

## Cây tìm kiếm nhị phân (BINARY SEARCH TREES)

Thích 0

Chia sẻ 0

Tweet

G+

0

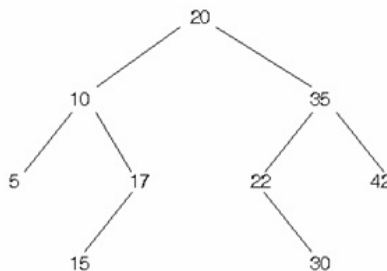
### CÂY TÌM KIẾM NHỊ PHÂN (BINARY SEARCH TREES)

#### Định nghĩa

Cây tìm kiếm nhị phân (TKNP) là cây nhị phân mà khoá tại mỗi nút cây lớn hơn khoá của tất cả các nút thuộc cây con bên trái và nhỏ hơn khoá của tất cả các nút thuộc cây con bên phải.

Lưu ý: dữ liệu lưu trữ tại mỗi nút có thể rất phức tạp như là một record chẳng hạn, trong trường hợp này khoá của nút được tính dựa trên một trường nào đó, ta gọi là trường khoá. Trường khoá phải chứa các giá trị có thể so sánh được, tức là nó phải lấy giá trị từ một tập hợp có thứ tự.

Ví dụ: hình III.15 minh hoạ một cây TKNP có khoá là số nguyên (với quan hệ thứ tự trong tập số nguyên).



Hình III.15: Ví dụ cây tìm kiếm nhị phân

**Qui ước:** Cũng như tất cả các cấu trúc khác, ta coi cây rỗng là cây TKNP

Nhận xét:

Trên cây TKNP không có hai nút cùng khoá.

Cây con của một cây TKNP là cây TKNP.

Khi duyệt trung tự (InOrder) cây TKNP ta được một dãy có thứ tự tăng. Chẳng hạn duyệt trung tự cây trên ta có dãy: 5, 10, 15, 17, 20, 22, 30, 35, 42.

#### Cài đặt cây tìm kiếm nhị phân

Cây TKNP, trước hết, là một cây nhị phân. Do đó ta có thể áp dụng các cách cài đặt như đã trình bày trong phần cây nhị phân. Sẽ không có sự khác biệt nào trong việc cài đặt cấu trúc dữ liệu cho cây TKNP so với cây nhị phân, nhưng tất nhiên, sẽ có sự khác biệt trong các giải thuật thao tác trên cây TKNP như tìm kiếm, thêm hoặc xoá một nút trên cây TKNP để luôn đảm bảo tính chất của cây TKNP.

Một cách cài đặt cây TKNP thường gặp là cài đặt bằng con trỏ. Mỗi nút của cây như là một mẫu tin (record) có ba trường: một trường chứa khoá, hai trường kia là hai con trỏ trỏ đến hai nút con (nếu nút con vắng mặt ta gán con trỏ bằng NIL)

Khai báo như sau

```
typedef <kiểu dữ liệu của khoá> KeyType;
```

```
typedef struct Node{KeyType Key;
```

```
Node* Left,Right;}
```

```
typedef Node* Tree;
```

Khởi tạo cây TKNP rỗng

Ta cho con trỏ quản lý nút gốc (Root) của cây bằng NIL.

```
void MakeNullTree(Tree *Root){
```

```
(*Root)=NULL;
```

```
}
```

Tìm kiếm một nút có khoá cho trước trên cây TKNP

Để tìm kiếm 1 nút có khoá x trên cây TKNP, ta tiến hành từ nút gốc bằng cách so sánh khoá của nút gốc với khoá x.

Nếu nút gốc bằng NULL thì không có khoá x trên cây.

Nếu x bằng khoá của nút gốc thì giải thuật dừng và ta đã tìm được nút chứa khoá x.

Nếu x lớn hơn khoá của nút gốc thì tiến hành (một cách đệ quy) việc tìm khoá x trên cây con bên phải.

Nếu x nhỏ hơn khoá của nút gốc thì tiến hành (một cách đệ quy) việc tìm khoá x trên cây con bên trái.

