CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT

Bài 02: Các thuật toán sắp xếp cơ bản

Nguyễn Thị Tâm
nguyenthitam.hus@gmail.com

Ngày 15 tháng 9 năm 2025

Nội dung

1. Giới thiệu
2. Sắp xếp nổi bọt - Bubble sort
3. Sắp xếp chọn - Selection sort
4. Sắp xếp chèn - Insertion sort
5. Sắp xếp trộn - Merge sort
6. Sắp xếp nhanh - Quick sort

Bài toán sắp xếp

Định nghĩa

Sắp xếp (*Sorting*) là quá trình tổ chức lại các dữ liệu theo thứ tự giảm dần hoặc tăng dần (*ascending or descending order)*

Bài toán sắp xếp (1)

Bài toán sắp xếp

Cho một mảng a gồm n phần tử $a[0], a[1], ..., a[n-1]$. Hãy sắp xếp mảng a theo thứ tự tăng dần hoặc giảm dần về giá trị.

Ví dụ

Mảng ban đầu: $a = \{4, 1, 3, 5, 6\}$
Mảng được sắp tăng dần: $\{1, 3, 4, 5, 6\}$
Mảng được sắp giảm dần: $\{6, 5, 4, 3, 1\}$

Độ quan trọng của thuật toán sắp xếp

40% thời gian hoạt động của máy tính là dành cho việc sắp xếp
– D.Knuth -

Bài toán sắp xếp (2)

Phân loại

- **Sắp xếp trong** (*internal sorting*): các bản ghi lưu trữ trong bộ nhớ trong (hay còn gọi là bảng)
- Sắp xếp ngoài (*external sorting*): các bản ghi lữu trữ ở bộ nhớ ngoài (tệp tin)

Đặc trưng của các thuật toán sắp xếp

- Tại chỗ (*in place*): nếu không gian nhớ phụ mà thuật toán đòi hỏi là O(1), nghĩa là bị chặn bởi hằng số không phụ thuộc vào độ dài của dãy cần sắp xếp
- Ổn định (*stable*): nếu các phần tử có cùng giá trị vẫn giữ nguyên thứ tự tương đối của chúng như trước khi sắp xếp

Bài toán sắp xếp (3)

Hai phép toán cơ bản mà các thuật toán sắp xếp sử dụng

- Đổi chỗ (swap): thời gian thực hiện là O(1)

```
void swap (int a[], int i, int j)
{
    int temp = a[i];
    a[i] = a[j];
    a[j] = temp;
}
```

- So sánh (compare): hàm compareTo(a,b)

Nội dung

1 Giới thiệu

2 Sắp xếp nổi bọt - Bubble sort

3 Sắp xếp chọn - Selection sort

4 Sắp xếp chèn - Insertion sort

5 Sắp xếp trộn - Merge sort

6 Sắp xếp nhanh - Quick sort

Sắp xếp nổi bọt

- Là thuật toán sắp xếp đơn giản nhất
- Nguyên tắc hoạt động
  - Bắt đầu với phần tử đầu tiên, so sánh với phần tử thứ hai. Nếu phần tử đầu tiên lớn hơn thì đổi chỗ nó với phần tử thứ hai
  - Quá trình tiếp tục cho các cặp phần tử liền kề trong dãy
  - Bắt đầu lại với hai phần tử đầu tiên, lặp lại cho đến khi không thể đổi chỗ các cặp phần tử được nữa

Sắp xếp nổi bọt

```
void bubbleSort(int a[]) {
    int i, j, n = a.length;
    for (i = 0; i < n - 1; i++) {
        for (j = 0; j < n - i - 1; j++) {
            if (a[j] > a[j + 1]) {
                // Swap a[j] and a[j + 1]
                swap(a,j,j+1);
            }
        }

}
```

