CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT

Data Structures & Algorithms

TÌM KIẾM

Nội dung

1. Nhu cầu tìm kiếm/sắp xếp.
2. Bài toán tìm kiếm.
3. Tìm kiếm tuyến tính.
4. Tìm kiếm nhị phân.

Nhu cầu

❖ Các yêu cầu trong một hệ thống thông tin:
- Lưu trữ
- **Tra cứu (tìm kiếm) →thường thực hiện nhất**
- Tính toán
- Báo biểu

❖ Dữ liệu đã được sắp xếp => tìm nhanh hơn.

=> Vấn đề:
- Tìm kiếm nhanh.
- Sắp xếp nhanh.

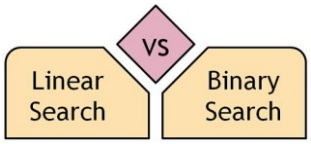
Searching

Bài toán tìm kiếm (Searching)

- Cho một cấu trúc dữ liệu mảng (a) để lưu trữ dãy số (danh sách) a_0, a_2 ,…,a_{n-1} .
- Tìm một phần tử có *khóa* bằng x trong mảng a.

Các Phương Pháp Tìm Kiếm Nội

1. Tìm kiếm tuyến tính
2. Tìm kiếm nhị phân

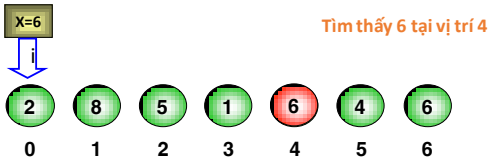


Tìm kiếm tuyến tính

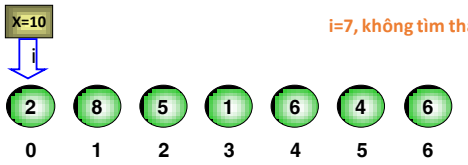
✓ Ý tưởng:

Thuật toán tiến hành so sánh x lần lượt với phần tử thứ nhất, thứ hai, ... của mảng a cho đến khi gặp được phần tử có khóa cần tìm, hoặc đã tìm hết mảng mà không thấy x

Tìm kiếm tuyến tính



Tìm kiếm tuyến tính



Tìm kiếm tuyến tính – Giải Thuật

Input: mảng a: a[0], a[1],…,a[n-1] và giá trị x

Output: vị trí của khóa x tìm được, trả về -1 nếu không tìm thấy

Các bước thực hiện:

- Bước 1: Gán i = 0; *// bắt đầu từ phần tử đầu tiên của dãy*
- Bước 2: So sánh a[i] với x, có 2 khả năng :
  - Nếu a[i] = x : Tìm thấy. Dừng.Trả về vị trí i.
  - a[i] != x : Sang Bước 3.
- Bước 3 :
  - i = i+1; *// xét tiếp phần tử kế trong mảng*
  - Nếu i > n : Hết mảng, không tìm thấy.Dừng
  - Ngược lại: Lặp lại Bước 2.

9

Tìm kiếm tuyến tính – Đánh giá

Trường hợp	Số lần so sánh	Giải thích
Tốt nhất	1	Phần tử đầu tiên có giá trị x
Xấu nhất	n	Phần tử cuối cùng có giá trị x
Trung bình	(n+1)/2	Giả sử xác suất các phần tử trong mảng nhận giá trị x là như nhau.

Độ phức tạp tính toán : $T(n) = O(n)$

Lưu ý: không phụ thuộc vào thứ tự của các phần tử mảng → dãy số bất kỳ

10

Tìm kiếm tuyến tính – Cài đặt

```
int LinearSearch(int a[], int n, int x)
{       int     i=0;
        while ((i<n) && (a[i]!=x ))   i++;
        if(i==n) return -1;          // tìm hết mảng nhưng không có x
        else     return i;           // a[i] là phần tử có khoá x
}
```

```
int LinearSearch(int a[],int n,int x)
{       int i=0;            // mảng gồm n phần tử từ a[0]..a[n-1]
        a[n] = x;           // thêm phần tử thứ n+1
        while (a[i]!=x )    i++;
        if (i==n)
                return -1; // tìm hết mảng nhưng không có x
        else
                return i;  // tìm thấy x tại vị trí i
}
```

11

Tìm kiếm tuyến tính – Lưu ý

- Thuật toán tìm tuyến tính sẽ duyệt tất cả các đối tượng trên "không gian tìm kiếm" để tìm ra đối tượng thỏa mãn "điều kiện tìm kiếm".
- Thông thường trong một bài toán kỹ thuật lập trình thì "không gian tìm kiếm" đơn giản nhất là: **mảng một chiều**, **ma trận**, một đoạn giá trị nào đó, **danh sách liên kết**, **cây**,…

12

Tìm kiếm tuyến tính – Lưu ý

- Mặt khác **"điều kiện tìm kiếm"** là tiêu chuẩn tìm kiếm được trình bày dưới dạng **một phát biểu không hình thức** và người lập trình phải **hình thức hóa nó bằng một "biểu thức logic"** trong chương trình.

13

Tìm kiếm tuyến tính – Lưu ý

- Thuật toán tìm kiếm tuyến tính sẽ duyệt toàn bộ không gian tìm kiếm để tìm đối tượng thỏa mãn tiêu chuẩn tìm kiếm **nên các đối tượng này không cần được sắp thứ tự**. Nói một cách khác là **dữ liệu không cần được tổ chức**.

