Bài 9.2: Thuật toán bubble sort

- ✓ Mục đích sử dụng
- ✓ Sắp xếp tăng dần
- ✓ Sắp xếp giảm dần
- ✓ Bubble sort tối ưu
- ✓ Bài tập thực hành

Mục đích sử dụng

- Sắp xếp là đặt các phần tử của một tập hợp theo thứ tự của một hoặc một số tiêu chí nào đó.
- Đây là chức năng thiết yếu trong hầu hết các ứng dụng, phần mềm.
- Giúp tối ưu chương trình, chuẩn hóa dữ liệu đầu vào, tạo đầu ra dễ đọc hiểu với con người.
- Ví dụ cần sắp xếp danh sách nhân viên theo lương giảm dần.
- Ví dụ cần sắp xếp sản phẩm của trang bán hàng theo thứ tự giá giảm dần…
- Thuật toán sắp xếp nổi bọt có độ phức tạp O(n^2)

Sắp xếp tăng dần

- Sau đây là mã giả của thuật toán sắp xếp nổi bọt theo thứ tự tăng dần của các phần tử:

```
// thuật toán sắp xếp nổi bọt
// arr: mảng đầu vào
// n: số phần tử của mảng
function bubbleSort(arr[], n):
    for(i từ 0 tới n - 2):
        for(j từ n - 1 tới i + 1):
            if(arr[j - 1] > arr[j]): // nếu phần tử đứng trước > đứng sau
                swap(arr[j], arr[j - 1]); // đổi chỗ hai phần tử trên
```

Sắp xếp tăng dần

- Triển khai của thuật toán sắp xếp bubble sort tăng dần:

```
// sắp xếp nổi bọt tăng dần
template<class T> void bubbleSort(T* arr, size_t size) {
    for (size_t i = 0; i < size - 1; i++)
    {
        for (size_t j = size - 1; j > i; j--)
        {   // nếu phần tử đứng trước lớn hơn phần tử đứng sau
            if (arr[j - 1] > arr[j]) {
                swap(arr[j], arr[j - 1]); // đổi chỗ
            }
        }
    }
}
```

Sắp xếp giảm dần

➢ Mã giả của thuật toán sắp xếp bubble sort giảm dần:

```
// thuật toán sắp xếp nổi bọt giảm dần
// arr: mảng đầu vào
// n: số phần tử của mảng
function bubbleSort(arr[], size):
    for(i từ 0 tới n - 2):
        for(j từ n - 1 tới i + 1):
            if(arr[j - 1] <>> arr[j]): // nếu phần tử đứng trước < đứng sau
                swap(arr[j], arr[j - 1]); // đổi chỗ hai phần tử trên
```

Sắp xếp giảm dần

- Mã thật của thuật toán sắp xếp bubble sort giảm dần:

```
// sắp xếp nổi bọt giảm dần
template<class T> void bubbleSortDESC(T* arr, size_t size) {
    for (size_t i = 0; i < size - 1; i++)
    {
        for (size_t j = size - 1; j > i; j--)
        {   // nếu phần tử đứng trước nhỏ hơn phần tử đứng sau
            if (arr[j - 1] < arr[j]) {
                swap(arr[j], arr[j - 1]); // đổi chỗ
            }
        }
    }
}
```

Bubble sort tối ưu

- Trong trường hợp nếu lần duyệt nào đó với một biến chạy i cố định mà vòng lặp bên trong với biến chạy j kết thúc, không xảy ra việc đổi chỗ cặp phần tử nào tức là lúc này tập hợp đã được sắp xếp đúng thứ tự.
- Do đó ta có thể dùng một biến đánh dấu để kiểm soát việc lặp của trường hợp này.
- Nếu vòng lặp thứ i nào đó kết thúc mà biến đánh dấu cho thấy không xảy ra việc đổi chỗ của bất kì cặp phần tử nào, ta break khỏi vòng lặp, kết thúc việc sắp xếp.
- Kết quả ta nhận được mảng đã sắp xếp.

Bubble sort tối ưu

➢ Mã giả của thuật toán sắp xếp nổi bọt tối ưu:

```
// thuật toán sắp xếp nổi bọt tối ưu
// arr: mảng đầu vào
// n: số phần tử của mảng
function bubbleSortOpt(arr[], n):
    isSwapped = true // biến đánh dấu sự đổi chỗ
    i = n - 1 // xuất phát từ cuối mảng
    while(i > 0): // khi i còn lớn hơn 0
        isSwapped = false // giả sử không xảy ra hành động đổi chỗ 2 phần tử
        for(j từ 0 tới i - 1):
            if(arr[j] > arr[j + 1]): // nếu phần tử đứng trước > đứng sau
                swap(arr[j], arr[j + 1]) // đổi chỗ hai phần tử
                isSwapped = true // đánh dấu đã xảy ra đổi chỗ
        if(isSwapped == false): // nếu kết thúc lần lặp nào đó mà
            break               // không xảy ra đổi chỗ thì break
        else:
            i--                 // giảm i để đến với lần lặp kế tiếp
```

Bubble sort tối ưu

➢ Mã thật của thuật toán sắp xếp nổi bọt tối ưu:

```
// sắp xếp nổi bọt tối ưu tăng dần
template<class T> void bubbleSortOpt(T* arr, size_t size) {
    bool isSwapped; // biến đánh dấu việc xảy ra hành động đổi chỗ
    size_t i = size - 1; // xuất phát từ cuối mảng
    while (i > 0) { // lặp chừng nào chưa xét hết các phần tử mảng
        isSwapped = false;
        for (size_t j = 0; j < i; j++)
        {
            if (arr[j] > arr[j + 1]) {
                swap(arr[j], arr[j + 1]);
                isSwapped = true;
            }
        }
        if (!isSwapped) {
            break;
        }
        else {
            i--;
        }
    }
}
```

Nội dung tiếp theo

Thuật toán sắp xếp chọn – Selection sort