

CÁC CHIẾN LƯỢC TÌM KIẾM



Mục tiêu

- Hiểu rõ tư tưởng của hai phương pháp tìm kiếm: tìm kiếm tuần tự và tìm kiếm nhị phân
- Nắm vững và cài đặt được tư tưởng của hai phương pháp tìm kiếm
- Ứng dụng của hai phương pháp tìm kiếm vào các bài toán cụ thể

Nội dung

- Bài toán tìm kiếm
- Thuật toán tìm kiếm tuần tự
- Thuật toán tìm kiếm nhị phân
- Cây tìm kiếm nhị phân

Bài toán tìm kiếm

- Tập đối tượng cho trước có thể có nhiều kiểu dữ liệu khác nhau. Thông thường yêu cầu tìm kiếm chỉ căn cứ một hoặc một vài thành phần (trường). Các thành phần đó gọi là *trường khóa tìm kiếm*.
- Quá trình tìm kiếm thường gồm 02 bước:
 - **1:** Dựa vào giá trị trường khóa và khóa để xác định đối tượng có giá trị trường khóa bằng với khóa tìm kiếm hoặc khẳng định không tìm thấy đối tượng cần tìm;
 - **2:** Kết xuất toàn bộ thông tin về đối tượng tìm được.

Bài toán tìm kiếm

- Ta chỉ xét bước 1 cho bài toán: Cho một dãy gồm n đối tượng, mỗi đối tượng i ($1 \leq i \leq n$) tương ứng với một khóa $k[i]$. Hãy tìm đối tượng có giá trị khóa bằng x cho trước.
 - Tìm được đối tượng i có $k[i] = x$; tìm kiếm thành công;
 - Không có đối tượng nào có khóa bằng x ; tìm kiếm thất bại.

Tìm kiếm tuần tự

- **Input:** Dãy khóa gồm N số nguyên k_1, k_2, \dots, k_N đôi một khác nhau và số nguyên x ;
- **Output:** Chỉ số i mà $k_i = x$ hoặc thông báo không có số hạng nào của dãy A có giá trị trùng với khóa x .

Tìm kiếm tuần tự

- *Ý tưởng*: Lần lượt từ số hạng thứ nhất, so sánh với khoá tìm kiếm x cho đến khi có sự trùng nhau. Nếu đã xét tới (k_N) mà không xảy ra sự trùng nhau thì dãy khoa không chứa giá trị x tìm kiếm.

Tìm kiếm tuần tự

▪ Thuật toán

Bước 1. Nhập N , các giá trị k_1, k_2, \dots, k_N và giá trị x .

Bước 2. $i \leftarrow 1$.

Bước 3. Nếu $k_i = x$ thì thông báo chỉ số i , rồi kết thúc.

Bước 4. $i \leftarrow i + 1$

Bước 5. Nếu $i > N$ thì Thông báo dãy khoa không có số hạng nào có giá trị trùng với x , rồi kết thúc.

Bước 6. Quay lại bước 3.

Tìm kiếm tuần tự

- Nhận xét và đánh giá.
 - Phép toán tích cực là phép so sánh:
 - Trường hợp tốt nhất độ phức tạp là $O(1)$
 - Trường hợp xấu nhất và trung bình độ phức tạp là $O(N)$

Tìm kiếm nhị phân

- **Input:** Dãy gồm N số nguyên k_1, k_2, \dots, k_N đôi một khác nhau và là dãy tăng; số nguyên x .
- **Output:** Chỉ số i mà $k_i = x$ hoặc thông báo không có số hạng nào của dãy có giá trị trùng với x .

Tìm kiếm nhị phân

- *Ý tưởng*: Sử dụng dãy đã sắp xếp ta tìm cách thu hẹp phạm vi tìm kiếm sau mỗi lần so sánh khóa với số hạng được chọn.

K_{Giua}

$$Giua = \lfloor (N+1)/2 \rfloor.$$

Tìm kiếm nhị phân

- Nếu $k_{Giua} = x$ thì Giua là chỉ số cần tìm.
- Nếu $k_{Giua} > x$ thì việc tìm kiếm tiếp theo chỉ xét trên dãy $k_1, k_2, \dots, k_{Giua-1}$
- Nếu $k_{Giua} < x$ thì thực hiện tìm kiếm trên dãy $k_{Giua+1}, k_{Giua+2}, \dots, k_N$.
- Quá trình trên sẽ được lặp lại

Tìm kiếm nhị phân

▪ Thuật toán

Bước 1. Nhập N , các giá trị k_1, k_2, \dots, k_N và giá trị khóa x .