Sắp xếp nổi bọt

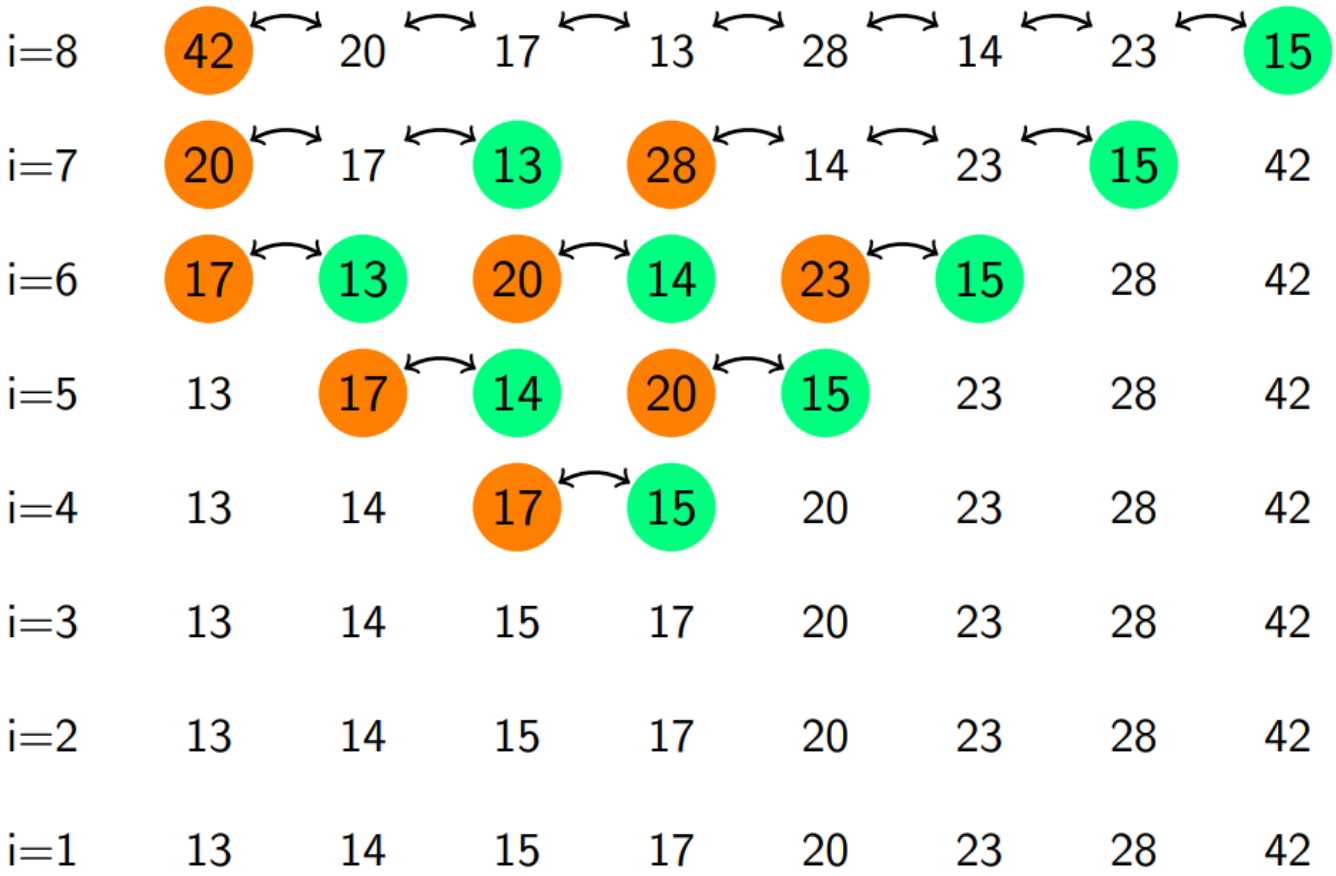
Ví dụ minh hoạ

Minh hoạ với dãy phần tử $a = \{42, 20, 17, 13, 28, 14, 23, 15\}$

Sắp xếp nổi bọt

Ví dụ minh hoạ

Minh hoạ với dãy phần tử $a = \{42, 20, 17, 13, 28, 14, 23, 15\}$



Sắp xếp nổi bọt

Phân tích thuật toán

- Trường hợp tốt nhất: 0 đổi chỗ, $\frac{n^2}{2}$ phép so sánh
- Trường hợp tồi nhất: $\frac{n^2}{2}$ phép so sánh và đổi chỗ
- Trường hợp trung bình: $\frac{n^2}{4}$ phép đổi chỗ, $\frac{n^2}{2}$ phép so sánh