**Ví dụ:** tìm nút có khoá 30 trong cây ở trong hình III.15

- So sánh 30 với khoá nút gốc là 20, vì  $30 > 20$  vậy ta tìm tiếp trên cây con bên phải, tức là cây có nút gốc có khoá là 35.

- So sánh 30 với khoá của nút gốc là 35, vì  $30 < 35$  vậy ta tìm tiếp trên cây con bên trái, tức là cây có nút gốc có khoá là 22.
- So sánh 30 với khoá của nút gốc là 22, vì  $30 > 22$  vậy ta tìm tiếp trên cây con bên phải, tức là cây có nút gốc có khoá là 30.
- So sánh 30 với khoá nút gốc là 30,  $30 = 30$  vậy đến đây giải thuật dừng và ta tìm được nút chứa khoá cần tìm.

Hàm dưới đây trả về kết quả là con trỏ tới nút chứa khoá x hoặc NULL nếu không tìm thấy khoá x trên cây TKNP.

```
Tree Search(KeyType x, Tree Root){
    if (Root == NULL) return NULL; //không tìm thấy khoá x
    else if (Root->Key == x) /* tìm thấy khoá x */
        return Root;
    else if (Root->Key < x) //tìm tiếp trên cây bên phải
        return Search(x, Root->right);
    else
        {tìm tiếp trên cây bên trái}
        return Search(x, Root->left);
}
```

?

Cây tìm kiếm nhị phân được tổ chức như thế nào để quá trình tìm kiếm được hiệu quả nhất?

**Nhận xét:** giải thuật này sẽ rất hiệu quả về mặt thời gian nếu cây TKNP được tổ chức tốt, nghĩa là cây tương đối "cân bằng". Về chủ đề cây cân bằng các bạn có thể tham khảo thêm trong các tài liệu tham khảo của môn này.

Thêm một nút có khóa cho trước vào cây TKNP

Theo định nghĩa cây tìm kiếm nhị phân ta thấy trên cây tìm kiếm nhị phân không có hai nút có cùng một khoá. Do đó nếu ta muốn thêm một nút có khoá x vào cây TKNP thì trước hết ta phải tìm kiếm để xác định có nút nào chứa khoá x chưa. Nếu có thì giải thuật kết thúc (không làm gì cả!). Ngược lại, sẽ thêm một nút mới chứa khoá x này. Việc thêm một khoá vào cây TKNP là việc tìm kiếm và thêm một nút, tất nhiên, phải đảm bảo cấu trúc cây TKNP không bị phá vỡ. Giải thuật cụ thể như sau:

Ta tiến hành từ nút gốc bằng cách so sánh khóa của nút gốc với khoá x.

Nếu nút gốc bằng NULL thì khoá x chưa có trên cây, do đó ta thêm một nút mới chứa khoá x.

Nếu x bằng khoá của nút gốc thì giải thuật dừng, trường hợp này ta không thêm nút.

Nếu x lớn hơn khoá của nút gốc thì ta tiến hành (một cách đệ quy) giải thuật này trên cây con bên phải.

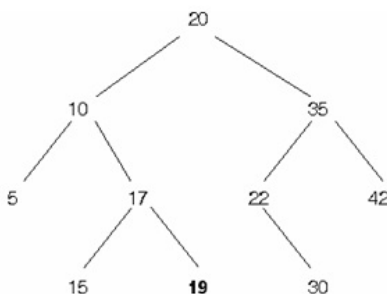
Nếu x nhỏ hơn khoá của nút gốc thì ta tiến hành (một cách đệ quy) giải thuật này trên cây con bên trái.

**Ví dụ:** thêm khoá 19 vào cây ở trong hình III.15

So sánh 19 với khoá của nút gốc là 20, vì  $19 < 20$  vậy ta xét tiếp đến cây bên trái, tức là cây có nút gốc có khoá là 10.

So sánh 19 với khoá của nút gốc là 10, vì  $19 > 10$  vậy ta xét tiếp đến cây bên phải, tức là cây có nút gốc có khoá là 17.