Bước 2. $Dau \leftarrow 1, Cuoi \leftarrow N$

Bước 3 $Giua \leftarrow (Dau + Cuoi) / 2$.

Bước 4. Nếu $k_{Giua} = x$ thì thông báo chỉ số $Giua$, rồi kết thúc

Bước 5. Nếu $k_{Giua} > x$ thì đặt $Cuoi = Giua - 1$ rồi chuyển đến bước 7.

Bước 6. Nếu $k_{Giua} < x$ thì đặt $Dau = Giua + 1$ rồi chuyển đến bước 7.

Bước 7. Nếu $Dau > Cuoi$ thì thông báo dãy không có số hạng có giá trị trùng với x , rồi kết thúc.

Bước 8. Quay lại bước 3.

Tìm kiếm nhị phân

- *Nhận xét và đánh giá*
 - Trường hợp tốt nhất, độ phức tạp tính toán là $O(1)$, trong trường hợp xấu và trung bình là $O(\lg N)$,
 - Tuy nhiên với dãy chưa được sắp xếp thì cần có chi phí cho việc sắp xếp.

Cây tìm kiếm nhị phân

Định nghĩa

- Cây nhị phân
- Giá trị của các nút của cây con trái nhỏ hơn giá trị của nút gốc
- Giá trị của các nút cây con phải lớn hơn giá trị của nút gốc

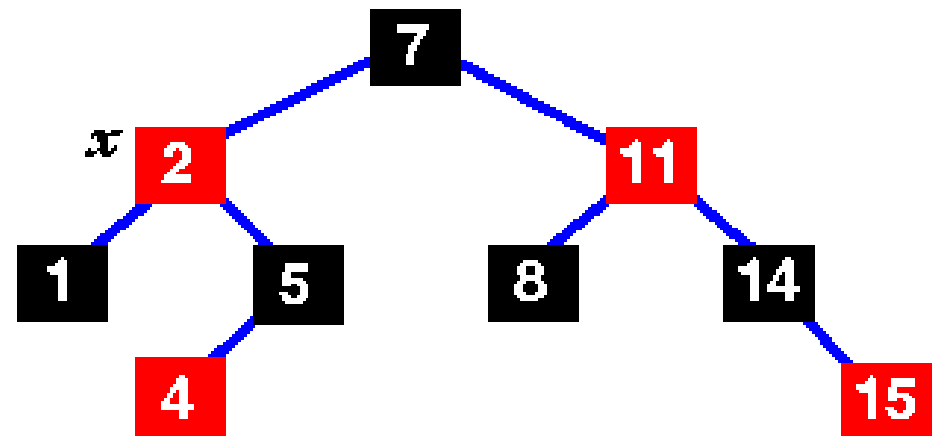
Cây tìm kiếm nhị phân (1)

```
struct Node{  
    Item data;  
    Node *left,*right;  
} *root;
```



Cây tìm kiếm nhị phân (2)

- Cây con trái và cây con phải cũng là cây tìm kiếm nhị phân.
- Hai giá trị khóa chứa trong hai nút bất kì khác nhau là khác nhau.



Cây tìm kiếm nhị phân (3)

Phép tìm kiếm:

Có nút nào chứa giá trị bằng khóa?

- Đối sánh giá trị nút đó với khóa:
- Bằng nhau thì ta biết nút có giá trị bằng khóa. Kết thúc
- Nhỏ hơn thực hiện trên cây con trái
- Lớn hơn phải thực hiện trên cây con phải



Cây tìm kiếm nhị phân (4)

Phép chèn

- Xin cấp bộ nhớ cho một nút mới,
- Gán giá trị mới vào trường Data .
- Con trỏ Left và Right gán giá trị đặc biệt Null.

