

Chương 2.  
TÌM KIẾM & SẮP XẾP  
CÁC THUẬT TOÁN SẮP XẾP CƠ BẢN

ThS. Nguyễn Chí Hiếu

2017

Giới thiệu bài toán sắp xếp

Sắp xếp chọn trực tiếp

Sắp xếp chèn trực tiếp

Sắp xếp đổi chỗ trực tiếp

Sắp xếp nổi bọt

Sắp xếp rung

# Giới thiệu bài toán sắp xếp

- ▶ Sắp xếp là quá trình xử lý các phần tử của một danh sách, dãy số, dãy ký tự, ... theo đúng thứ tự (thỏa tiêu chuẩn nào đó).
- ▶ Trong bài toán sắp xếp, hai thao tác cơ bản là so sánh và gán giữa hai phần tử trong dãy.

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

# Khái niệm nghịch thế

## Định nghĩa

Xét dãy số  $a$  gồm  $n$  phần tử

$$a = \{a_0, a_1, \dots, a_{n-1}\}$$

- ▶ Nếu  $i < j$  và  $a_i > a_j$  thì gọi đó là một nghịch thế (trường hợp tăng dần).
- ▶ Mảng không thứ tự: chứa nghịch thế.
- ▶ Mảng có thứ tự: không chứa nghịch thế.

# Sắp xếp chọn trực tiếp (*selection sort*)

## Ý tưởng

Cho dãy  $a = \{a_0, a_1, \dots, a_{n-1}\}$  gồm  $n$  phần tử, giải thuật là một vòng lặp thực hiện  $n - 1$  lần.

Tại mỗi lần lặp thứ  $i = 0, 1, \dots, n - 2$

- ▶ Chọn phần tử nhỏ nhất trong dãy  $a_i, a_{i+1}, \dots, a_{n-1}$ .
- ▶ Hoán vị phần tử được chọn với  $a_i$ .



# Sắp xếp chọn trực tiếp (*selection sort*)

Thuật toán 1: SelectionSort(a[], n)

- Đầu vào: mảng a gồm n phần tử.
- Đầu ra: mảng a có thứ tự tăng dần.

```
1  for i ← 0 to n - 2
2      min ← i
3      for j ← i + 1 to n - 1
4          if a[j] < a[min]
5              min ← j
6      Swap(a[i], a[min])
```

## Giải thích

- ▶ Dòng 2: vị trí phần tử có giá trị nhỏ nhất trong mảng chưa có thứ tự.
- ▶ Dòng 6: gọi hàm hoán vị 2 giá trị.

# Sắp xếp chọn trực tiếp (*selection sort*)

## Ví dụ 1

Cho dãy số  $a$  gồm 6 phần tử: 8, 5, 7, 1, 9, 10. Áp dụng giải thuật sắp xếp chọn trực tiếp sắp dãy  $a$  theo thứ tự tăng dần.

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

# Sắp xếp chọn trực tiếp (*selection sort*)

duy hie

$$\begin{cases} i = 0, \\ j = 1, \\ \text{min} = 0 \end{cases}$$

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

i

j

min

$$\begin{cases} i = 0, \\ j = 2, \\ \text{min} = 1 \end{cases}$$

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

i

j

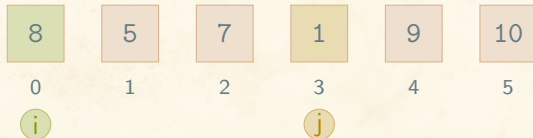
min



# Sắp xếp chọn trực tiếp (*selection sort*)

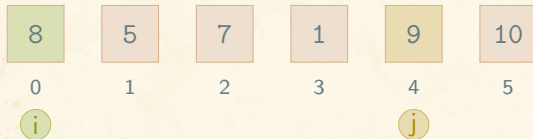
duyetho

$$\begin{cases} i = 0, \\ j = 3, \\ \text{min} = 1 \end{cases}$$



min

$$\begin{cases} i = 0, \\ j = 4, \\ \text{min} = 3 \end{cases}$$

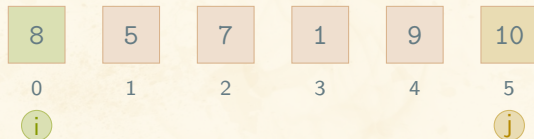


min

# Sắp xếp chọn trực tiếp (*selection sort*)

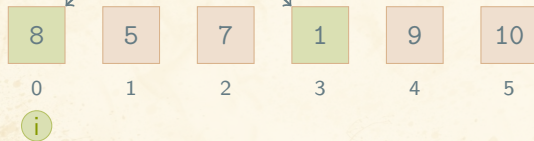
duy h. me

$$\begin{cases} i = 0, \\ j = 5, \\ \text{min} = 3 \end{cases}$$



min

$$\begin{cases} i = 0, \\ j = 6, \\ \text{min} = 3 \end{cases}$$

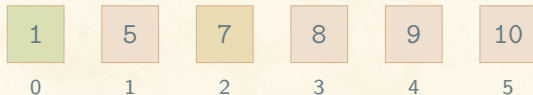


min

# Sắp xếp chọn trực tiếp (*selection sort*)

duy h. me

$$\begin{cases} i = 1, \\ j = 2, \\ \text{min} = 1 \end{cases}$$



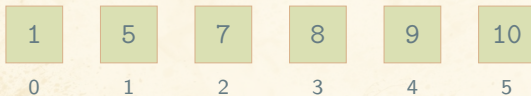
i

j

min

...