Yêu cầu: sửa lại chương trình để trong trường hợp tốt nhất độ phức tạp là O(n)

Sắp xếp nổi bọt

Cải tiến

```
void bubbleSort(int a[]) {
    int i, j, n = a.length;
    for (i = 0; i < n - 1; i++) {
        boolean swapped = false;
        for (j = 0; j < n - i - 1; j++) {
            if (a[j] > a[j + 1]) {
                swap(a,j,j+1);
                swapped = true;
            }
        }
        if (!swapped) {
            break;
        }
    }
}
```

Nội dung

1 Giới thiệu

2 Sắp xếp nổi bọt - Bubble sort

3 Sắp xếp chọn - Selection sort

4 Sắp xếp chèn - Insertion sort

5 Sắp xếp trộn - Merge sort

6 Sắp xếp nhanh - Quick sort

Sắp xếp chọn

- Là thuật toán so sánh tại chỗ
- Nguyên tắc hoạt động
  - Tìm giá trị nhỏ nhất của mảng
  - Hoán đổi nó với phần tử ở vị trí đầu tiên
  - Lặp lại quá trình này cho phần còn lại của mảng

Sắp xếp chọn

Cách 1

```
void selectionSort (int a[])
{
    int i, j, n = a.length;
    for (i = 0; i < n-1; i++){
        1) find j: a[j] = min {a[i+1],....,a[n-1]}
        2) swap(a, i, j);
    }


}
```

Sắp xếp chọn

Cách 2

```
void selectionSort (int a[])
{
    int i, j, n = a.length;
    for (i = 0; i < n-1; i++){
        for (int j =i+1; j<n; j++){
            if (a[i] > a[j]){
                swap(a, i, j);
            }
        }
    }


}
```