Cây tìm kiếm nhị phân (5)

Từ gốc, đối sánh với giá trị mới cần thêm vào:

- Nếu bằng nhau  kết thúc
- Ngược lại  duyệt theo cây con trái (nếu $<$) hoặc cây con phải (nếu $>$)

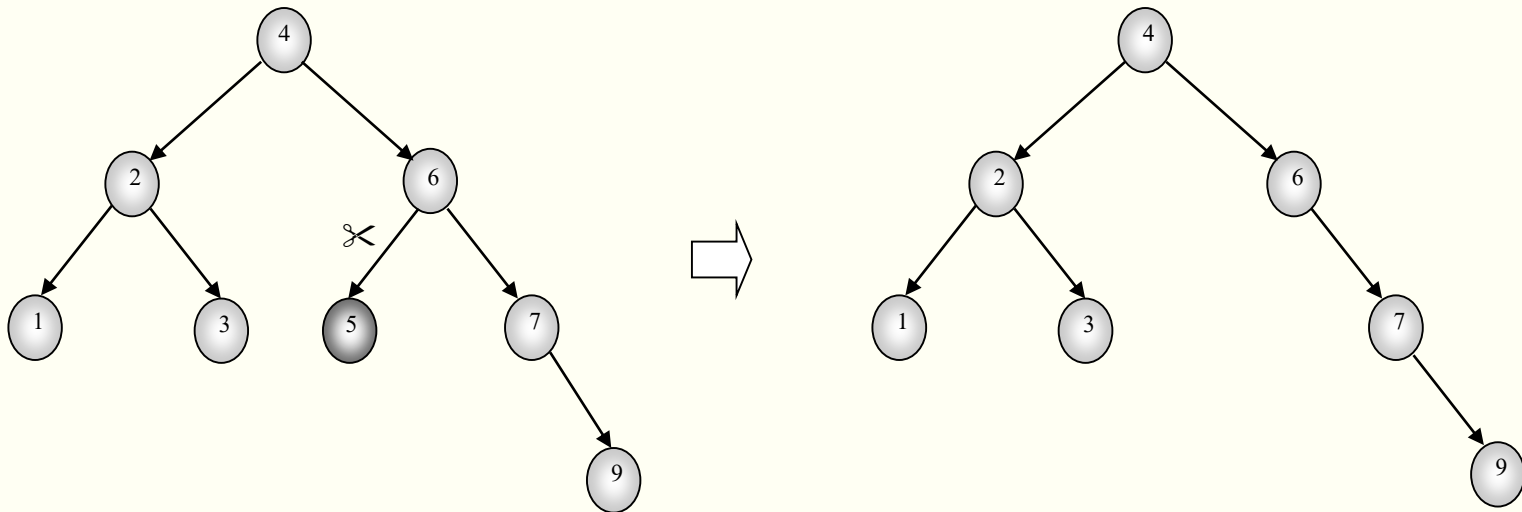
Cây tìm kiếm nhị phân (6)

- Khi gặp nút x không có nút con trái/phải xác định được vị trí cần chèn.
- Chỉnh sửa các trường liên kết để con trở trái/phải của nút x trở tới nút mới.

Cây tìm kiếm nhị phân (7)