$$\{ i = 5 \}$$



# Sắp xếp chọn trực tiếp (*selection sort*)

## Đánh giá giải thuật

- Số phép so sánh

$$\sum_{i=0}^{n-2} \sum_{j=i+1}^{n-1} 1 = \sum_{i=0}^{n-2} ((n-1) - i) = \frac{n(n-1)}{2}.$$

- Mỗi lần hoán vị thực hiện 3 phép gán.

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

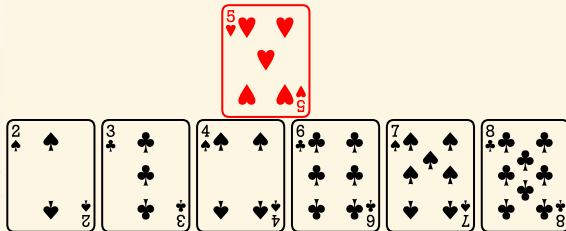
# Sắp xếp chèn trực tiếp (*insertion sort*)

## Ý tưởng

Cho dãy  $a = \{a_0, a_1, \dots, a_{n-1}\}$  gồm  $n$  phần tử, giải thuật là một vòng lặp thực hiện  $n - 1$  lần.

Giả sử  $a_0, a_1, \dots, a_{i-1}$  đã có thứ tự, tại mỗi lần lặp thứ  $i = 1, 2, \dots, n - 1$

- ▶ Chèn phần tử  $a_i$  vào đúng vị trí trong dãy  $a_0, a_1, \dots, a_{i-1}$  để được dãy  $a_0, a_1, \dots, a_{i-1}, a_i$  có thứ tự.





# Sắp xếp chèn trực tiếp (*insertion sort*)

Thuật toán 2: InsertionSort(a[], n)

- Đầu vào: mảng a gồm n phần tử.
- Đầu ra: mảng a có thứ tự tăng dần.

```
1  for i ← 1 to n - 1
2    x ← a[i]
3    pos ← i - 1
4    while pos ≥ 0 and a[pos] > x
5      a[pos + 1] ← a[pos]
6      pos ← pos - 1
7    a[pos + 1] ← x
```

# Sắp xếp chèn trực tiếp (*insertion sort*)

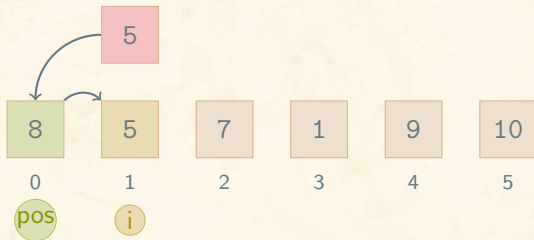
## Ví dụ 2

Cho dãy số  $a$  gồm 6 phần tử: 8, 5, 7, 1, 9, 10. Áp dụng giải thuật sắp xếp chèn trực tiếp sắp dãy  $a$  theo thứ tự tăng dần.

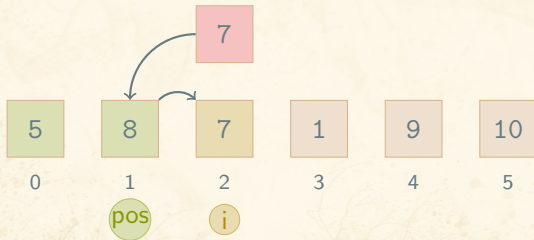
8	5	7	1	9	10
0	1	2	3	4	5

# Sắp xếp chèn trực tiếp (*insertion sort*)

$$\begin{cases} i = 1, \\ pos = 0 \\ x = a[i] = 5 \end{cases}$$



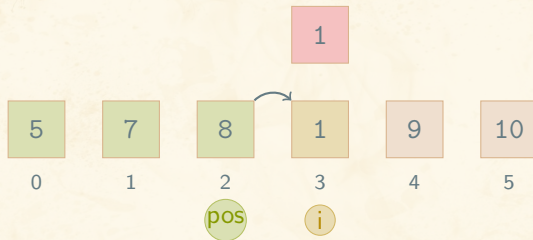
$$\begin{cases} i = 2, \\ pos = 1 \\ x = a[i] = 7 \end{cases}$$



