

# TÌM KIẾM

## (SEARCHING)

# Nội dung

2

1. Khái quát về tìm kiếm
2. Tìm tuyến tính (Linear Search)
3. Tìm nhị phân (Binary Search)

# Khái quát về tìm kiếm

3

- Tìm kiếm là một yêu cầu rất thường xuyên trong đời sống hàng ngày cũng như trong tin học
- Ví dụ:
  - ▣ Tìm kiếm một sinh viên trong lớp
  - ▣ Tìm kiếm một tập tin, thư mục trong máy
- Để đơn giản, xét bài toán tìm kiếm như sau:
  - ▣ Cho một dãy số gồm các phần tử  $a_1, a_2, \dots, a_n$ . Cho biết trong dãy này có phần tử nào có giá trị bằng  $X$  (cho trước) hay không?

# Khái quát về tìm kiếm

4

- Xét hai cách tìm kiếm:
  - ▣ Tìm kiếm tuyến tính (**Linear Search**) hay còn gọi là tìm kiếm tuần tự (**Sequential Search**)
  - ▣ Tìm kiếm nhị phân (**Binary Search**)

# Nội dung

5

1. Khái quát về tìm kiếm
2. **Tìm tuyến tính (Linear Search)**
3. **Tìm nhị phân (Binary Search)**

## 2. Tìm tuyến tính (Linear Search)

6

### Ý tưởng:

- Bắt đầu từ phần tử đầu tiên của danh sách, so sánh lần lượt từng phần tử của danh sách với giá trị X cần tìm
  - Nếu có phần tử bằng X thì trả về vị trí tìm thấy, thuật toán dừng lại (thành công)
  - Nếu đến cuối danh sách mà không có phần tử nào bằng X, thuật toán dừng lại (không thành công)
- If we find a match, the search terminates successfully by returning the index of the element
- If the end of the list is encountered without a match, the search terminates unsuccessfully

## 2. Tìm tuyến tính (Linear Search)

7

### Thuật toán:

*Input: Danh sách A và phần tử cần tìm X*

B1:  $i = 0$  ; // bắt đầu từ phần tử đầu tiên

B2: so sánh  $A[i]$  với  $X$ , có 2 khả năng :

- ▣  $A[i] = X$  : Tìm thấy  $X$  tại vị trí  $i$ . Dừng
- ▣  $A[i] \neq X$  : Sang B3

B3:  $i = i + 1$  // Xét phần tử tiếp theo trong mảng

Nếu  $i = n$  : Hết mảng, không tìm thấy. Dừng

Ngược lại: lặp lại B2

## 2. Tìm tuyến tính (Linear Search)

8

5

Khóa tìm

Vị trí = 2

|   |    |   |    |   |   |   |    |
|---|----|---|----|---|---|---|----|
| 0 | 1  | 2 | 3  | 4 | 5 | 6 | 7  |
| 7 | 13 | 5 | 21 | 6 | 2 | 8 | 15 |



Tìm thành công

Số lần so sánh: 3



## 2. Tìm tuyến tính (Linear Search)

9

9

Khóa tìm

|   |    |   |    |   |   |   |    |
|---|----|---|----|---|---|---|----|
| 0 | 1  | 2 | 3  | 4 | 5 | 6 | 7  |
| 7 | 13 | 5 | 21 | 6 | 2 | 8 | 15 |