d) Phép xóa

- Nếu nút x là nút lá ta chỉ cần “cắt bỏ” x



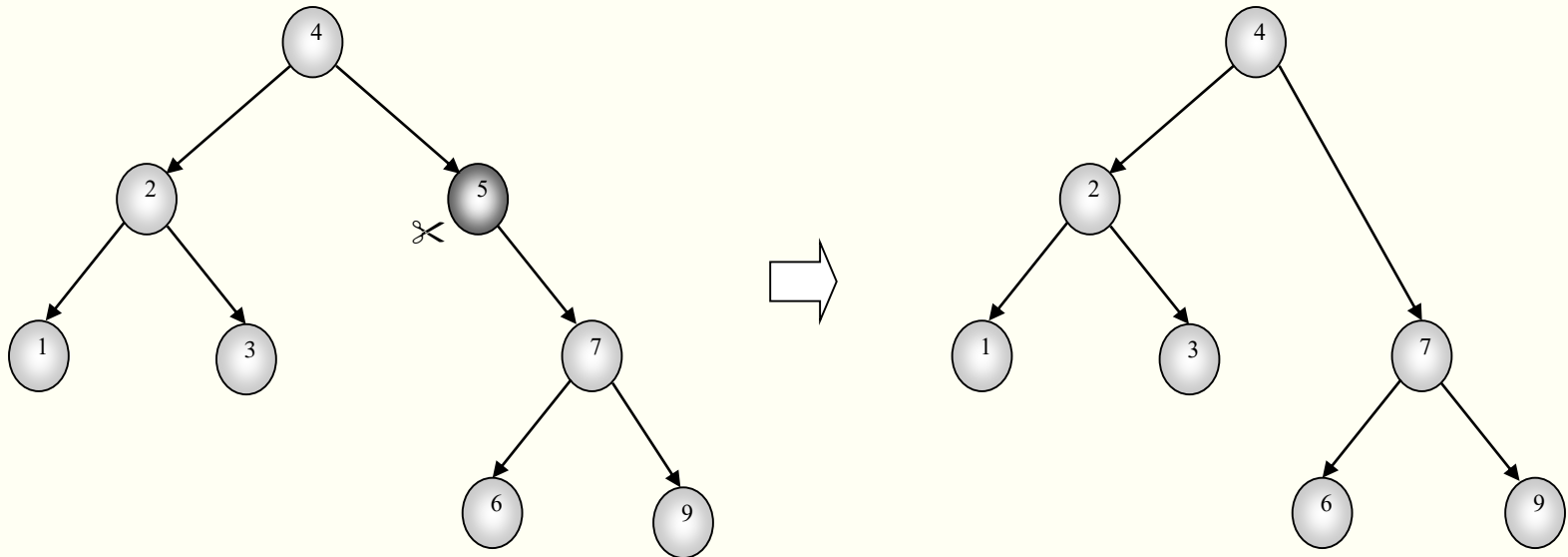
Cây tìm kiếm nhị phân (8)

d) Phép xóa

- Nếu x có một trong hai cây con là cây rỗng thì điều chỉnh lại con trở của nút cha của x

Cây tìm kiếm nhị phân (9)

Ví dụ



Cây tìm kiếm nhị phân (10)

d) Phép xóa

- Nếu nút x có hai cây con (có gốc tương ứng là x_1 và x_2) và một trong hai cây con (chẳng hạn x_1) không có con thì ta thay nút x_1 làm nút cha của cây con có gốc là x_2 .

Cây tìm kiếm nhị phân (11)

- Trường hợp tổng quát:
 1. Thay giá trị của x bằng giá trị nút lá bên phải cùng của cây con trái (hoặc bằng giá trị nút lá bên trái cùng của cây con phải)
 2. Cắt bỏ nút có giá trị vừa gán cho nút x .

Cây tìm kiếm nhị phân (12)

e) Đánh giá thời gian thực hiện các phép toán trên cây tìm kiếm nhị phân

- Với phép tìm kiếm.

Phép toán tích cực: phép so sánh giá trị khóa x cho trước với các giá trị các nút nằm trên đường đi từ gốc đến đỉnh xảy ra quan hệ bằng.

Coi thời gian thực một phép toán so sánh = thời gian đi từ một đỉnh đến đỉnh con của nó \rightarrow thời gian thực hiện tìm kiếm có thể coi là thời gian đi trên đường đi từ gốc đến một đỉnh nào đó.

Cây tìm kiếm nhị phân (13)

Trường hợp với cây nhị phân đầy đủ thì độ cao của cây là xấp xỉ $\log n$. Giả sử mức thấp nhất là k ta có:

$$1 + 2 + 2^2 + \dots + 2^{k-1} < n$$

$$1 + 2 + 2^2 + \dots + 2^k \geq n$$

Hay $2^k - 1 < n$ và $2^{k+1} - 1 \geq n$, do vậy $\log(n+1) - 1 \leq k < \log(n+1)$, nghĩa là k xấp xỉ $\log n$. Vì thế độ phức tạp thuật toán là $O(\log n)$

Trong trường hợp xấu nhất, nghĩa là cây suy biến lệch trái hoặc lệch phải thì độ phức tạp thuật toán là $O(n)$

- Với các phép toán chèn và xóa cũng có cách đánh giá tương tự.

Tổng kết

- Một số chiến lược tìm kiếm
 - Tuần tự
 - Nhị phân
 - Cây tìm kiếm nhị phân