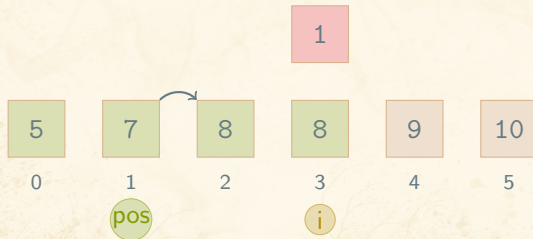
# Sắp xếp chèn trực tiếp (*insertion sort*)

duy hie

$$\begin{cases} i = 3, \\ pos = 2 \\ x = a[i] = 1 \end{cases}$$



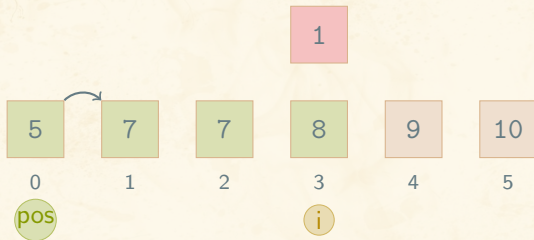
$$\begin{cases} i = 3, \\ pos = 1 \\ x = a[i] = 1 \end{cases}$$



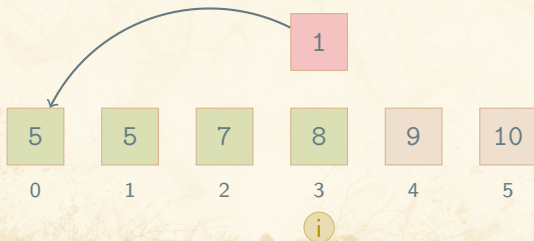
# Sắp xếp chèn trực tiếp (*insertion sort*)

duy hie

$$\begin{cases} i = 3, \\ pos = 0 \\ x = a[i] = 1 \end{cases}$$



$$\begin{cases} i = 3, \\ pos = -1 \\ x = a[i] = 1 \end{cases}$$

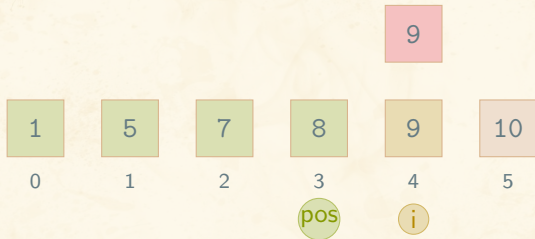




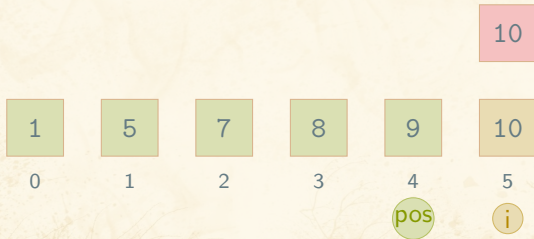
# Sắp xếp chèn trực tiếp (*insertion sort*)

duy hie

$$\begin{cases} i = 4, \\ pos = 3 \\ x = a[i] = 9 \end{cases}$$



$$\begin{cases} i = 5, \\ pos = 4 \\ x = a[i] = 10 \end{cases}$$



# Sắp xếp chèn trực tiếp (*insertion sort*)

## Đánh giá giải thuật

► Phép so sánh:  $a[pos] > x$ .

► Trường hợp tốt nhất

$$\sum_{i=1}^{n-1} 1 = n - 1.$$

► Trường hợp xấu nhất

$$\sum_{i=1}^{n-1} \sum_{pos=0}^{i-1} 1 = \sum_{i=1}^{n-1} (i - 1) = \frac{n(n - 1)}{2}.$$

# Sắp xếp chèn trực tiếp (*insertion sort*)

## Đánh giá giải thuật

► Phép gán:  $x = a[i]$ ,  $a[pos + 1] = a[pos]$ ,  $a[pos + 1] = x$ .

► Trường hợp tốt nhất

$$\sum_{i=1}^{n-1} 2 = 2(n-1).$$

► Trường hợp xấu nhất

$$\sum_{i=1}^{n-1} \left( \left( \sum_{pos=0}^{i-1} 1 \right) + 2 \right) = \frac{n(n-1)}{2} + 2(n-1) = \frac{(n-1)(n+4)}{2}.$$

# Sắp xếp chèn trực tiếp (*insertion sort*)

## Đánh giá giải thuật

Trường hợp	Số phép so sánh	Số phép gán
Tốt nhất	$n - 1$	$2(n - 1)$
Xấu nhất	$\frac{n(n-1)}{2}$	$\frac{(n-1)(n+4)}{2}$
<b>Độ phức tạp thời gian</b>	$T(n) = O(n^2)$	

# Sắp xếp đổi chỗ trực tiếp (*interchange sort*)

## Ý tưởng

Cho dãy  $a = \{a_0, a_1, \dots, a_{n-1}\}$  gồm  $n$  phần tử, giải thuật là một vòng lặp thực hiện  $n - 1$  lần.

Tại mỗi lần lặp thứ  $i = 0, 2, \dots, n - 2$

- ▶ Tìm tất cả nghịch thế chứa phần tử  $a_i$ .
- ▶ Đổi chỗ/hoán vị phần tử  $a_i$  và phần tử tương ứng trong nghịch thế.



