درخت جست‌وجوی دودویی

تعریف

درخت جستجوی دودویی، یک درخت دودویی ریشه‌دار (به این معنی که هر رأس حداکثر دو بچه دارد) با این ویژگی است که مقدار هر رأس، از تمام مقادیر زیر درخت چپ بزرگ‌تر و از تمام مقادیر زیر درخت راست خود کوچک‌تر است. این داده‌ساختار برای داده‌های مقایسه‌پذیر قابل استفاده است و همان‌طور که می‌توان حدس زد برای الگوریتم‌هایی مثل مرتب‌سازی و جستجو بر مبنای مقایسه بسیار پرکاربرد است.

ترتیب داده‌ها

شکل درختی که پس از افزودن مجموعه‌ای از داده‌ها تشکیل می‌شود، کاملاً بستگی به ترتیب افزوده شدن آن‌ها دارد. اما فارغ از شکل درخت، نمایش میان‌ترتیب عناصر درخت، نمایش مرتب آنان است.

ارتفاع درخت جستجوی دودویی حداقل $lg(n)$ است و بسته به ترتیب افزوده شدن داده‌ها ممکن است تا n زیاد شود. امّا ارتفاع متوسط درخت‌های جستجوی دودویی از $O(lgn)$ می‌باشد. انواع خاصی از درخت‌های جست‌وجوی دودویی، مثل درخت قرمز-سیاه تضمین می‌کنند که طول درخت همواره از $O(lgn)$ باشد.

پیاده‌سازی

درخت جستجوی دودویی عملیات‌های درج، حذف و جست‌وجو را در زمانی از مرتبه ارتفاع درخت انجام می‌دهد.

```
void insert(int x) {
    if (root == NULL) {
        root = new node; root-> val = x;
        return;
    }
    node *cur = root;
    while (cur-> val != x)
        if (x < cur-> val) {
            if (cur-> l == NULL) {
                cur-> l = new node; cur-> l-> val = x;
            }
            cur = cur-> l;
        }
        else {
            if (cur-> r == NULL) {
                cur-> r = new node; cur-> r-> val = x;
            }
            cur = cur-> r;
        }
}
void remove(int x) {
    node *cur = root, *par = NULL;
    while (cur-> val != x) {
        par = cur;
        cur = (x < cur-> val? cur-> l: cur-> r);
    }
    node *&rel = (cur == root? root: (cur == par-> l? par-> l: par-> r));
    if (!cur-> l && !cur-> r)
        rel = NULL;
    else if (!cur-> l)
        rel = cur-> r;
    else if (!cur-> r)
        rel = cur-> l;
```

```
    else {
        int y = successor(cur)-> val;
remove(y);
cur-> val = y;
}
}
bool search(int x) {
    node *cur = root;
    while (cur != NULL && cur-> val != x)
        cur = (x < cur-> val? cur-> l: cur-> r);
    return cur != NULL;
}
```