

Chương 6

Các thuật toán tìm kiếm

1. Bài toán tìm kiếm

Tìm kiếm là một đòi hỏi thường xuyên trong việc xử lý các bài toán tin học, nhất là đối với các bài toán quản lý. Tuy nhiên, ở đây ta chỉ xét các thuật toán tìm kiếm một cách tổng quát, không liên quan đến mục đích xử lý cụ thể nào.

Ta phát biểu bài toán tìm kiếm như sau:

Cho một bảng gồm n bản ghi $R_1, R_2, \dots, R_n$. Mỗi bản ghi R_i có một khóa tìm kiếm là $R_i.key$ ($1 \leq i \leq n$). Hãy tìm bản ghi có giá trị khóa tìm kiếm là KH cho trước.

KH được gọi là khóa tìm kiếm.

Công việc tìm kiếm sẽ hoàn thành khi có một trong hai tình huống sau đây xảy ra.

1. Tìm được bản ghi có giá trị khóa tìm kiếm bằng KH, lúc đó việc tìm kiếm là thành công.

2. Không tìm được bản ghi nào có khóa tìm kiếm bằng KH, lúc đó việc tìm kiếm là không thành công.

Cũng giống như việc sắp xếp, khóa của mỗi bản ghi chính là để tìm kiếm nhận biết của bản ghi đó trong tìm kiếm. Trong các thuật toán ta coi nó là để điền cho bản ghi đó và trong các ví dụ ta cũng chỉ nói tới khóa. Để đơn giản, ta coi các khóa $R_i.key$ ($1 \leq i \leq n$) là các số khác nhau. Trong chương này ta cũng chỉ xét các phương pháp tìm kiếm cơ bản, phổ biến, để với dữ liệu được lưu trữ ở bộ nhớ trong và được gọi là tìm kiếm trong. Có hai phương pháp tìm kiếm trong là tìm kiếm tuần tự và tìm kiếm nhị phân mà ta sẽ xét trong các phần sau đây.

2. Tìm kiếm tuần tự

2.1. Nguyên tắc tìm kiếm

Tìm kiếm tuần tự là phương pháp tìm kiếm rất đơn giản. Nguyên tắc tìm kiếm của phương pháp này có thể tóm tắt như sau:

Bắt đầu từ bên ghi thứ nhất, lần lượt so sánh khóa tìm kiếm KH với các khóa tiếp theo của các bên ghi trong bảng, cho đến khi tìm được một bên ghi mong muốn trong bảng (trường hợp tìm kiếm thành công, trả về vị trí của bên ghi tìm được) hoặc đã hết bảng mà không thấy (trường hợp tìm kiếm không thành công).

Ví dụ minh họa

Giả sử các khóa tiếp theo của các bên ghi trong bảng có 6 bên ghi là dãy số:

38 12 -56 95 23 11 và khóa cần tìm là KH=95, khi đó việc tìm kiếm được thực hiện như mô tả dưới đây.

Với $i=1 < n$ $R_1.key=38 \neq KH$, chuyển sang so sánh bên ghi tiếp theo.

Với $i=2 < n$ $R_2.key=12 \neq KH$, chuyển sang so sánh bên ghi tiếp theo.

Với $i=3 < n$ $R_3.key=-56 \neq KH$, chuyển sang so sánh bên ghi tiếp theo.

Với $i=4 < n$ $R_4.key=95 = KH$, kết thúc tìm kiếm và trả về vị trí tìm được là $i=4$

Ví dụ trên đây minh họa cho một phép tìm kiếm thành công, bên được tìm thấy ví dụ minh họa cho phép tìm kiếm không thành công.

2.2. Giải thuật

Thuật toán dưới đây thể hiện giải thuật tìm kiếm tuần tự, tìm kiếm khóa KH trên một dãy khóa X, có n phần tử. Nếu có nó sẽ trả về chỉ số của khóa ấy, nếu không có nó trả về giá trị 0.

int Sequence_Search(X, n, KH)

{

1. {Khởi tạo}

i=1;

2. {tìm khóa trong dãy}

```

        while (X[i]!=KH) && (i<=n)
            i=i+1;
3. {Kiểm tra và trả về kết quả tìm kiếm}

    if (i<=n) return (i);
    else return (0);
}

