Bubble Sort

Table of contents

Idea

Xuất phát từ đầu dãy hoặc cuối dãy, đổi chỗ các cặp phần tử liền kề để đưa phần tử lớn hơn trong cặp phần tử đó về đúng cuối dãy hiện hành. Sau đó sẽ không xét đến nó ở bước tiếp theo, do vậy ở lần xử lý thứ i thì vị trí cuối dãy là n - i - 1.

Cần phân biệt rõ ràng với Interchange Sort vì thuật toán Bubble Sort không so sánh tất cả các cặp tồn tại mà chỉ so sánh các cặp nghịch thế liền kề với nhau.

Properties

Điểm mạnh

Là một thuật toán dễ cài đặt, dễ hiểu và hoạt động tốt cho mảng có số lượng phần tử nhỏ.

Điểm yếu

Không hiệu quả đối với mảng có số lượng phần tử lớn, thường thì Bubble Sort được dùng trong giảng dạy hơn là áp dụng thực tiễn.

> Khi nào nên sử dụng?

- Mảng đầu vào có kích thước cực nhỏ và hoặc gần như có thứ tự hoặc đã có thứ tự.

Complexity Analysis

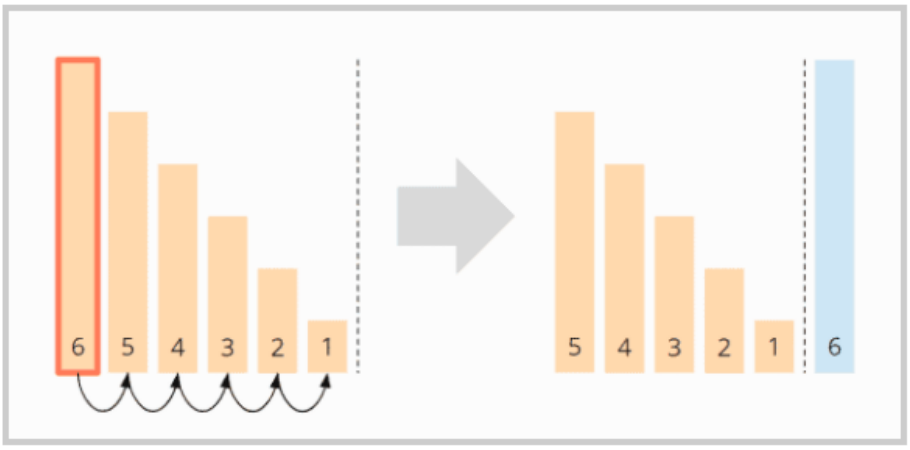
Tham khảo happycoders.

Bubble Sort không dễ chứng minh và phân tích trực tiếp như Selection Sort và Insertion Sort. Vì vậy ta sẽ chia nó ra ba trường hợp để phân tích

Best case
Do thuật toán sử dụng cờ lệnh và biết được khi nào mảng đã được sắp xếp, nên khi mảng đầu vào là dãy đã được sắp xếp thì cũng chính là trường hợp tốt nhất của thuật toán.

Lúc này thuật toán sẽ quét qua từng phần tử rồi dừng nên độ phức tạp sẽ là tuyến tính $O(n)$.

Worst case
Trường hợp xấu nhất xảy ra khi dữ liệu mà ta muốn sắp bị đảo ngược, chẳng hạn giảm dần khi ta muốn sắp tăng dần. Lúc này, với mỗi vòng lặp i, ta cần n - i lần hoán vị để đưa phần tử thứ i về đúng vị trí cuối mảng.



Và ta có n vòng lặp như vậy, số lần thực hiện hoán vị của hai vòng lặp lồng nhau này là:

$$\frac{n(n-1)}{2}$$

(Có thể xem lại bài Properties để biết lý do). Do đó mà độ phức tạp sẽ là $O(n^2)$.

Average case
Phân tích chi tiết ở đây.

Complexity

Cases	Complexity
Best case	$O(n)$
Worst case	$O(n^2)$
Average case	$O(n^2)$

Space Complexity: $O(1)$.

Code

```
void bubbleSort(int *a,int n)
{
    for (int i = 0; i < n - 1; i++)
    {
        bool isSorted = true;// Tạo ra một cờ lệnh

        //Do chúng ta so sánh hai phần tử j và j + 1,nên: j + 1 < n
        //Sau mỗi vòng lặp thì giảm đi một phần tử nên trừ đi i: j + 1 < n - i
        //Chuyển vế đổi dấu ta có điều kiện là j < n - i - 1

        //Tức là cuối mảng sau mỗi lần lặp xem như đã sắp xếp.
        for(int j = 0; j < n - i - 1; j++){
            if(a[j] < a[j + 1])
            {
                swap(a[j], a[j + 1]);
                isSorted = false;
            }
        }
```

```
            //Nếu như không có sự thay đổi nào thì dừng sắp xếp.
            if(isSorted) return;
        }
    }
```