# Sắp xếp đổi chỗ trực tiếp (*interchange sort*)

Thuật toán 3: InterchangeSort( $a[]$ ,  $n$ )

- Đầu vào: mảng  $a$  gồm  $n$  phần tử.
- Đầu ra: mảng  $a$  có thứ tự tăng dần.

```
1  for i ← 0 to n - 2
2      for j ← i + 1 to n - 1
3          if a[j] < a[i]
4              Swap(a[i], a[j])
```

## Giải thích

- Dòng 3: kiểm tra nghịch thế giữa  $a[i]$  và  $a[j]$ .

# Sắp xếp đổi chỗ trực tiếp (*interchange sort*)

## Ví dụ 3

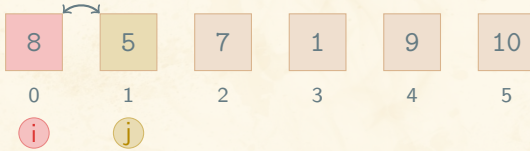
Cho dãy số  $a$  gồm 6 phần tử: 8, 5, 7, 1, 9, 10. Áp dụng giải thuật sắp xếp đổi chỗ trực tiếp sắp dãy  $a$  theo thứ tự tăng dần.

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

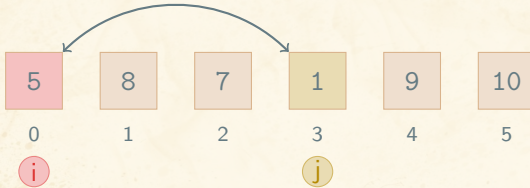
# Sắp xếp đổi chỗ trực tiếp (*interchange sort*)

duy hie

$$\begin{cases} i = 0, \\ j = 1 \end{cases}$$

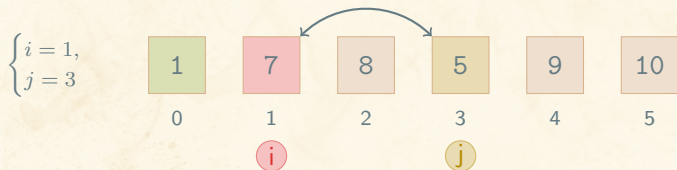
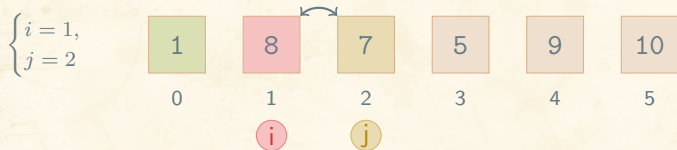


$$\begin{cases} i = 0, \\ j = 3 \end{cases}$$



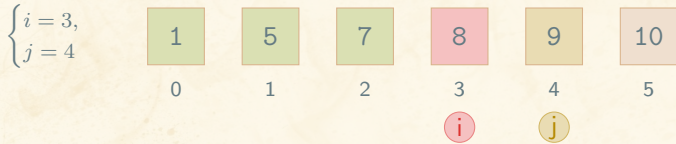
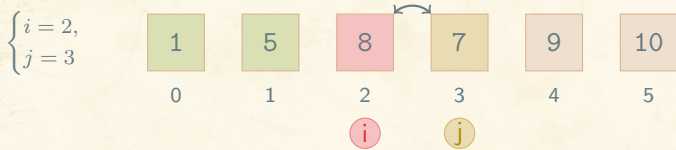
# Sắp xếp đổi chỗ trực tiếp (*interchange sort*)

duy h. me



# Sắp xếp đổi chỗ trực tiếp (*interchange sort*)

duy hie

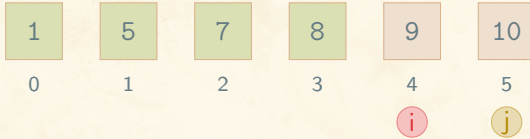




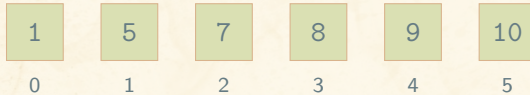
# Sắp xếp đổi chỗ trực tiếp (*interchange sort*)

duy hie

$$\begin{cases} i = 4, \\ j = 5 \end{cases}$$



$$\begin{cases} i = 5 \end{cases}$$



# Sắp xếp đổi chỗ trực tiếp (*interchange sort*)

## Đánh giá giải thuật

- Số phép so sánh

$$\sum_{i=0}^{n-2} \sum_{j=i+1}^{n-1} 1 = \sum_{i=0}^{n-2} ((n-1) - i) = \frac{n(n-1)}{2}.$$

- Mỗi lần hoán vị thực hiện 3 phép gán.