Sắp xếp chọn

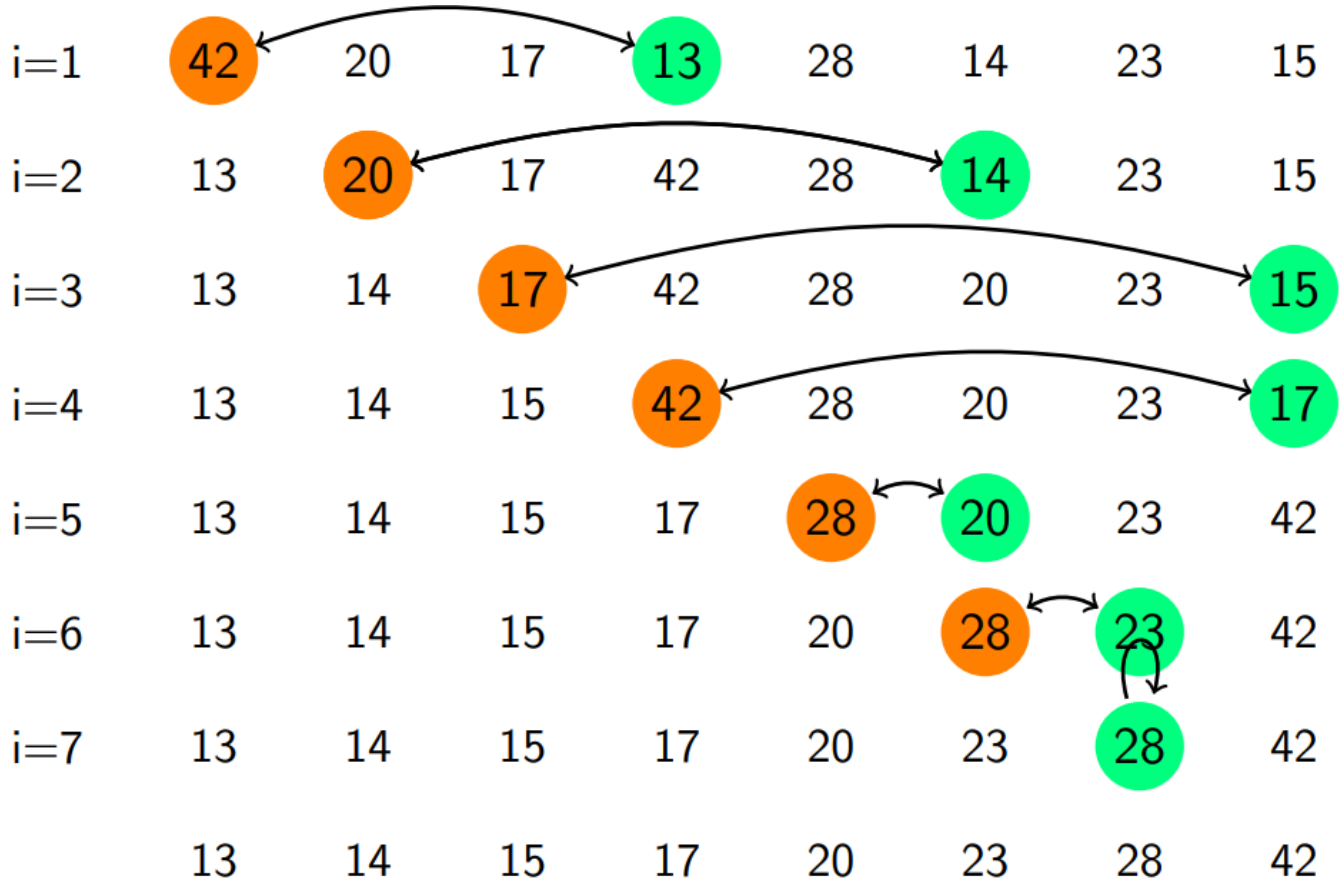
Ví dụ

Minh hoạ với dãy phần tử $a = \{42, 20, 17, 13, 28, 14, 23, 15\}$

Sắp xếp chọn

Ví dụ

Minh hoạ với dãy phần tử $a = \{42, 20, 17, 13, 28, 14, 23, 15\}$



Sắp xếp chọn

Phân tích thuật toán

- Trường hợp tốt nhất: 0 đổi chỗ (cách 2), $n-1$ đổi chỗ (cách 1), $\frac{n^2}{2}$ phép so sánh
- Trường hợp tồi nhất: $n-1$ phép đổi chỗ và $\frac{n^2}{2}$ phép so sánh
- Trường hợp trung bình: n phép đổi chỗ và $\frac{n^2}{2}$ phép so sánh

