

# CHƯƠNG 3. CÁC THUẬT TOÁN TÌM KIẾM & SẮP XẾP

## CÁC THUẬT TOÁN TÌM KIẾM

ThS. Nguyễn Chí Hiếu

2021

## NỘI DUNG

1. Giới thiệu bài toán tìm kiếm
2. Tìm kiếm tuyến tính
3. Tìm kiếm nhị phân

# NỘI DUNG

## 1. Giới thiệu bài toán tìm kiếm

## 2. Tìm kiếm tuyến tính

## 3. Tìm kiếm nhị phân

## Giới thiệu bài toán tìm kiếm

### Bài toán tìm kiếm

- ▶ Tìm kiếm là quá trình tìm phần tử có giá trị cho trước trong một danh sách, dãy số, dãy ký tự, ... Kết quả trả về là vị trí phần tử (nếu tìm thấy) hay trả về không có phần tử cần tìm.
- ▶ Trong bài toán tìm kiếm, thao tác cơ bản là so sánh giữa phần tử cần tìm và các phần tử trong dãy.

8	5	7	1	9	10
0	1	2	3	4	5



# NỘI DUNG

1. Giới thiệu bài toán tìm kiếm

2. Tìm kiếm tuyến tính

3. Tìm kiếm nhị phân

## Tìm kiếm tuyến tính/tuần tự (*linear search*)

### Ý tưởng

Giải thuật lần lượt so sánh phần tử  $x$  cần tìm với phần tử thứ nhất, thứ hai, ... của dãy và dừng thực hiện khi:

- ▶ Tìm được phần tử có khóa cần tìm.
- ▶ Duyệt hết các phần tử của dãy (không tìm thấy).

## Tìm kiếm tuyến tính/tuần tự (*linear search*)

**Thuật toán 1:** LinearSearch(a[], n, x)

- Đầu vào: mảng a gồm n phần tử và phần tử x.
- Đầu ra: vị trí của x hay -1 (không tìm thấy).

```
1  i ← 0
2  while i < n and a[i] ≠ x
3      i ← i + 1
4  if i = n
5      return -1
6  else
7      return i
```

## Tìm kiếm tuyến tính/tuần tự (*linear search*)

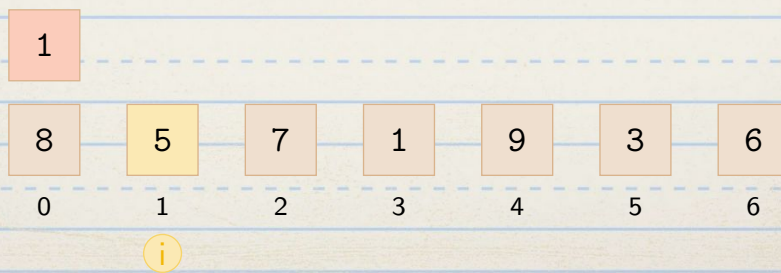
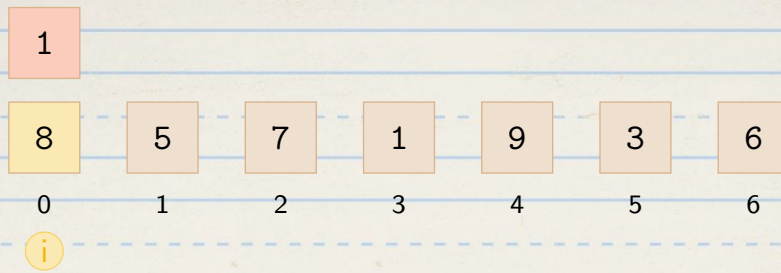
### Ví dụ 1

Cho dãy số a gồm 7 phần tử: 8, 5, 7, 1, 9, 3, 6. Tìm phần tử có giá trị 1.

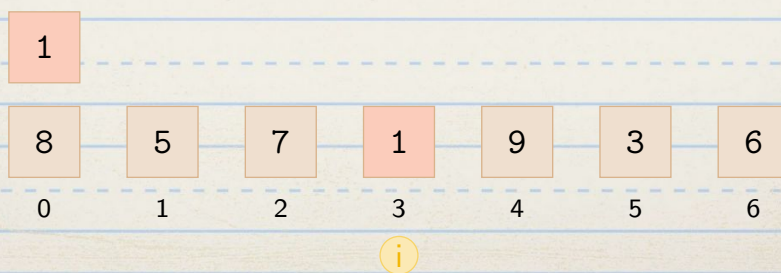
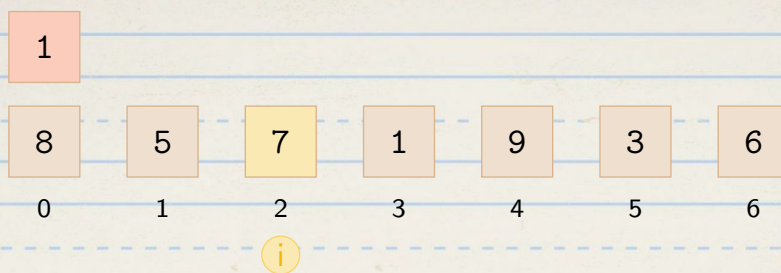
1						
8	5	7	1	9	3	6
0	1	2	3	4	5	6



