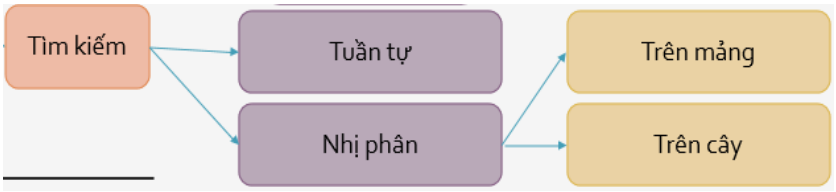
Mục lục:

1. Tìm kiếm tuần tự 2
2. Tìm kiếm nhị phân 3
3. Cây nhị phân tìm kiếm. 5

Đề cương cấu trúc dữ liệu và giải thuật

Chương 7: Các giải thuật tìm kiếm



- *Phát biểu bài toán tìm kiếm:* Cho một bảng gồm n bản ghi r_1, r_2, …, r_n; r_i tương ứng với 1 khóa k_i. Hãy tìm bản ghi có giá trị khóa tương ứng bằng x cho trước.
- *Ta thấy rõ ràng như sau:*

Gọi x là khóa tìm kiếm hay giá trị tìm kiếm thì công việc tìm kiếm sẽ hòa thành khi có 1 trong 2 trường hợp sau xảy ra:

Tìm được bản ghi có giá trị khóa tương ứng bằng x. Lúc đó ta nói phép tìm kiếm được thỏa mãn.

Nếu không tìm được bản ghi có giá trị khóa tương ứng bằng x thì ta nói phép tìm kiếm không thỏa mãn. Trong trường hợp này có thêm yêu cầu bổ sung bản ghi mới có khóa vào bảng thì giải thuật này được gọi là tìm kiếm có bổ sung.

1. Tìm kiếm tuần tự

- *Ý tưởng:*

Bắt đầu tìm từ bản ghi thứ nhất, lần lượt so sánh khóa tìm kiếm tương ứng của các bản ghi trong bảng cho đến khi tìm thấy bản ghi mong muốn hoặc đã hết danh sách mà chưa tìm thấy

- *Giả mã:*
  + Vào: Dãy khóa k, số lượng phần tử n, giá trị tìm kiếm x.
  + khóa bằng x thì đưa ra thứ tự của khóa đó, nếu không thì đưa ra giá trị 0.

{Hàm nhày nhận vào dãy khóa tìm kiếm k, số lượng phần tử n và giá trị tìm kiếm x thông qua đối số. Thực hiện tìm kiếm xem có khóa nào bằng x, nếu có thì đưa ra thứ tự của khóa đó, nếu không thì đưa ra giá trị 0}

```
Function SequenceSearch(k, n, x)
    1){Khởi đầu}
        i:=1; k[n+1]:=x;
    2){Tìm kiếm trong dãy}
        While k[i]≠x do i:=i+1;
    3){Không tìm thấy}
        If i=n+1 then return 0
        Else return i;
Return
```

- *Đánh giá giải thuật*: O(n).

2. Tìm kiếm nhị phân

- *Ý tưởng:*

Phương pháp tìm kiếm thực hiện trên dãy đã sắp xếp, có nội dung như sau:

Tương tự như tra tìm trong từ điển hoặc danh bạ điện thoại. Chỉ là trong tra cứu ta chọn từ ngẫy nhiên, còn trong tìm kiếm nhị phân luôn chọn khóa ở giữa dãy khóa.

Giả sử có dãy khóa k_L, …, k_R thì khóa ở giữa là k_i với i=(L+R) div 2. Tìm kiếm sẽ kết thúc nếu $x=k_i$.

Nếu $x<k_i$ tìm kiếm sẽ thực hiện tiếp tục với k_L, …, k_{i-1} với cách tương tự.

Nếu $x>k_i$ tìm kiếm sẽ thực hiện tiếp tục với k_{i+1}, …, k_R với cách tương tự.

Quá trình tìm kiếm kết thúc khi tìm thấy một khóa mong muốn hoặc dãy khóa rỗng.

- *Giả mã:*
  + Vào: Dãy khóa k (đã sắp xếp theo thứ tự tăng dần), số lượng phần tử n, giá trị tìm kiếm x.
  + Ra: Nếu tìm thấy thì cho ra chỉ số của khóa, nếu không thì cho ra 0.

{Hàm này nhận vào dãy khóa k đã sắp xếp theo thứ tự tăng dần gồm n phần tử và giá trị tìm kiếm x thông qua đối số. Thực hiện tìm kiếm nhị phân, nếu tìm thấy thì cho ra chỉ số của khóa, nếu không thì cho ra 0.}

```
Function BinarySearch(k, n, x)
    1){Khởi tạo}
        L:=1; R:=n;
    2){Tìm kiếm}
        While L<=R do
            Begin
                3){Tính chỉ số giữa}
                    M:=(L+R) div 2;
                4){So sánh}
                    If x<k[m] then R:=m-1
                    Else If x>k[m] then L:=m+1
                    Else return m;
            End;
    5){Không tìm thấy}
```

```
        Return 0;
Return
```