Trường hợp	Số phép so sánh	Số hoán vị
Tốt nhất	$\frac{n(n-1)}{2}$	0
Xấu nhất	$\frac{n(n-1)}{2}$	$\frac{n(n-1)}{2}$
<b>Độ phức tạp thời gian</b>	$T(n) = O(n^2)$	

# Sắp xếp nổi bọt (*bubble sort*)

## Ý tưởng

Cho dãy  $a = \{a_0, a_1, \dots, a_{n-1}\}$  gồm  $n$  phần tử, giải thuật là một vòng lặp thực hiện  $n - 1$  lần.

Tại mỗi lần lặp thứ  $i = 0, 1, \dots, n - 2$

- ▶ Bắt đầu từ cuối/đầu dãy, tìm nghịch thế giữa  $a_j$  và  $a_{j-1}$ , với  $j = n - 1, n - 2, \dots, 1$ .
- ▶ Hoán vị hai phần tử trong nghịch thế này.

📌 Phần tử nhỏ (nhẹ) sẽ nổi lên trên và phần tử lớn (nặng) sẽ chìm xuống đáy.

# Sắp xếp nổi bọt (*bubble sort*)

---

Thuật toán 4: BubbleSort(a[], n)

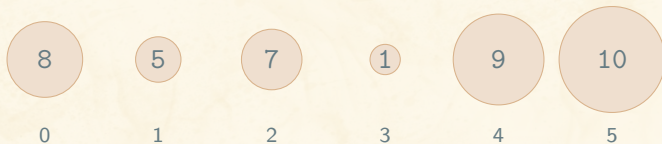
- Đầu vào: mảng a gồm n phần tử.
- Đầu ra: mảng a có thứ tự tăng dần.

```
1  for i ← 0 to n - 2
2    for j ← n - 1 downto i
3      if a[j] < a[j - 1]
4        Swap(a[j], a[j - 1])
```

# Sắp xếp nổi bọt (*bubble sort*)

## Ví dụ 4

Cho dãy số  $a$  gồm 6 phần tử: 8, 5, 7, 1, 9, 10. Áp dụng giải thuật sắp xếp nổi bọt sắp dãy  $a$  theo thứ tự tăng dần.

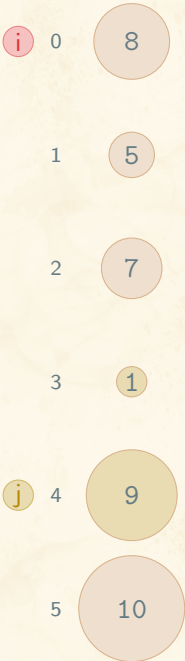




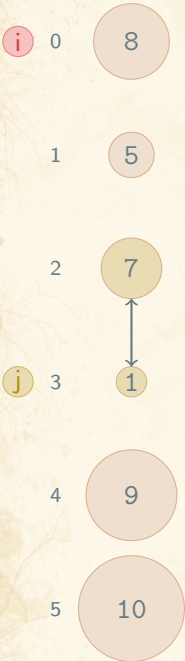
$$\begin{cases} i = 0, \\ j = 5 \end{cases}$$



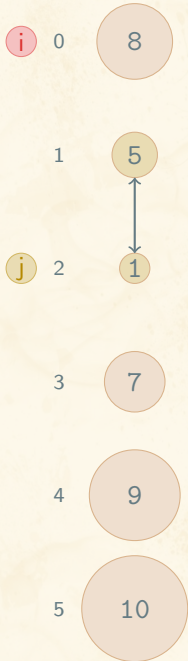
$$\begin{cases} i = 0, \\ j = 4 \end{cases}$$



$$\begin{cases} i = 0, \\ j = 3 \end{cases}$$



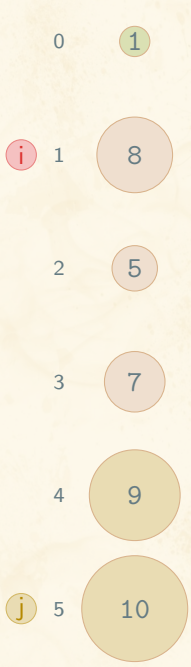
$$\begin{cases} i = 0, \\ j = 2 \end{cases}$$



$$\begin{cases} i = 0, \\ j = 1 \end{cases}$$



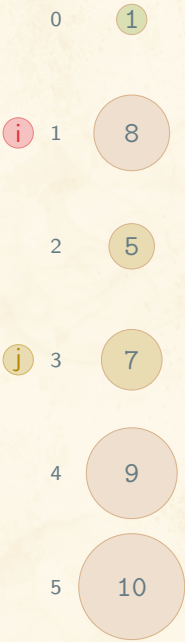
$$\begin{cases} i = 1, \\ j = 5 \end{cases}$$



$$\begin{cases} i = 1, \\ j = 4 \end{cases}$$



$$\begin{cases} i = 1, \\ j = 3 \end{cases}$$



$$\begin{cases} i = 1, \\ j = 2 \end{cases}$$



$$\begin{cases} i = 2, \\ j = 5 \end{cases}$$

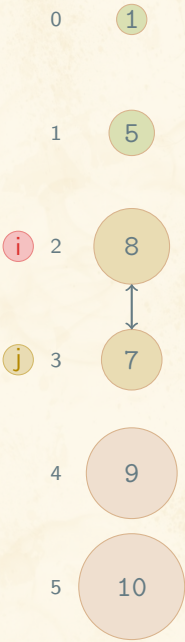




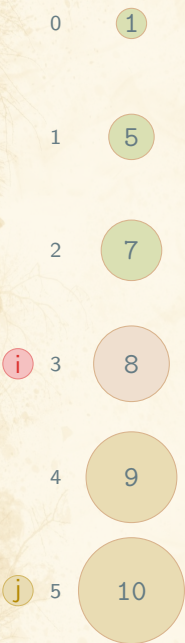
$$\begin{cases} i = 2, \\ j = 4 \end{cases}$$



