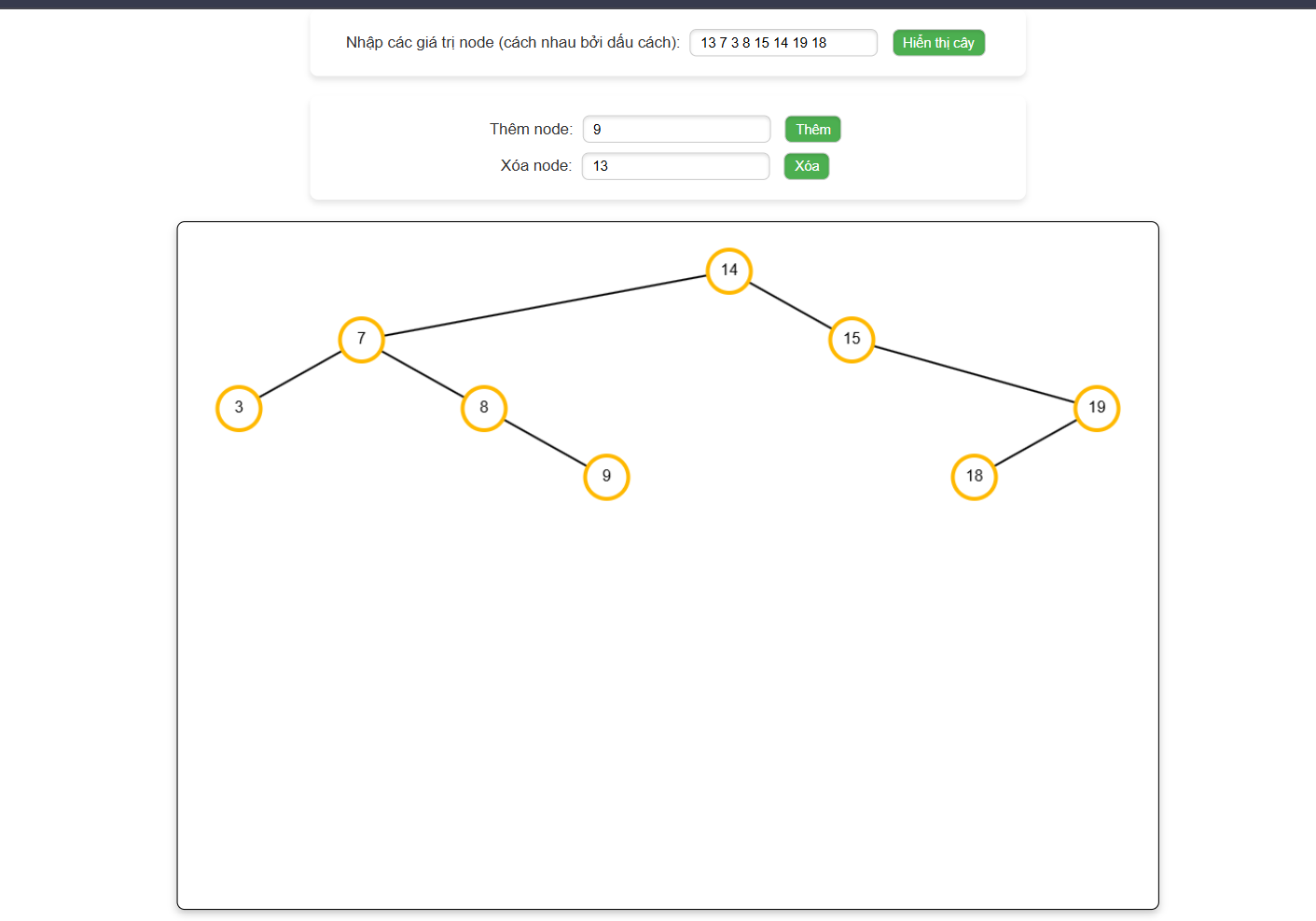
****

**Kết quả:  
**

**3.3.1.3 Xóa một nút:**

**Mã nguồn:**

**Kết quả:**

****

**KẾT LUẬN**

**1. Các kết quả đạt được**

Trong quá trình nghiên cứu và thực hiện đề tài "Thao tác trên cây nhị phân tìm kiếm", chúng ta đã xây dựng một công cụ trực quan mô phỏng các thao tác cơ bản trên cây nhị phân tìm kiếm (BST) như thêm, xóa, và duyệt cây. Công cụ này giúp người dùng không chỉ hiểu rõ hơn về cấu trúc cây nhị phân mà còn trực tiếp quan sát các thay đổi trong cây qua từng thao tác, giúp củng cố lý thuyết đã học và nâng cao khả năng lập trình.