Ưu điểm: số lần đổi chỗ ít

Nội dung

1 Giới thiệu

2 Sắp xếp nổi bọt - Bubble sort

3 Sắp xếp chọn - Selection sort

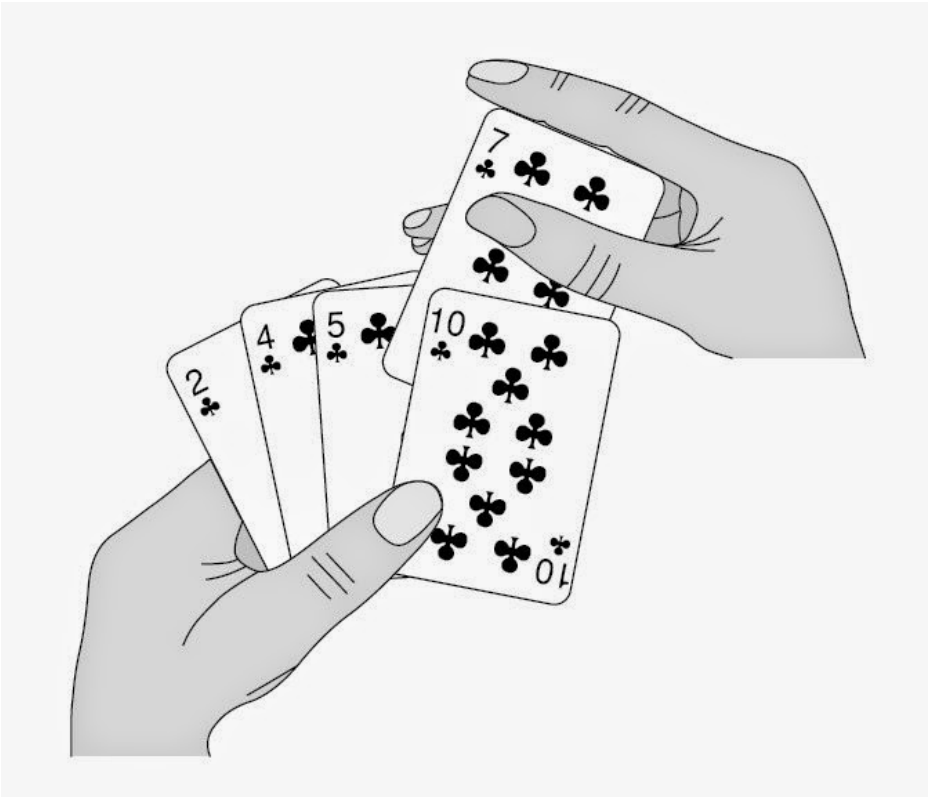
4 Sắp xếp chèn - Insertion sort

5 Sắp xếp trộn - Merge sort

6 Sắp xếp nhanh - Quick sort

Sắp xếp chèn

- Dựa trên kinh nghiệm của những người chơi bài
- Khi có $i-1$ lá bài được sắp xếp ở trên tay. Nếu rút thêm được lá bài thứ i thì sắp xếp lại thế nào?
- Cách làm là: sẽ so sánh lá bài mới với các lá bài thứ $i-1$, $i-2$,... để tìm ra vị trí " thích hợp" và "chèn vào"



Sắp xếp chèn

Shift Elements

Di chuyển các phần tử bên trái vị trí i để tìm vị trí đúng cho $a[i]$

```
void shiftElement(int a[], int j) {
    int value = a[j];
    while ((j > 0) && (a[j-1] > value)) {
        a[j] = a[j-1];
        j--;
    }
    a[j] = value;
}
```

Sắp xếp chèn

```
void insertionSort(int a[], int n) {
    int i;
    for (i = 1; i < n; i++) {
        if (a[i] < a[i-1]) {
            shiftElement(a, i);
        }
    }
}
```