## Tìm kiếm nhị phân (*binary search*)



## Tìm kiếm nhị phân (*binary search*)



## Tìm kiếm tuyến tính/tuần tự (*linear search*)

### Nhận xét

- ▶ Tại mỗi vòng lặp, giải thuật phải sử dụng 2 phép so sánh  $i < n$  và  $a[i] \neq x$ .
- ▶ Cải tiến giải thuật bằng cách sử dụng phương pháp lính canh (*sentinel*) để loại bỏ điều kiện  $i < n$ .

## Tìm kiếm tuyến tính/tuần tự (*linear search*)

### Phương pháp lính canh

**Thuật toán 2:** `LinearSearchEx(a[], n, x)`,

- Đầu vào: mảng  $a$  gồm  $n$  phần tử và phần tử  $x$ .
- Đầu ra: vị trí phần tử  $x$  hay  $-1$  (không tìm thấy).

```
1  i ← 0
2  a[n] ← x
3  while a[i] ≠ x
4      i ← i + 1
5  if i = n
6      return -1
7  else
8      return i
```



## Tìm kiếm tuyến tính/tuần tự (*linear search*)

### Đánh giá giải thuật

Phương pháp lính canh

Trường hợp	Số phép so sánh
Tốt nhất	1
Xấu nhất	$n + 1$
Trung bình	$\frac{n+1}{2}$
<b>Độ phức tạp thời gian</b>	$T(n) = O(n)$

## NỘI DUNG

1. Giới thiệu bài toán tìm kiếm

2. Tìm kiếm tuyến tính

3. Tìm kiếm nhị phân

## Tìm kiếm nhị phân (*binary search*)

### Ý tưởng

- ▶ Xét một dãy  $a$  có thứ tự gồm  $n$  phần tử:  $a_0 < a_1 < a_2 < \dots < a_{n-1}$ .
- ▶ Tìm phần tử  $x$  trong dãy  $a$  bằng cách so sánh  $x$  với phần tử giữa của dãy đang xét.
  - ▶ Nếu  $a_i = x$ , tìm thấy phần tử  $x$  tại vị trí  $i$ .
  - ▶ Nếu  $a_i > x$  thì chỉ có thể tìm  $x$  trong dãy con trái

$$a_0 \leq x \leq a_{i-1}.$$

- ▶ Ngược lại,  $a_i < x$  thì chỉ có thể tìm  $x$  trong dãy con phải

$$a_{i+1} \leq x \leq a_{n-1}.$$

- ▶ Sau mỗi lần so sánh, tiếp tục thực hiện tìm kiếm nhị phân với dãy con.

## Tìm kiếm nhị phân (*binary search*)

### Phương pháp đệ quy

**Thuật toán 3:** `BinarySearch(a[], left, right, x)`

- Đầu vào: mảng  $a$  gồm các phần tử và phần tử  $x$ .
- Đầu ra: vị trí của  $x$  hay  $-1$  (không tìm thấy).

```
1  if left > right
2      return -1
3  mid ← (left + right) / 2
4  if a[mid] = x
5      return mid
6  else if a[mid] > x
7      BinarySearch(a, left, mid - 1, x)
8  else
9      BinarySearch(a, mid + 1, right, x)
```



## Tìm kiếm nhị phân (*binary search*)

### Phương pháp lặp

**Thuật toán 4:** BinarySearch(a[], n, x)

- Đầu vào: mảng a gồm n phần tử và phần tử x.
- Đầu ra: vị trí của x hay -1 (không tìm thấy).

```
1  left ← 0, right ← n - 1
2  while left ≤ right
3      mid ← (left + right) / 2
4      if a[mid] = x
5          return mid
6      else a[mid] > x
7          right ← mid - 1
8      else
9          left ← mid + 1
10 return -1
```