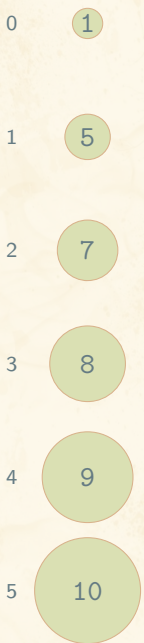
$$\begin{cases} i = 2, \\ j = 3 \end{cases}$$



$$\begin{cases} i = 3, \\ j = 5 \end{cases}$$



$$\begin{cases} i = 5 \end{cases}$$



# Sắp xếp nổi bọt (*bubble sort*)

## 📌 Nhận xét

- ▶ Không nhận biết được dãy đã có thứ tự hay có thứ tự từng phần.
- ▶ Các phần tử nhỏ được đưa về đúng vị trí rất nhanh, nhưng các phần tử lớn thì rất chậm.

# Sắp xếp nổi bọt (*bubble sort*)

## Đánh giá giải thuật

- Số phép so sánh

$$\sum_{i=0}^{n-2} \sum_{j=i+1}^{n-1} 1 = \sum_{i=0}^{n-2} ((n-1) - i) = \frac{n(n-1)}{2}.$$

- Mỗi lần hoán vị thực hiện 3 phép gán.

Trường hợp	Số phép so sánh	Số hoán vị
Tốt nhất	$\frac{n(n-1)}{2}$	0
Xấu nhất	$\frac{n(n-1)}{2}$	$\frac{n(n-1)}{2}$
Độ phức tạp thời gian	$T(n) = O(n^2)$	

# Sắp xếp rung (*shaker sort*)

## Ý tưởng

Tương tự như sắp xếp nổi bọt, nhưng khắc phục được khuyết điểm của giải thuật này bằng cách

- ▶ Lướt đi: xuất phát từ cuối dãy không thứ tự, đẩy phần tử nhỏ nhất về đầu dãy.
- ▶ Lướt về: xuất phát từ đầu dãy không thứ tự, đẩy phần tử lớn nhất về cuối dãy.
- ▶ Ghi nhận lại vị trí những đoạn đã có thứ tự.





# Sắp xếp rung (*shaker sort*)

Thuật toán 5: ShakerSort(a[], n)

- Đầu vào: mảng a gồm n phần tử.
- Đầu ra: mảng a có thứ tự tăng dần.

```
1  left ← 0, right ← n - 1
2  k ← right
3  while left < right
4      for j ← right downto left - 1
5          if a[j] < a[j - 1]
6              Swap(a[j], a[j - 1])
7              k ← j
8  left ← k
9
10 for i ← left to right - 1
11     if a[i] > a[i + 1]
12         Swap(a[i], a[i + 1])
13         k ← i
14 right ← k
```

# Sắp xếp rung (*shaker sort*)

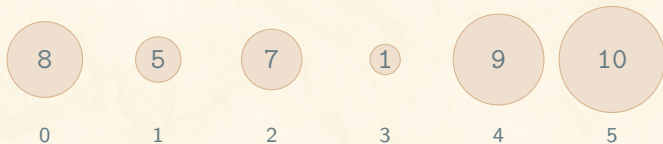
## Giải thích

- ▶ Dòng 2: ghi nhận vị trí xảy ra hoán vị cuối cùng.
- ▶ Dòng 4 → 7: đưa phần tử nhỏ về đầu dãy.
- ▶ Dòng 8: loại các phần tử có thứ tự ở đầu dãy.
- ▶ Dòng 10 → 13: đưa phần tử lớn về cuối dãy.
- ▶ Dòng 14: loại các phần tử có thứ tự ở cuối dãy.

# Sắp xếp rung (*shaker sort*)

## Ví dụ 5

Cho dãy số  $a$  gồm 6 phần tử: 8, 5, 7, 1, 9, 10. Áp dụng giải thuật sắp xếp rung sắp dãy  $a$  theo thứ tự tăng dần.