*) Nếu viết dưới dạng đệ quy thì:

```
Function BinarySearch(L, R, x)
    1)If L>R then Loc:=0
    Else
        Begin
            If x<k[m] then Loc:=BinarySearch(L, m-1, x)
            Else If x>k[m] then Loc:=BinarySearch(m+1, R, x)
            Else Loc:=m;
        End
    2)Return Loc;
Return
```

- *Đánh giá giải thuật*: $O(\log_2 n)$

3. Cây nhị phân tìm kiếm.

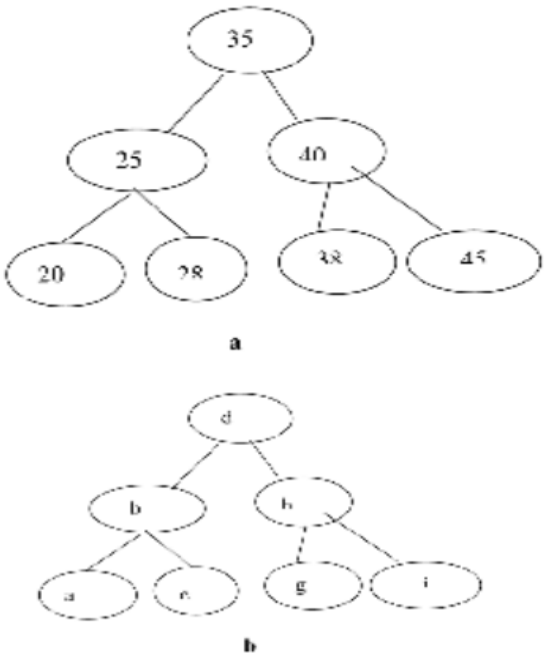
- *Định nghĩa cây nhị phân tìm kiếm:*

Cây nhị phân tìm kiếm ứng với n khóa $k_1, k_2, \ldots, k_n$ là một cây nhị phân mà mỗi nút của nó đều được định danh bởi một khóa nào đó trong các khóa đã cho. Đối với mọi nút trên cây tính chất sau đây luôn được thỏa mãn:

- Mọi khóa thuộc cây con trái của một nút đều nhỏ hơn khóa ứng với nút đó.
- Mọi khóa thuộc cây con phải của một nút đều lớn hơn khóa ứng với nút đó.

Chú ý: Khóa là số thì so sánh số bình thường, còn nếu khóa là chữ thì ta so sánh xâu nhị ký tự.

Ví dụ có các cây nhị phân tìm kiếm sau:



- *Ý tưởng:*
  - Đối với một cây nhị phân để tìm kiếm xem một khóa x nào đó có trên cây đó không? Ta có thể thực hiện như sau:

So sánh x với khóa ở gốc và một trong 4 tình huống sau đây sẽ xuất hiện:

i. *Không có gốc cây (cây rỗng):* x không có trên cây, phép tìm kiếm không thỏa mãn.
ii. *x trùng với khóa ở gốc*: Phép tìm kiếm được thỏa mãn.
iii. *x nhỏ hơn khóa ở gốc*: Tìm kiếm được thực hiện tiếp tục bằng cách xét cây con trái của gốc với cách làm tương tự.
iv. *x lớn hơn khóa ở gốc*: Tìm kiếm được thực hiện tiếp tục bằng cách xét cây con phải của gốc với cách làm tương tự.

Nếu tìm x=m trên cây b thì phép tìm kiếm không thỏa mãn.

- Nếu phép tìm kiếm không thỏa mãn ta bổ sung luôn x vào cây nhị phân tìm kiếm. Phép bổ sung không ảnh hưởng đến cây, đến vị trí các khóa trên cây.
- Mỗi nút của cây nhị ohaan tìm kiếm có cấu trúc như sau:

Left	Key	Infor	Right

Trong đó:

Left: Con trỏ trỏ tới gốc con trái.

Right: Con trỏ trỏ tới gốc cây con phải.

Key: Khóa của nút.

Infor: Thông tin

- Giải thuật tìm kiếm trên cây nhị phân tìm kiếm như sau:

Cho cây nhị phân tìm kiếm có gốc được trỏ bởi T. Tìm nút trên cây có khóa bằng x.

Nếu tìm kiếm được thì con trỏ trỏ tới nút đó, giá trị trả về là q, đó là địa chỉ của nút ta tìm được (con trỏ q trỏ tới nút đó).

Nếu không tìm được ta bổ sung x vào cây T, con trỏ q trỏ tới nút mới đó.

- *Giả mã:*

```
Function BST(T, x)
    1){Khởi tạo con trỏ}
        p:=NULL; q:=T;
    2){Tìm kiếm}
        While q≠ NULL do
            Case
                x<Key(q): p:=q; q:=Left(q);
```

x=Key(q): return q;

x>Key(q): p:=q; q:=Right(q)

End Case

3){Bổ sung}

q ⇐ AVAIL;

Key(q):=x;

Left(q):=Right(q):=NULL;

Case

T=NULL: T:=q;

x<Key(p): Left(p):=q;

else: Right(p):=q;

End Case

Return q;

Return

Câu hỏi ôn tập cuối chương:

Câu 33: Nêu khái niệm cây nhị phân tìm kiếm. Viết giải thuật xây dựng cây nhị phân tìm kiếm cho dãy khóa có n phần tử. Ứng dụng tạo cây con cho dãy khóa … bằng giải thuật BST.