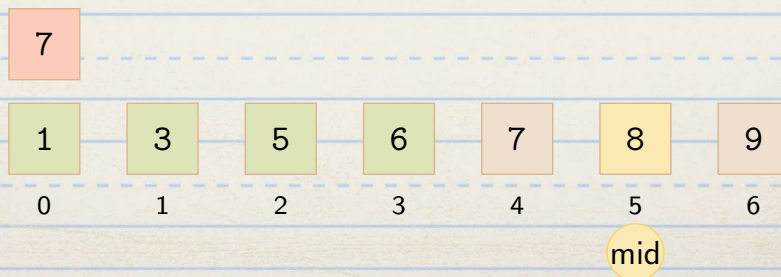
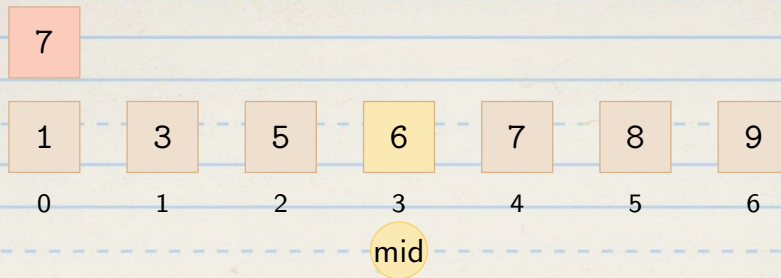
## Tìm kiếm nhị phân (*binary search*)

### Ví dụ 2

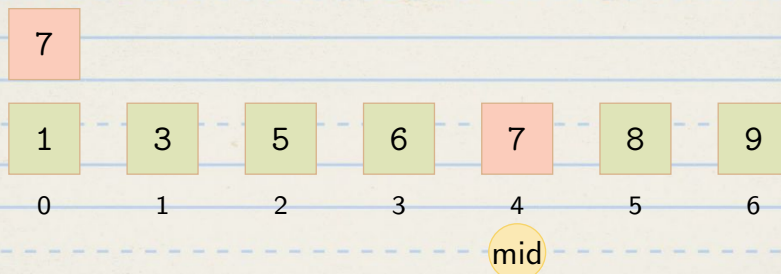
Cho dãy số *a* đã sắp thứ tự gồm 7 phần tử: 1, 3, 5, 6, 7, 8, 9. Tìm phần tử có giá trị 7.

7							
1	3	5	6	7	8	9	
0	1	2	3	4	5	6	

## Tìm kiếm nhị phân (*binary search*)



## Tìm kiếm nhị phân (*binary search*)





## Tìm kiếm nhị phân (*binary search*)

### Đánh giá giải thuật (Phương pháp đệ quy)

$$T(n) = \begin{cases} 1 & , n = 1 \\ T\left(\frac{n}{2}\right) + 1 & , n > 1 \end{cases}$$

Trường hợp	Số phép so sánh
Tốt nhất	1
Xấu nhất	$\log_2 n$
Trung bình	$\frac{\log_2 n}{2}$
<b>Độ phức tạp thời gian</b>	$T(n) = O(\log_2 n)$

## Bài tập

1. Cho dãy các số nguyên 12, 2, 8, 5, 1, 6, 4, 15. Áp dụng thuật toán LinearSearch và BinarySearch tìm phần tử có giá trị 15. Nhận xét số bước thực hiện giữa hai thuật toán.
2. Cho một dãy số gồm 32 phần tử đã có thứ tự. Giả sử phần tử cần tìm thuộc 1 trong 4 vị trí đầu tiên của dãy. Hãy cho biết giữa LinearSearch và BinarySearch, thuật toán nào số bước thực hiện ít hơn?
3. Cho dãy các số nguyên 1, 2, 4, 5, 6, 8, 12, 15. Cải tiến thuật toán LinearSearch trong trường hợp *dãy đã có thứ tự tăng dần*.
4. Xây dựng thuật toán tìm kiếm tam phân (*Ternary Search*) để tìm phần tử  $x$  trong dãy số đã có thứ tự.

## Tài liệu tham khảo



Donald E. Knuth.  
*The Art of Computer Programming, Volume 3.*  
Addison-Wesley, 1998.



Dương Anh Đức, Trần Hạnh Nhi.  
*Nhập môn Cấu trúc dữ liệu và Thuật toán.*  
Đại học Khoa học tự nhiên TP Hồ Chí Minh, 2003.



Niklaus Wirth.  
*Algorithms + Data Structures = Programs.*  
Prentice-Hall, 1976.



Robert Sedgewick.  
*Algorithms in C.*  
Addison-Wesley, 1990.