# Sắp xếp rung (shaker sort)

duy hie

$$\begin{cases} left = 0, \\ right = 5, \\ j = 5 \end{cases}$$

8

0

left

5

1

7

2

1

3

9

4

10

j

right

$$\begin{cases} left = 0, \\ right = 5, \\ j = 4 \end{cases}$$

8

0

left

5

1

7

2

1

3

9

4

j

10

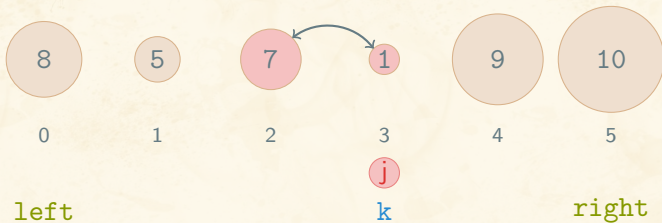
5

right

# Sắp xếp rung (shaker sort)

duyetho

$$\begin{cases} left = 0, \\ right = 5, \\ j = 3, \\ k = 3 \end{cases}$$



$$\begin{cases} left = 0, \\ right = 5, \\ j = 2, \\ k = 2 \end{cases}$$

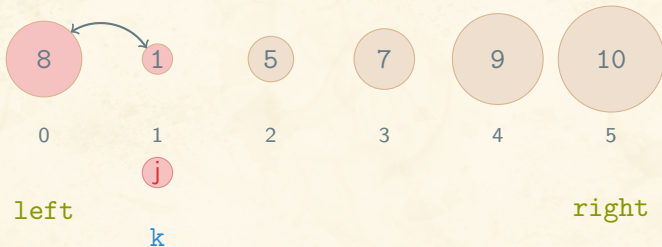




# Sắp xếp rung (shaker sort)

duyetho

$$\begin{cases} left = 0, \\ right = 5, \\ j = 1, \\ k = 1 \end{cases}$$



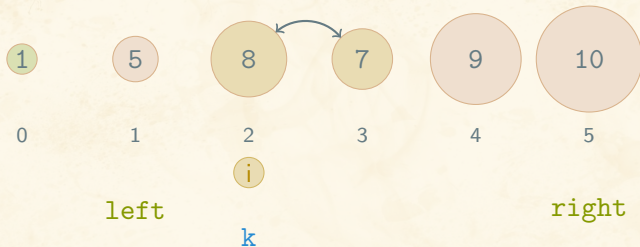
$$\begin{cases} left = 1, \\ right = 5, \\ i = 1, \\ k = 1 \end{cases}$$



# Sắp xếp rung (shaker sort)

duyetho

$$\begin{cases} \text{left} = 1, \\ \text{right} = 5, \\ i = 2, \\ k = 2 \end{cases}$$



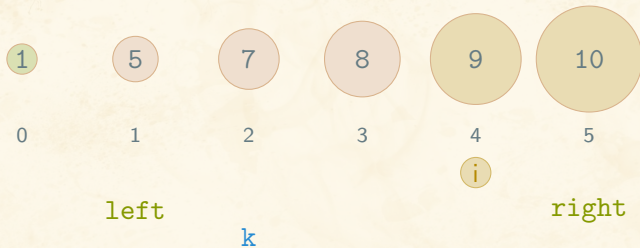
$$\begin{cases} \text{left} = 1, \\ \text{right} = 5, \\ i = 3, \\ k = 2 \end{cases}$$



# Sắp xếp rung (shaker sort)

duy h. me

$$\begin{cases} left = 1, \\ right = 5, \\ i = 4, \\ k = 2 \end{cases}$$

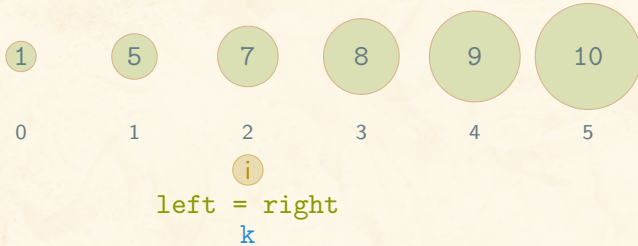


$$\begin{cases} left = 1, \\ right = 2, \\ j = 2, \\ k = 2 \end{cases}$$



# Sắp xếp rung (*shaker sort*)

$$\begin{cases} left = 2, \\ right = 2, \\ i = 2, \\ k = 2 \end{cases}$$



# Sắp xếp rung (*shaker sort*)

## Đánh giá giải thuật

- ▶ Trong trường hợp dãy số có thứ tự bộ phận thuật toán sẽ thực hiện số phép so sánh ít hơn Bubble Sort.
- ▶ Số phép hoán vị tương tự giải thuật Bubble Sort.

Trường hợp	Số phép so sánh	Số hoán vị
Tốt nhất	$n - 1$	0
Xấu nhất	$\frac{n(n-1)}{2}$	$\frac{n(n-1)}{2}$
Độ phức tạp thời gian	$T(n) = O(n^2)$	



## Chương 2. TÌM KIẾM & SẮP XẾP

### └ Sắp xếp rung

### └ Sắp xếp rung (*shaker sort*)

Sắp xếp rung (*shaker sort*)

Đánh giá giải thuật

- ▶ Trong trường hợp dãy số có thứ tự bộ phận thuật toán sẽ thực hiện số phép so sánh ít hơn Bubble Sort.
- ▶ Số phép hoán vị tương tự giải thuật Bubble Sort.

