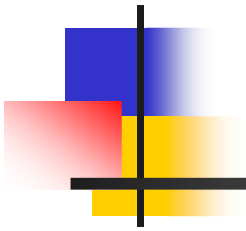
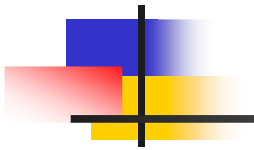


Phần 3

Các thuật toán tìm kiếm



Bài toán

- Tìm kiếm tuần tự
- Tìm kiếm nhị phân
- Cây nhị phân tìm kiếm

Bài toán

- Giả sử R là mảng gồm *n* bản ghi, bản ghi thứ k có một khoá X(k), với k=1,2,...,n. Hãy tìm bản ghi có khoá bằng (hoặc bé hơn, lớn hơn) giá trị b cho trước.
- Giá trị *b* được gọi là khoá tìm kiếm hay đối trị tìm kiếm.
- Bài toán được gọi là giải xong nếu như tìm được bản ghi thứ *k* nào đó mà X(k) = b hoặc khẳng định được rằng không có bản ghi nào thoả mãn điều kiện đặt ra, tức là X(k)≠ *b* với mọi k=1,2,...,n.

Tìm kiếm tuần tự

- **ý tưởng**
  - Bắt đầu từ bản ghi thứ nhất (k=1), lần lượt so sánh khoá tìm kiếm b với khoá tương ứng của các bản ghi trong mảng, cho tới khi hoặc là tìm được bản ghi thoả mãn điều kiện hoặc là hết mảng mà không thấy

Tìm kiếm tuần tự

Thuật toán

Input : Mảng X[1..n], giá trị *b*

OutPut : Chỉ số $k \in [1..n]$ mà tại đó X(k)=b hoặc 0 nếu X(k)≠b với $\forall k \in [1..n]$

```
Function TKTT ( X: mảng; b : Khoá tìm kiếm):  Chỉ số;
Begin
    k:=1;
    While (k ≤ n) & (X(k) ≠ b) do k:=k+1;
    If  k=n+1  then  TKTT:= 0    else      TKTT := k;
End;
```


Tìm kiếm tuần tự

Rõ ràng là trong trường hợp xấu nhất ta cần 2n phép so sánh. Tuy nhiên nếu ta thực hiện một cải tiến nhỏ

Input : Mảng X[1..n+1], giá trị *b*

OutPut : Chỉ số $k \in [1..n]$ mà tại đó X(k)=b hoặc 0 nếu X(k)≠b với $\forall k \in [1..n]$

```
Function TKTT ( X: mảng; b : Khoá tìm kiếm):  Chỉ số;
Begin
    X(n+1)=b;   k:=1;
    While   X(k) ≠ b  do k:=k+1;
    If  k=n+1  then  TKTT:= 0   else      TKTT := k;
End;
```

thì trong trường hợp xấu nhất ta cần n+1 phép so sánh. Trong cả hai trường hợp ta có độ phức tạp của thuật toán là O(n).

Tìm kiếm nhị phân

- **ý tưởng**
- Trên mảng các bản ghi mà khoá đã được sắp xếp tăng dần so sánh khoá tìm kiếm b với khoá của bản ghi ở giữa của đoạn mảng đang xét. Chẳng hạn, mảng đang xét là X[l..r] thì phần tử (bản ghi) ở giữa là X(k) với k=(l+r)/2; nếu X(k)=b thì việc tìm kiếm kết thúc, nếu X(k)>b thì việc tìm kiếm tiếp tục được thực hiện trên mảng con X[l,k-1], trong trường hợp X(k)>b thì việc tìm kiếm sẽ được thực hiện trên mảng con X[k+1,r].
- Quá trình tìm kiếm trên các mảng con sẽ được tiếp tục với thủ tục nêu trên.

