Cây tìm kiếm nhị phân (Binary Search Tree)

Cây tìm kiếm nhị phân là gì ?

Một cây tìm kiếm nhị phân (Binary Search Tree – viết tắt là BST) là một cây mà trong đó tất cả các nút đều có các đặc điểm sau:

- Cây con bên trái của một nút có khóa (key) nhỏ hơn hoặc bằng giá trị khóa của nút cha (của cây con này).
- Cây con bên phải của một nút có khóa lớn hơn hoặc bằng giá trị khóa của nút cha (của cây con này).

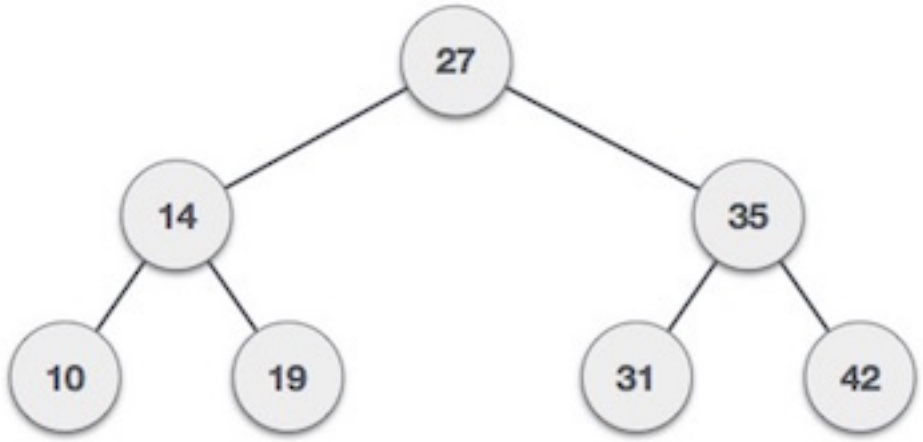
Vì thế có thể nói rằng, một cây tìm kiếm nhị phân (BST) phân chia tất cả các cây con của nó thành hai phần: *cây con bên trái* và *cây con bên phải* và có thể được định nghĩa như sau:

```
left_subtree (keys)  ≤  node (key)  ≤  right_subtree (keys)
```

Biểu diễn cây tìm kiếm nhị phân (BST)

Cây tìm kiếm nhị phân (BST) là một tập hợp bao gồm các nút được sắp xếp theo cách để chúng có thể duy trì hoặc tuân theo các đặc điểm của cây tìm kiếm nhị phân. Mỗi một nút thì đều có một khóa và giá trị liên kết với nó. Trong khi tìm kiếm, khóa cần tìm được so sánh với các khóa trong cây tìm kiếm nhị phân (BST) và nếu tìm thấy, giá trị liên kết sẽ được thu nhận.

Ví dụ một cây tìm kiếm nhị phân (BST):



Từ hình ví dụ minh họa trên ta thấy rằng, khóa của nút gốc có giá trị 27 và tất cả khóa bên trái của cây con bên trái đều có giá trị nhỏ hơn 27 này và tất cả các khóa bên phải của cây con bên phải đều có giá trị lớn hơn 27.

Hoạt động cơ bản trên cây tìm kiếm nhị phân

Dưới đây là một số hoạt động cơ bản có thể được thực hiện trên cây tìm kiếm nhị phân:

- **Hoạt động tìm kiếm**: tìm kiếm một phần tử trong một cây.
- **Hoạt động chèn**: chèn một phần tử vào trong một cây.
- **Hoạt động duyệt tiền thứ tự**: duyệt một cây theo cách thức duyệt tiền thứ tự.
- **Hoạt động duyệt trung thứ tự**: duyệt một cây theo cách thứ duyệt trung thứ tự.
- **Hoạt động duyệt hậu thứ tự**: duyệt một cây theo cách thức duyệt hậu thứ tự.

Nút (Node) trong cây tìm kiếm nhị phân

Một nút có một vài dữ liệu, tham chiếu tới các nút con bên trái và nút con bên phải của nó.

```
struct node {    int data;       struct node *leftChild;    struct node
*rightChild; };
```

Hoạt động tìm kiếm trong cây tìm kiếm nhị phân

Mỗi khi một phần tử được tìm kiếm: bắt đầu tìm kiếm từ nút gốc, sau đó nếu dữ liệu là nhỏ hơn giá trị khóa (key), thì sẽ tìm phần tử ở cây con bên trái; nếu lớn hơn thì sẽ tìm phần tử ở cây con bên phải. Dưới đây là giải thuật cho mỗi nút:

```
struct node* search(int data){    struct node *current = root;    printf("Truy
cap cac phan tu: ");          while(current->data != data){
if(current != NULL) {          printf("%d ",current->data);
current->rightChild;          }                                //không tìm thấy
if(current == NULL){             return NULL;          }       }                  }
```

Hoạt động chèn trong cây tìm kiếm nhị phân

Mỗi khi một phần tử được chèn: đầu tiên chúng ta cần xác định vị trí chính xác của phần tử này. Bắt đầu tìm kiếm từ nút gốc, sau đó nếu dữ liệu là nhỏ hơn giá trị khóa (key), thì tìm kiếm vị trí còn trống ở cây con bên trái và chèn dữ liệu vào đó; nếu dữ liệu là nhỏ hơn thì tìm kiếm vị trí còn sống ở cây con bên phải và chèn dữ liệu vào đó.


```
void insert(int data){    struct node *tempNode = (struct node*)
malloc(sizeof(struct node));    struct node *current;    struct node *parent;
tempNode->data = data;    tempNode->leftChild = NULL;    tempNode->rightChild =
NULL;     //Nếu cây là trống    if(root == NULL){       root = tempNode;
}else {       current = root;       parent = NULL;        while(1){
parent = current;                                    //tới cây con bên trái
if(data < parent->data){             current = current->leftChild;
//chèn dữ liệu vào cây con bên trái
if(current == NULL){                parent->leftChild = tempNode;
return;             }          }//tới cây con bên phải          else{
current = current->rightChild;             //chèn dữ liệu vào cây con bên phải
if(current == NULL){                parent->rightChild = tempNode;
return;             }          }       }               } }
```