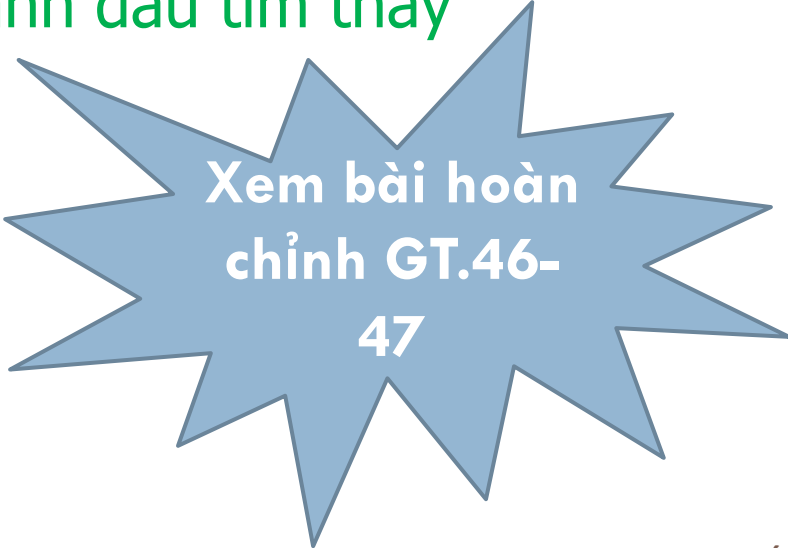
Không tìm thấy

Số lần so sánh: 8

## 2. Tìm tuyến tính (Linear Search)

10

```
void LSearch (int list[], int n, int key) {  
    int flag = 0;      // giả sử lúc đầu chưa tìm thấy  
    for( int i=0; i<n; i++ )  
        if ( list[i] == key ) {  
            cout<< "found at position"<<i;  
            flag = 1;    // đánh dấu tìm thấy  
            break;  
        }  
    if ( flag == 0 )  
        cout<<"not found";  
}
```



Xem bài hoàn  
chỉnh GT.46-  
47

## 2. Tìm tuyến tính (Linear Search)

11

```
int LSearch(int list[], int n, int key)
{
    int find= -1;
    for (int i=0; i<n; i++)
        if (list[i] == key)
        {
            find = i;
            break;
        }
    return find;
}
```

## 2. Tìm tuyến tính (Linear Search)

12

- Phân tích, đánh giá thuật toán

| Trường hợp | Số lần so sánh | Giải thích                                                         |
|------------|----------------|--------------------------------------------------------------------|
| Tốt nhất   | 1              | Phần tử đầu tiên có giá trị x                                      |
| Xấu nhất   | $n+1$          | 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. |

- Vậy giải thuật tìm tuyến tính có độ phức tạp tính toán cấp n:  $T(n) = O(n)$

# Nội dung

13

1. Khái quát về tìm kiếm
2. Tìm tuyến tính (Linear Search)
3. **Tìm nhị phân (Binary Search)**

# 3. Tìm nhị phân (Binary Search)

14

- Điều kiện:
  - ▣ Danh sách phải được sắp xếp trước
- Ý tưởng:
  - ▣ So sánh giá trị muốn tìm X với phần tử nằm ở vị trí giữa của danh sách:
    - Nếu **bằng**, tìm kiếm dừng lại (thành công)
    - Nếu **X lớn hơn** thì tiếp tục tìm kiếm ở phần danh sách bên phải phần tử giữa
    - Nếu **X nhỏ hơn** thì tiếp tục tìm kiếm ở phần danh sách bên trái phần tử giữa
  - ▣ We compare the element with the element placed approximately in the middle of the list
    - If a match is found, the search terminates successfully
    - Otherwise, we continue the search for the key in a similar manner either in the upper half or the lower half

# 3. Tìm nhị phân (Binary Search)

15

10

Khóa tìm

Vị trí = 3

Khóa cần tìm nhỏ hơn  
Khóa cần tìm lớn hơn  
Khóa cần tìm bằng

| 0    | 1 | 2 | 3  | 4   | 5  | 6  | 7  | 8  | 9     |
|------|---|---|----|-----|----|----|----|----|-------|
| 2    | 5 | 8 | 10 | 12  | 13 | 15 | 18 | 21 | 24    |
| ↑    |   |   |    | ↑   |    |    |    |    | ↑     |
| left |   |   |    | mid |    |    |    |    | right |

Tìm thấy

# 3. Tìm nhị phân (Binary Search)

16

## Thuật toán:

*Input: Danh sách A đã được sắp xếp và phần tử cần tìm X*

B1:  $\text{Left} = 0, \text{Right} = n-1$

B2:  $\text{Mid} = (\text{Left} + \text{Right})/2$  // lấy vị trí cận giữa

B3: So sánh X với  $A[\text{Mid}]$ , có 3 khả năng xảy ra:

- $A[\text{Mid}] = X$  // tìm thấy. Dừng thuật toán

- $A[\text{Mid}] > X$

$\text{Right} = \text{Mid}-1$  // Tiếp tục tìm trong dãy  $A[0] \dots A[\text{Mid}-1]$

- $A[\text{Mid}] < X$

$\text{Left} = \text{Mid}+1$  // Tiếp tục tìm trong dãy  $A[\text{Mid}+1] \dots A[\text{Right}]$