```

Để “tìm kiếm” thì tiếp theo ta có thể cải thiện giải thuật trên bằng cách thêm vào một khoá phụ X_{n+1} , có giá trị bằng KH như sau:

```

int Improvement_Sequence_Search(X, n, KH)

```

```

{
    1. {Khởi tạo}

        i=1; X[n+1]=KH;

    2. {tìm khoá trong dãy}

        while (X[i]!=KH)

            i=i+1;

    3. {Kiểm tra và trả về kết quả tìm kiếm}

        if (i==n+1) return (0);
        else return (i);
}

```

2.3. Đánh giá hiệu suất của giải thuật

Để đánh giá hiệu suất của phép tìm kiếm ta có thể đưa vào số lượng các phép so sánh. Ta thấy với giải thuật trên (cải thiện), trong trường hợp tốt nhất, chỉ cần 1 phép so sánh, $C_{\min}=1$, còn trong trường hợp xấu nhất số phép toán so sánh là $C_{\max}=n+1$. Nếu hiểu tổng số khoá tìm kiếm trùng với một khoá nào đó của bảng là độ phức tạp thì $C_{tb}=(n+1)/2$. Tóm lại, cả hai trường hợp xấu nhất cũng như trung bình, thì tiếp theo tìm kiếm theo giải thuật trên là $O(n)$.

Trong trình tự sắp xếp dãy khoá đã được sắp xếp theo chiều tăng dần (hoặc giảm dần), thì giá trị trung bình của hai giá trị thu được sẽ nhỏ hơn. Tuy nhiên, trong trình tự sắp xếp này ta có thể áp dụng phương pháp tìm kiếm khác, với thời gian nhanh hơn nhiều, đó là phương pháp tìm kiếm nhị phân.

3. Tìm kiếm nhị phân

3.1. Nguyên tắc

Tìm kiếm nhị phân là phương pháp tìm kiếm khá thông dụng. Trong phương pháp này ta áp dụng chiến lược “chia để trị” để giải quyết bài toán tìm kiếm. Việc này cũng giống như việc ta tìm một tờ trong quyển tài liệu. Ta có thể hiểu đơn giản như sau: để tìm tờ, ta mở tài liệu vào trang “giữa”, tìm tờ trong trang này, nếu có, việc tìm kiếm là thành công, còn nếu không có ta tìm “nửa trước” hoặc nửa sau của tài liệu. Việc tìm “nửa trước” hay “nửa sau” cũng được thực hiện giống như trên (nghĩa là ta cũng mở vào trang giữa). Việc tìm kiếm được lặp lại khi tìm được tờ trong một trang “giữa” nào đó (tìm kiếm thành công), hoặc khi tài liệu chỉ còn một trang, mà không có tờ cần tìm (tìm kiếm không thành công).

Như vậy, trong tìm kiếm nhị phân, giả sử với dãy khoá là $X_1, X_2, \dots, X_r$ nó luôn chứa khoá cần giữa là X_j , với $j = [(1+r)/2]$ để so sánh với khoá tìm kiếm KH. Tìm kiếm sẽ kết thúc nếu $KH = X_j$. Nếu $KH < X_j$ tìm kiếm được thực hiện tiếp với $X_1, X_2, \dots, X_{j-1}$, còn $KH > X_j$, tìm kiếm được thực hiện tiếp với $X_{j+1}, X_{j+2}, \dots, X_r$. Với dãy khoá sau, một kết quả thu được trình bày dưới đây. Quá trình tìm kiếm được tiếp tục khi tìm thấy khoá mong muốn hoặc dãy khoá xét đó là rỗng (không thấy).

Ví dụ minh họa

Giả sử các khoá trình tự của các bản ghi trong bảng có 6 bản ghi là dãy số tăng dần:

-56 11 12 23 38 95 và khoá cần tìm là $KH=95$, khi đó việc tìm kiếm được thực hiện như mô tả dưới đây.

	X_1	X_2	X_3	X_4	X_5	X_6	
	[-56	11	<u>12</u>	23	38	95]	
Bước 1:	j=3, khoá cần tìm là $X_3=12$, $KH > X_3$						Tìm kiếm tiếp tục với dãy khoá
				[23	<u>38</u>	95]	
Bước 2:	j=5, khoá cần tìm là $X_5=38$, $KH > X_5$						Tìm kiếm tiếp tục với dãy khoá
						<u>95]</u>	
Bước 3:	j=6, khoá cần tìm là $X_6=95$, $KH = X_6$						Tìm kiếm thành công!