Tìm kiếm nhị phân

Thuật toán

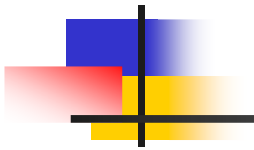
Input : Mảng X[1..n], giá trị *b*

OutPut : Chỉ số $k \in [1..n]$ mà tại đó X(k)=b hoặc 0 nếu $X(k) \neq b$ với $\forall k \in [1..n]$

```
Function TKNP (b : Khoá tìm kiếm): Chỉ số;
Begin
    TKNP := 0; can_trai :=1;   can_phai:=n;
    While   can_trai ≤  can_phai   do
    begin
          k:= (can_trai + can_phai) / 2 ;
          If   X(k) = b   then  TKNP:= k ; Exit   _loop;
          else       If   b< X(k)  then  can_phai :=       k-1
                     else  can_trai      :=         k+1;
    end
End;
```

Tìm kiếm nhị phân

- Ta có nhận xét rằng có thể bỏ công việc Exit_loop mà thuật toán vẫn thực hiện đúng nhưng thuật toán luôn rơi vào tình trạng xấu nhất về số lượng phép toán.
- **3. Đánh giá**
- Ta sẽ xét số vòng lặp mà thuật toán phải thực hiên. Giả sử $n=2^d$. Rõ ràng là trong trường hợp xấu nhất thuật toán thực hiện tới d vòng lặp. Mỗi vòng lặp có thể có tới 3 phép so sánh. Như vậy số tối đa phép so sánh mà thuật toán phải thực hiện là 3d hay $3[\log_2 n]+3$ với n tuỳ ý. Vậy độ phức tạp của thuật toán là $O(\log_2 n)$.



Cây nhị phân tìm kiếm

- **Khái niệm**
- Cây tìm kiếm nhị phân là cây nhị phân trong đó:
- mỗi con của một đỉnh hoặc là con bên phải hoặc là con bên trái;
- không có đỉnh nào có hơn 1 con bên phải hoặc hơn 1 con bên trái;
- Nói theo cách khác, đó là một cây nhị phân được sắp mà mỗi đỉnh được gán một giá trị khoá, khoá này phải thoả mãn yêu cầu: khoá của một đỉnh **lớn hơn** khoá của tất cả các đỉnh thuộc cây con *bên trái* và **nhỏ hơn** khoá của tất cả các đỉnh thuộc cây con *bên phải* của nó

Cây nhị phân tìm kiếm

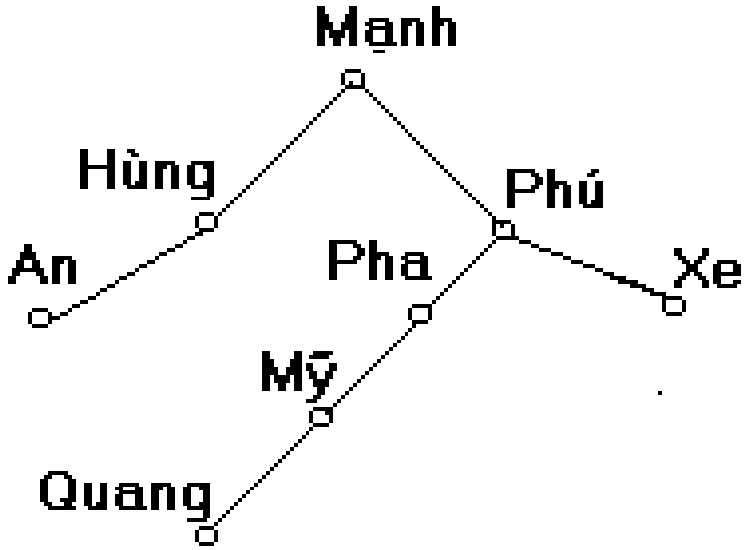
ý tưởng

- Tạo một cây nhị phân tìm kiếm mà khoá tại mỗi đỉnh của nó chính là khoá trong danh sách (hay mảng đang được xem xét). Việc lưu trữ giá trị khoá trong danh sách vào khoá của các đỉnh được thực hiện như sau:
- Khởi tạo cây có một đỉnh – gọi là gốc. Phần tử đầu tiên trong danh sách được dùng làm khoá của gốc.
- Lần lượt xét từng giá trị trong danh sách, từ phần tử thứ hai cho đến phần tử cuối cùng trong danh sách. Việc thêm một phần tử mới từ danh sách vào cây được thực hiện như sau:
- So sánh nó với khoá của các đỉnh đã có trên cây, bắt đầu từ **gốc**; rẽ sang trái nếu giá trị của phần tử này bé hơn khoá, rẽ sang phải nếu giá trị của phần tử này lớn hơn khoá.