So sánh 19 với khoá của nút gốc là 17, vì  $19 > 17$  vậy ta xét tiếp đến cây bên phải. Nút con bên phải bằng NULL, chứng tỏ rằng khoá 19 chưa có trên cây, ta thêm nút mới chứa khoá 19 và nút mới này là con bên phải của nút có khoá là 17, xem hình III.16



Hình III.16: Thêm khoá 19 vào cây hình III.15

Thủ tục sau đây tiến hành việc thêm một khoá vào cây TKNP.

```
void InsertNode(KeyType x, Tree *Root ){
    if (*Root == NULL){ /* thêm nút mới chứa khoá x */
        (*Root)=(Node*)malloc(sizeof(Node));
        (*Root)->Key = x;
        (*Root)->left = NULL;
        (*Root)->right = NULL;
    }
    else
```

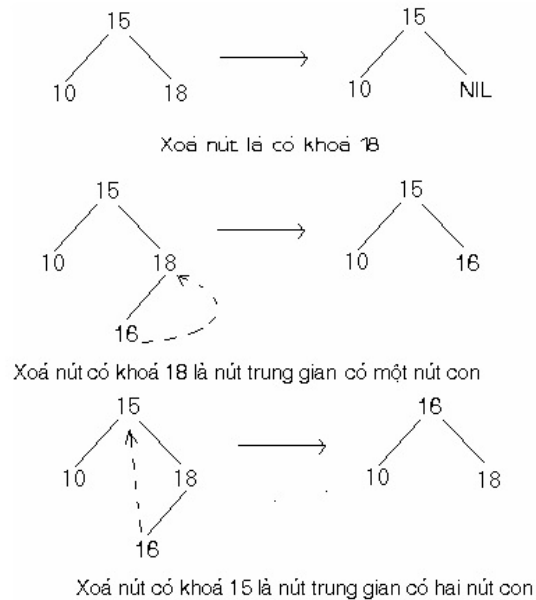
```

if (x < (*Root)->Key) InsertNode(x,Root->left);
else if (x > (*Root)->Key) InsertNode(x,Root->right);
}

```

Xóa một nút có khóa cho trước ra khỏi cây TKNP

Giả sử ta muốn xóa một nút có khóa x, trước hết ta phải tìm kiếm nút chứa khóa x trên cây. Việc xóa một nút như vậy, tất nhiên, ta phải bảo đảm cấu trúc cây TKNP không bị phá vỡ. Ta có các trường hợp như hình III.17:



Hình III.17 Ví dụ về giải thuật xóa nút trên cây

Nếu không tìm thấy nút chứa khóa x thì giải thuật kết thúc.

Nếu tìm gặp nút N có chứa khóa x, ta có ba trường hợp sau (xem hình III.17)

- Nếu N là lá ta thay nó bởi NULL.
- N chỉ có một nút con ta thay nó bởi nút con của nó.
- N có hai nút con ta thay nó bởi nút lớn nhất trên cây con trái của nó (nút cực phải của cây con trái) hoặc là nút bé nhất trên cây con phải của nó (nút cực trái của cây con phải). Trong giải thuật sau, ta thay x bởi khóa của nút cực trái của cây con bên phải rồi ta xóa nút cực trái này. Việc xóa nút cực trái của cây con bên phải sẽ rơi vào một trong hai trường hợp trên.

Giải thuật xóa một nút có khóa nhỏ nhất

Hàm dưới đây trả về khóa của nút cực trái, đồng thời xóa nút này.

```

KeyType DeleteMin (Tree *Root) {
    KeyType k;
    if ((*Root)->left == NULL) {
        k = (*Root)->key;
        (*Root) = (*Root)->right;
        return k;
    }
    else return DeleteMin(Root->left);
}

```

Thủ tục xóa một nút có khóa cho trước trên cây TKNP

```

void DeleteNode(key X, Tree *Root) {
    if ((*Root) != NULL)
        if (x < (*Root)->Key) DeleteNode(x, Root->left)
        else if (x > (*Root)->Key) DeleteNode(x, Root->right)
        else
            if ((*Root)->left == NULL && (*Root)->right == NULL)
                (*Root) = NULL;
            else