Ứng dụng được phát triển bằng HTML, CSS và JavaScript, mang đến một giao diện thân thiện và dễ sử dụng. Mọi thao tác trên cây nhị phân như chèn và xóa nút đều được thực hiện mượt mà và dễ dàng, giúp người dùng thấy rõ sự thay đổi của cây thông qua giao diện đồ họa trực quan.

**2. Ưu và nhược điểm của đề tài**

**Ưu điểm:**

* Giúp nâng cao hiểu biết về cấu trúc cây nhị phân tìm kiếm (BST): Đề tài giúp người học hiểu rõ hơn về cách hoạt động của cây nhị phân tìm kiếm thông qua các thao tác cơ bản như chèn, xóa, và duyệt cây. Việc sử dụng công cụ trực quan giúp dễ dàng nhận diện các thay đổi trong cây và tạo ra sự kết nối lý thuyết với thực hành.
* Giao diện trực quan, dễ sử dụng: Với việc sử dụng HTML, CSS và JavaScript, ứng dụng cung cấp một giao diện đồ họa trực quan giúp người dùng theo dõi các thao tác trên cây nhị phân một cách sinh động. Các thao tác như thêm và xóa nút được thể hiện rõ ràng, giúp học viên dễ dàng nhận biết và tiếp thu.
* Ứng dụng đa nền tảng: Vì được phát triển bằng các công nghệ web, ứng dụng có thể chạy trên nhiều nền tảng khác nhau, từ máy tính để bàn đến các thiết bị di động, miễn là có trình duyệt web hỗ trợ.
* Khả năng mở rộng và nâng cấp: Mã nguồn của ứng dụng có thể dễ dàng mở rộng để hỗ trợ các thao tác phức tạp hơn hoặc cải tiến tính năng như thêm các loại cây khác (cây AVL, cây đỏ-đen) hoặc cải thiện giao diện người dùng.
* Học thực tế về lập trình đồ họa và tương tác người dùng: Việc sử dụng Canvas và JavaScript để vẽ đồ họa trực quan giúp học viên không chỉ nắm vững kiến thức về cây nhị phân mà còn có cơ hội tìm hiểu và làm quen với việc xử lý đồ họa và tương tác người dùng trong lập trình web.

**Nhược điểm:**

* Chưa có chức năng tìm kiếm nút theo giá trị và tạo cây từ một file có mảng dữ liệu trong đó.
* Giới hạn về hiệu suất khi xử lý cây lớn: Mặc dù cây nhị phân tìm kiếm có thể thực hiện thao tác tìm kiếm, chèn, và xóa với độ phức tạp O(log n) trong trường hợp cây cân bằng, nhưng với cây không cân bằng hoặc các trường hợp cực đoan, ứng dụng có thể gặp vấn đề hiệu suất, đặc biệt là khi số lượng nút trong cây quá lớn.
* Khó khăn trong việc duy trì cây cân bằng: Đề tài chỉ tập trung vào việc thao tác cơ bản với cây nhị phân tìm kiếm, mà không có cơ chế tự động cân bằng cây (như trong cây AVL hay cây đỏ-đen). Điều này có thể khiến hiệu suất của cây nhị phân tìm kiếm giảm khi có các thao tác liên tiếp mà không thực hiện cân bằng.
* Khả năng mở rộng hạn chế: Mặc dù đề tài có thể mở rộng thêm các chức năng, nhưng việc mở rộng để hỗ trợ nhiều loại cây nâng cao hay tối ưu hóa các thuật toán như cây AVL hoặc cây đỏ-đen sẽ đụng phải một số khó khăn về mặt thiết kế và hiệu suất.
* Phụ thuộc vào trình duyệt: Vì ứng dụng sử dụng Canvas và JavaScript, nó hoàn toàn phụ thuộc vào trình duyệt web. Nếu người dùng sử dụng các trình duyệt cũ hoặc không hỗ trợ đầy đủ các tính năng của HTML5, có thể gặp phải một số lỗi hoặc không hiển thị đúng.

**TÀI LIỆU THAM KHẢO**

1. Cây nhị phân tìm kiếm DSA – W3school, truy cập ngày 1 tháng 1 năm 2025, từ <https://www.w3schools.com/dsa/dsa_data_binarysearchtrees.php>
2. Sách cấu trúc dữ liệu và giải thuật – TS.GVC. Nguyễn Đức Thuần