Trường hợp	Số phép so sánh	Số hoán vị
Tốt nhất	$n - 1$	0
Xấu nhất	$\frac{n(n-1)}{2}$	$\frac{n(n-1)}{2}$
Độ phức tạp thời gian	$T(n) = O(n^2)$	

### • Hỏi & đáp

- Tại sao cần phải có nhiều giải thuật sắp xếp?  
Mỗi giải thuật có một số ưu điểm đối với từng trường hợp riêng biệt.
- Giải thuật nào ổn định nhất? Giải thuật là ổn định (*stable*) nếu sau khi thực hiện sắp xếp, thứ tự tương đối của các mẫu tin có khóa bằng nhau không đổi.  
Bubble Sort kiểm tra nghịch thế giữa 2 phần tử kề nhau  $a_j$  và  $a_{j-1}$ .

# Bài tập

1. Cho dãy ký tự X, I, N, C, H, A, O. Áp dụng các giải thuật sắp xếp đã học để sắp theo đúng thứ tự bảng chữ cái Latinh.
2. Cho ví dụ minh họa ưu điểm của Shaker Sort so với Bubble Sort khi sắp xếp một dãy số.
3. Bài toán di chuyển đĩa

- ▶ Cho một dãy đĩa gồm 2 màu sáng và tối



- ▶ Xây dựng giải thuật di chuyển các đĩa sáng về bên trái và đĩa tối về bên phải. Mỗi bước chỉ được hoán vị hai đĩa kề nhau.



- ▶ Đếm số hoán vị hai đĩa khi thực hiện.

Gợi ý: xem dãy đĩa như một dãy số, đĩa màu sáng có giá trị là 0 và đĩa tối có giá trị là 1.



## Chương 2. TÌM KIẾM & SẮP XẾP

### └ Bài tập

### └ Bài tập

- Xét dãy số 2, 3, 4, 5, 6, 7, 8, 9, 1, sắp xếp dãy theo thứ tự tăng dần.
  - Shaker sort: chỉ cần 1 lần duyệt để đưa phần tử 1 về đầu dãy.
  - Bubble sort: cần  $n - 1 = 8$  lần duyệt (không nhận biết một phần dãy đã có thứ tự).

#### Bài tập

1. Cho dãy ký tự  $X, I, W, C, W, A, O$ . Áp dụng các giải thuật sắp xếp đã học để sắp theo đúng thứ tự bảng chữ cái Latinh.
2. Cho ví dụ minh họa ưu điểm của Shaker Sort so với Bubble Sort khi sắp xếp một dãy số.
3. Bài toán di chuyển đĩa
  - ▶ Cho một dãy đĩa gồm 2 màu sáng và tối
  - ▶ Xây dựng giải thuật di chuyển các đĩa sáng về bên trái và đĩa tối về bên phải. Mỗi bước chỉ được hoán vị hai đĩa kế nhau.
  - ▶ Dừng số hoán vị hai đĩa khi thực hiện.




Gợi ý: xem dãy đĩa như một dãy số, đĩa màu sáng có giá trị là 0 và đĩa tối có giá trị là 1.

## Chương 2. TÌM KIẾM & SẮP XẾP

### └ Bài tập

### └ Bài tập

#### Bài tập

1. Cho dãy ký tự X, I, W, C, W, A, O. Áp dụng các giải thuật sắp xếp đã học để sắp theo đúng thứ tự bảng chữ cái Latinh.
2. Cho ví dụ minh họa ưu điểm của Shaker Sort so với Bubble Sort khi sắp xếp một dãy số.  

3. Bài toán di chuyển đĩa  
▶ Cho một dãy đĩa gồm 2 màu sáng và tối  
  
▶ Xây dựng giải thuật di chuyển các đĩa sáng về bên trái và đĩa tối về bên phải. Mỗi bước chỉ được hoán vị hai đĩa kế nhau.  
  
▶ Dừng số hoán vị hai đĩa khi thực hiện.  
Gợi ý: xem dãy đĩa như một dãy số, đĩa màu sáng có giá trị là 0 và đĩa tối có giá trị là 1.

- Áp dụng giải thuật Bubble Sort và một cờ để đánh dấu sau khi một cặp đĩa được hoán vị.

BubbleSortEx(a[], n)

for i ← 0 downto n - 2

swapped = false

for j ← n - 1 to i

if a[j] < a[j - 1]

Swap(a[j], a[j - 1])

swapped ← true

if swapped = false

break

- Lưu ý, số hoán vị sẽ được giảm rất nhiều lần so với Bubble Sort.



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.



Donald E. Knuth.

*The Art of Computer Programming, Volume 3.*

Addison-Wesley, 1998.



Niklaus Wirth.

*Algorithms + Data Structures = Programs.*

Prentice-Hall, 1976.



Robert Sedgewick.

*Algorithms in C.*

Addison-Wesley, 1990.