B4: Nếu  $(\text{Left} \leq \text{Right})$  // Còn phần tử chưa xét

Lặp lại B2

Ngược lại: Kết thúc

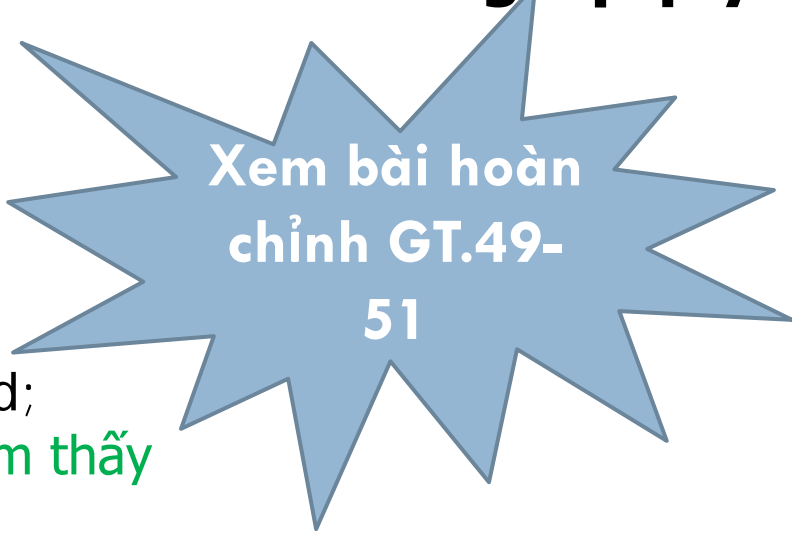


### 3. Tìm nhị phân (Binary Search)

17

```
void BSearch (int list[], int n, int key)
{
    int left, right, mid, flag = 0;
    left = 0; right = n-1;
    while (left <= right) {
        mid = (left + right)/2;
        if (list[mid] == key) {
            cout<<"found at "<< mid;
            flag =1;    // đánh dấu tìm thấy
            break;
        }
        else if (list[mid] < key)    left = mid +1;
        else
            right = mid -1;
    }
    if (flag == 0)
        cout<<"not found";
}
```

**Không đệ quy**



Xem bài hoàn  
chỉnh GT.49-  
51

### 3. Tìm nhị phân (Binary Search)

18

```
int BSearch_Recursion (int list[], int key, int left, int right)    Đệ quy
{
    if (left <= right)
    {
        int mid = (left + right)/2;
        if (key == list[mid])
            return mid;      // trả về vị trí tìm thấy key
        else if (key < list[mid])
            return BSearch_Recursion (list, key, left, mid-1);
        else return BSearch_Recursion (list, key, mid+1, right);
    }
    return -1;    // không tìm thấy
}
```

### 3. Tìm nhị phân (**Binary Search**)

19

- Phân tích, đánh giá thuật toán:

| 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 |

- Vậy giải thuật tìm nhị phân có độ phức tạp tính toán cấp n:  $T(n) = O(\log_2 n)$

# Nhận xét

20

- Giải thuật Tìm Nhị Phân tiết kiệm thời gian hơn rất nhiều so với giải thuật Tìm Tuyến Tính do:

| $O(\log_2 n) < O(n)$ | $n$     | $\log_2 n$ |
|----------------------|---------|------------|
|                      | 10      | 3          |
|                      | 100     | 6          |
|                      | 1,000   | 9          |
|                      | 10,000  | 13         |
|                      | 100,000 | 16         |

- Tìm Tuyến Tính là phương pháp tổng quát nhất để tìm kiếm trên một dãy bất kỳ
- Tìm Nhị Phân chỉ áp dụng được cho những dãy đã có thứ tự

# Nhận xét

21

- Khi muốn áp dụng giải thuật tìm Nhị Phân cần phải xét đến thời gian sắp xếp dãy số để thỏa điều kiện dãy số có thứ tự
- Thời gian này không nhỏ, và khi dãy số biến động cần phải tiến hành sắp xếp lại
- Tất cả các nhu cầu đó tạo ra khuyết điểm chính cho giải thuật tìm Nhị Phân
- Ta cần cân nhắc nhu cầu thực tế để chọn một trong hai giải thuật tìm kiếm trên sao cho có lợi nhất