Sắp xếp chèn

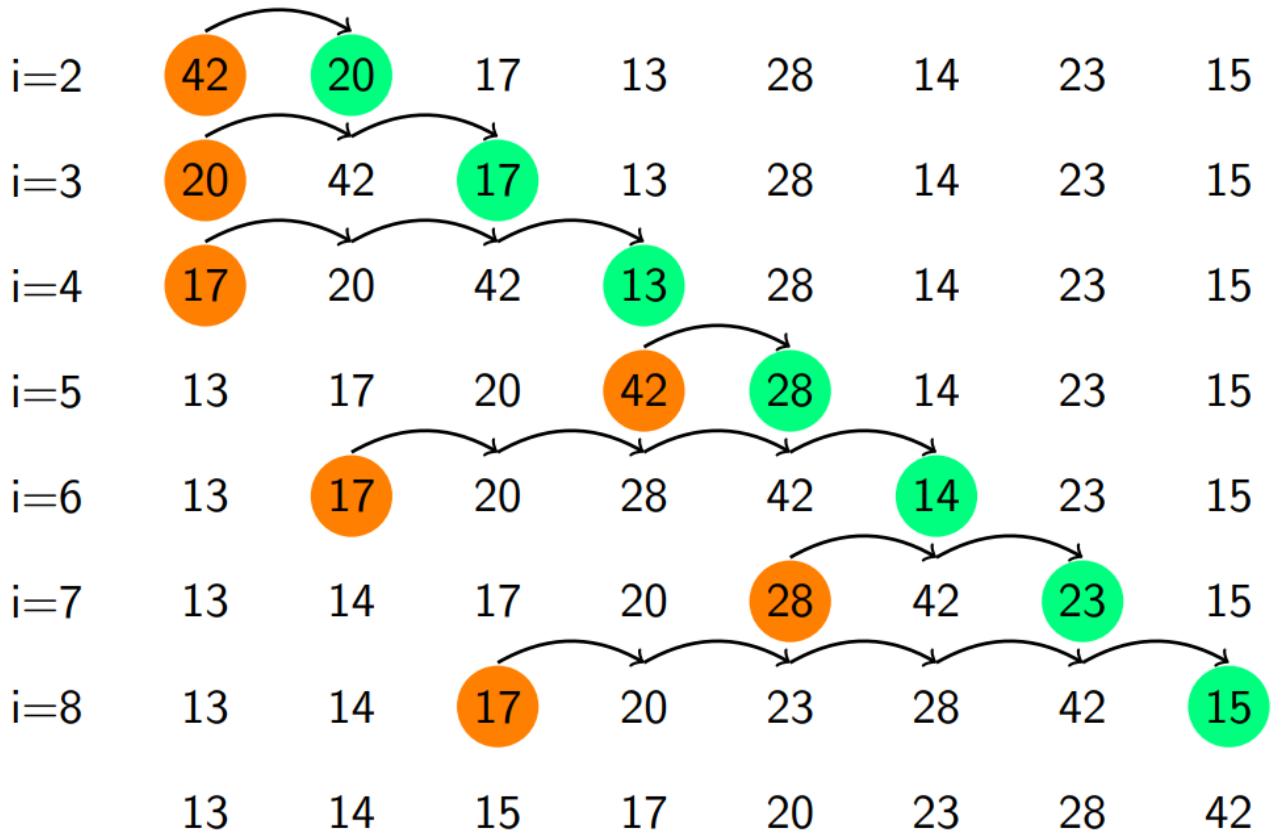
Ví dụ

Minh hoạ với dãy phần tử $a = \{42, 20, 17, 13, 28, 14, 23, 15\}$

Sắp xếp chèn

Ví dụ

Minh hoạ với dãy phần tử $a = \{42, 20, 17, 13, 28, 14, 23, 15\}$



Sắp xếp chèn

Phân tích thuật toán

- Sắp xếp chèn là tại chỗ và ổn định.
- Thời gian của thuật toán
  - Trường hợp tốt nhất: 0 có hoán đổi (dãy cho vào đã được sắp xếp)
  - Trường hợp tồi nhất: có $\frac{n^2}{2}$ hoán đổi và so sánh, khi dạy đầu vào có thứ tự ngược với chiều cần sắp xếp
  - Trường hợp trung bình: cần $\frac{n^2}{4}$ hoán đổi và so sánh
- Thuật toán tốt với dãy đã *gần được sắp xếp*, nghĩa là phần tử đưa vào gần với vị trí cần sắp xếp

Tổng kết ba thuật toán sắp xếp cơ bản

Thuật toán	**Insertion**	**Bubble**	**Selection**
Số lần so sánh			
Tốt nhất	$O(n)$	$O(n)$	$O(n^2)$
Trung bình	$O(n^2)$	$O(n^2)$	$O(n^2)$
Tệ nhất	$O(n^2)$	$O(n^2)$	$O(n^2)$
Số lần đổi chỗ			
Tốt nhất	0	0	$O(n)$
Trung bình	$O(n^2)$	$O(n^2)$	$O(n)$
Tệ nhất	$O(n^2)$	$O(n^2)$	$O(n)$