Cây nhị phân tìm kiếm

- Trong trường hợp giá trị của phần tử này bé hơn khoá của đỉnh được xét mà đỉnh này không có con bên trái thì tạo ra đỉnh mới cho phần tử này như là con bên trái của nó. Phần tử đang xét được chọn làm khoá của đỉnh mới. Tương tự như vậy nếu giá trị phần tử mới lớn hơn khoá của đỉnh được xét mà đỉnh này không có con bên phải.

- Ví dụ, với danh sách các xâu kí tự "Mạnh, Phú, Hùng, Pha, Xe, Mỹ, Quang và An" thì cây nhị phân tìm kiếm ứng với nó có thể được mô tả như sau:



Cây nhị phân tìm kiếm

Thuật toán

- Thủ tục chính cần xây dựng ở đây là thêm một phần tử trong danh sách vào cây T. Mỗi nút của cây được khai báo s½n cấu trúc con trái và con phải, ngoài ra còn s½n cấu trúc khoá và cấu trúc lưu trữ nếu cần thiết.
- Input : Cây T và phần tử PT
- OutPut : Cây T có thêm phần tử PT ở vị trí lá nào đó và là cây nhị phân tìm kiếm

Cây nhị phân tìm kiếm

```
Procedure Thêm_PT( Pt: Phần tử; var T : cây);
Begin
        If  T = Nil        then     T:=Pt
        else
        begin
                If  Pt.Khoá < T.Khoá  then Thêm_PT(Pt, T.Con trái)
                else  Thêm_PT(Pt, T.Con phải);
        end;
End;
```

Cây nhị phân tìm kiếm

Thủ tục tạo cây nhị phân tìm kiếm T có thể được mô tả như sau:

```
Khởi tạo T;        T:= Phần tử thứ 1 trong mảng;
                   T.Con trái := Nil;    T.Con phải := Nil;
for i:=2 to n do
   begin
                   Khởi tạo Pt
                   Pt  :=  phần tử thứ  i  trong mảng
                   Pt.Con trái := Nil;   Pt.Con phải := Nil;
                   Thêm_PT(  Pt, T);
   end;
```

Thủ tục tìm một phần tử trên cây T đã được khởi tạo trên đây. Nếu tìm thấy Ptr sẽ mang giá trị của phần tử này, trong trường hợp ngược lại nó có giá trị NIL.

Cây nhị phân tìm kiếm

3. Đánh giá

- Trong trường hợp xấu nhất, khi mà danh sách hầu như đã được sắp xếp, thuật toán tìm phần tử có số phép toán so sánh cần thiết là 2n. Khi này, thuật toán hoạt động tương tự như thuật toán tìm kiếm tuần tự. Như vậy, hiển nhiên, độ phức tạp của thuật toán tìm kiếm trên cây nhị phân này là O(n).
- Nếu cây nhị phân xây dựng được là cây nhị phân hoàn chỉnh (hay cây nhị phân cân đối) thì rõ ràng số phép toán so sánh chỉ là cỡ $O(\log_2 n)$.
- Như vậy dạng cây nhị phân quyết định tới số phép toán so sánh phải thực hiện. Mặt khác, dạng cây nhị phân tìm kiếm lại phụ thuộc vào danh sách ban đầu (dãy khoá đưa vào).
- Tuy nhiên, ta cũng dễ dàng nhận thấy là cây nhị phân với ưu điểm nổi trội của nó là việc thêm, bớt một phần tử là rất thuận tiện.

Bài toán

- Tìm kiếm tuần tự
- Tìm kiếm nhị phân
- Cây nhị phân tìm kiếm