Ví dụ minh họa tìm kiếm không thành công, bạn đọc xem như là một bài tập.

Sau đây là giải thuật tìm kiếm nhị phân.

3.2. Giải thuật.

Giải thuật được viết dưới dạng một thủ tục dùng để quy, tìm kiếm nhị phân. Nếu tìm thấy, giải thuật trả về vị trí của khoá được tìm thấy (giá trị j). Nếu không tìm thấy, giải thuật trả về giá trị 0.

```
int BINARY_SEARCH(X,l,r,KH)
```

```
{
```

```
1. {Trả về giá trị của dãy để xét tiếp, tìm kiếm không thành công}
```

```
    if (l>r)
```

```
        return 0;
```

```
2. {Tìm kiếm trong dãy để xét}
```

```
    else {
```

```
        j=(l+r) / 2;
```

2.1. {Tìm kiếm thành công}

if (KH == X[j])

BINARY_SEARCH=j;

2.2. {Tìm kiếm ở dãy trái}

else

if (KH < X[j])

BINARY_SEARCH(X,l,j-1,KH);

2.3. {Tìm kiếm ở dãy phải}

else

BINARY_SEARCH(X,j+1,r,KH);

}

}

Trong gọi thuật trên l, r tương ứng là chỉ số của khoá đầu tiên và chỉ số của khoá cuối cùng của dãy đang xét.

Gọi thuật tìm kiếm như phân cũng có thể được viết dưới dạng lisp, việc này bên dưới có thể tìm làm.

3.3. Đánh giá gọi thuật

Trong gọi thuật trên ta thấy số lượng phép so sánh phụ thuộc vào giá trị khoá KH. Trường hợp thuận lợi nhất khi tìm thấy dãy khoá $X_1, X_2, \dots, X_n$, mà gọi là gọi số là BINARY_SEARCH(X,1,n,KH), là $KH=X[(n+1+ \text{div } 2)]$, nghĩa là chỉ cần một phép so sánh, lúc đó $T_1=O(1)$. Trường hợp xấu nhất có phức tạp hơn. Ta gọi $w(r-l+1)$ là hàm biểu thị số lượng phép so sánh trong trường hợp xấu nhất trong việc tìm gọi BINARY_SEARCH(X,l,r,KH) và đặt $n=r-l+1$ (trong dãy khoá mà $l=1, r=n$) thì trong trường hợp xấu nhất ta sẽ phải gọi đệ quy, vậy ta có:

$$w(n) = 1 + w(n / 2)$$

Với phương pháp truy hồi có thể viết

$$w(n) = 1 + 1 + w(n/2^2)$$

$$w(n) = 1 + 1 + 1 + w(n/2^3)$$

Như vậy $w(n)$ có dạng $w(n) = k + w(n/2^k)$

Khi $(n/2^k) = 1$ ta có $w(n/2^k) = w(1)$ và khi đó tìm kiếm phải kết thúc. Song $(n/2^k) = 1$ thì suy ra $2^k \leq n \leq 2^{k+1}$, do đó $k \leq \log_2 n < k+1$, nghĩa là có thể viết $k = \log_2 n$. Vì vậy, cuối cùng ta có

$$w(n) = \lfloor \log_2 n \rfloor + 1$$

$$\text{hay } T_X(n) = O(\log_2 n)$$

Ngay khi ta cũng chứng minh được $T_{\text{ib}}(n) = O(\log_2 n)$

Rõ ràng so với tìm kiếm tuần tự (có thể gian trung bình là $O(n)$), chi phí tìm kiếm nhị phân ít hơn khá nhiều. Hiện tại không có phương pháp tìm kiếm nào, dựa trên việc so sánh giá trị khoá để có thể đạt được kết quả tốt hơn.

Tuy nhiên, nên nhớ rằng tìm kiếm nhị phân chỉ thực hiện trên dãy khoá đã được sắp xếp, nên trong tìm kiếm cũng phải tính đến chi phí cho việc sắp xếp. Nếu dãy khoá luôn bị nhiễu, thì chi phí cho việc sắp xếp sẽ lớn, và đó là một nhược điểm lớn với phương pháp tìm kiếm này.