```

```
if ((*Root)->left == NULL) (*Root) = (*Root)->right
else
if ((*Root)->right==NULL) (*Root) = (*Root)->left
else (*Root)->Key = DeleteMin(Root->right);
}
```

**TỔNG KẾT CHƯƠNG**

Chương này giới thiệu một số khái niệm cơ bản về cây tổng quát, cây nhị phân và cây tìm kiếm nhị phân. Bên cạnh đó, chương này cũng đề cập đến cách lưu trữ cây trong bộ nhớ như cài đặt cây bằng mảng, con trỏ, danh sách các con, con trái nhất, anh em ruột phải và cách cài đặt các phép toán cơ bản trên các dạng cây khác nhau theo từng cách cài đặt.

Thích 0

Chia sẻ 0

Tweet

 0

0 bình luận

Sắp xếp theo 

Cũ nhất




Thêm bình luận...

[Plugin bình luận của Facebook](#)

TÀI VỀ

TÀI SỬ DỤNG(/user/reuse/m/85442db8/1)



unknown (/profile/3)

0 GIÁO TRÌNH (/PROFILE/3?TYPES=2) | 1060 TÀI LIỆU (/PROFILE/3?TYPES=1)

(/profile/3)

**ĐÁNH GIÁ:**

0 dựa trên 0 đánh giá

Tuyển tập sử dụng module này

- Cấu Trúc Dữ Liệu (/c/cau-truc-du-lieu/3bed1b9f)
- Cấu Trúc Dữ Liệu (/c/cau-truc-du-lieu/c5e6dc01)
- Cấu Trúc Dữ Liệu (/c/cau-truc-du-lieu/faed1c05)

NỘI DUNG CÙNG TÁC GIẢ

- Hàm và Script file (/m/ham-va-script-file/e984717a)
- Bài khí (/m/bai-khi/eaae03df)
- Mục tiêu của quá trình tiết trùng (/m/muc-tieu-cua-qua-trinh-tiet-trung/bff925a5)
- Vị trí, đối tượng, phương pháp và chức năng của chủ nghĩa xã hội khoa học (/m/vi-tri-doi-tuong-phuong-phap-va-chuc-nang-cua-chu-nghia-xa-hoi-khoa-hoc/c9a76cb8)
- Trình tự, nội dung lập quy hoạch sử dụng đất chi tiết và kế hoạch sử dụng đất chi tiết kỳ đầu (/m/trinh-tu-noi-dung-lap-quy-hoach-su-dung-dat-chi-tiet-va-ke-hoach-su-dung-dat-chi-tiet-ky-dau/e3ebd4f7)
- Khái niệm về các chất dinh dưỡng và thành phần lương thực thực phẩm (/m/khai-niem-ve-cac-chat-dinh-duong-va-thanh-phan-luong-thuc-thuc-pham/b6536f36)
- Vai trò của đạo đức mới trong việc xây dựng chủ nghĩa xã hội (/m/vai-tro-cua-dao-duc-moi-trong-viec-xay-dung-chu-nghia-xa-hoi/71036e56)
- Nguyên lý đánh bắt lưới kéo (/m/nguyen-ly-danh-bat-luoi-keo/b06be6f8)
- Các hệ vi sinh vật trong đồ hộp (/m/cac-he-vi-sinh-vat-trong-do-hop/693261f4)
- Phương pháp mọc môi và kỹ thuật cấy (/m/phuong-phap-moc-moi-va-ky-thuat-cau/cd61c2f8)

TRƯỚC

TIẾP

NỘI DUNG TƯƠNG TỰ

- Cây nhị phân và ứng dụng~ (/m/cay-nhi-phan-va-ung-dung/58472201)
- Cây tìm kiếm nhị phân (/m/cay-tim-kiem-nhi-phan/be1667a8)
- Mở đầu về thiết kế, đánh giá thuật toán và kiến thức bổ trợ (/m/mo-dau-ve-thiet-ke-danh-gia-thuat-toan-va-kien-thuc-bo-tro/f5007ea2)