14

Tìm kiếm tuyến tính

Bài 1. Viết hàm tìm kiếm giá trị x trong mảng 1 chiều các số thực trả về tìm thấy hay không.

Bài 2. Viết hàm tìm vị trí giá trị nhỏ nhất trong mảng một chiều các số thực.

Bài 3. Viết hàm tìm tất cả các vị trí có giá trị nhỏ nhất trong mảng một chiều các số thực.

Bài tập: Viết chương trình với menu cho người dùng lựa chọn nhập một mảng và lựa chọn các chức năng được viết ra ở bài 1,2,3 ở trên.

15

Đặt vấn đề

Đối với những dãy số đã ***có thứ tự*** (giả sử tăng),

$a_{i-1} \le a_i \le a_{i+1}$

❖ **NX**: nếu $x > a_i$ thì x thuộc $[a_{i+1}, a_{n-1}]$

nếu $x < a_i$ thì x thuộc $[a_0, a_{i-1}]$

→ Giới hạn phạm vi tìm kiếm sau mỗi lần so sánh x

16

Tìm kiếm nhị phân – Ý tưởng

Tại mỗi bước tiến hành so sánh x với phần tử nằm ở ***vị trí giữa của dãy*** tìm kiếm hiện hành, dựa vào kết quả so sánh này để quyết định giới hạn dãy tìm kiếm ở bước kế tiếp là nửa trên hay nửa dưới của dãy tìm kiếm hiện hành.

17

Tìm kiếm nhị phân – Thuật giải

Input: mảng a: a[0], a[1],…,a[n-1] đã có thứ tự và giá trị x

Output: vị trí của khóa x tìm được, trả về -1 nếu không tìm thấy

Các bước thực hiện:

- ✓ Bước 1: left = 0; right = n -1;
- ✓ Bước 2:
- mid = (left+right)/2; *// lấy mốc so sánh*
- So sánh a[mid] với x, có 3 khả năng :
  - a[mid] = x: Tìm thấy. Dừng
  - a[mid] > x: *//tìm tiếp x trong dãy con* $a_{left} .. a_{mid-1}$
    right =midle - 1;
  - a[mid] < x: *//tìm tiếp x trong dãy con* $a_{mid+1} .. a_{right}$
    left = mid + 1;

18

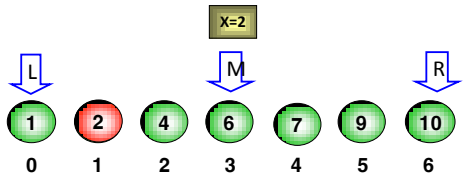
Tìm kiếm nhị phân – Thuật giải

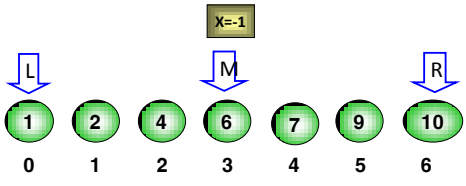
✓ Bước 3:

- Nếu left ≤ right //còn phần tử chưa xét, tìm tiếp.
  Lặp lại Bước 2.
- Ngược lại: Dừng; //Đã xét hết tất cả các phần tử.

19

Tìm thấy 2 tại vị trí 1





L=0

R=-1 => không tìm thấy X=-1

Tìm kiếm nhị phân – Cài đặt

```
int BinarySearch(int a[],int n,int x )
{       int left =0; right = n-1;
        int mid;
        do      {
                mid = (left + right)/2;
                if (x = a[mid])     return mid;        //Thấy x tại mid
                else
                        if (x < a[mid])     right = mid -1;
                        else left = mid +1;
        }while (left <= right);
        return -1; // Tìm hết dãy mà không có x
}
```

22

Tìm kiếm nhị phân – Đánh giá

Trường hợp	Số lần so sánh	Giải thích
Tốt nhất	1	Phần tử giữa của mảng có giá trị x
Xấu nhất	$\log_2 n$	Không có x trong mảng
Trung bình	$\log_2 (n/2)$	Giả sử xác suất các phần tử trong mảng nhận giá trị x là như nhau

Độ phức tạp tính toán : $T(n) = O(\log_2 n) < O(n)$

Lưu ý: chỉ áp dụng được cho những dãy đã có thứ tự →xét thời gian sắp xếp dãy số

23

Tìm kiếm nhị phân – Lưu ý

Điều kiện tiên quyết để áp dụng thuật toán tìm nhị phân là các đối tượng trong "không gian tìm kiếm" phải được sắp xếp thứ tự theo một tiêu chuẩn nào đó và ta sẽ dựa trên tiêu chuẩn này để tìm.

24

Bài tập – TÌM KIẾM NHỊ PHÂN

1. Anh/chị hãy viết hàm cài đặt thuật toán nhị phân trên mảng số nguyên có thứ tự giảm dần bằng ngôn ngữ C/C++.

2. Trình bày các bước (vẽ từng bước) theo hàm đã cài đặt ở câu 1 thực hiện tìm giá trị
X=50 trong mảng các số nguyên có giá trị như sau: 90 80 60 26 24 12

25

26