Rồi, ta đến với **Bubble Sort** – thuật toán tưởng đơn giản nhưng lại là **bài học nền tảng rất hay** để rèn tư duy **so sánh – hoán vị – ổn định – tối ưu hóa vòng lặp**.

Nhiều bạn hay bỏ qua Bubble Sort vì "nó chậm", nhưng hiểu **sâu Bubble Sort** giúp em giải thích được nhiều kiến thức nền trong phỏng vấn: từ tư duy sort đến tối ưu hoán vị, nhận diện ổn định, best case,...

**🎓 CHUYÊN SÂU VỀ BUBBLE SORT**

**✅ 1. Ý tưởng cơ bản**

So sánh từng cặp phần tử kề nhau, nếu sai thứ tự thì hoán đổi. Lặp lại nhiều lượt cho đến khi không cần đổi nữa.

**📦 Tưởng tượng:**

* Như các **bong bóng khí nổi lên mặt nước** → phần tử lớn “bị đẩy” về cuối mảng.
* Mỗi lượt đẩy phần tử lớn nhất còn lại về đúng chỗ.

**🔁 2. Ví dụ**

Mảng: [5, 3, 1, 4]

* Lượt 1: [3,1,4,5]
* Lượt 2: [1,3,4,5]
* Lượt 3: [1,3,4,5] (không đổi nữa → xong)

**🧠 3. Code chuẩn Bubble Sort**

void bubbleSort(int a[], int n) {

for (int i = 0; i < n - 1; i++) {

bool swapped = false; // kiểm tra đã hoán đổi chưa

for (int j = 0; j < n - i - 1; j++) {

if (a[j] > a[j + 1]) {

swap(a[j], a[j + 1]);

swapped = true;

}

}

if (!swapped) break; // nếu không hoán đổi nữa thì mảng đã sắp xếp

}

}

**📊 4. Độ phức tạp**

| **Trường hợp** | **Độ phức tạp** | **Ghi chú** |
| --- | --- | --- |
| **Best Case** | O(n) | Mảng đã sắp xếp và có swapped == false |
| **Average** | O(n²) | Phải so sánh gần đủ |
| **Worst Case** | O(n²) | Mảng ngược hoàn toàn |

**✅ 5. Tính ổn định**

Bubble Sort **ổn định**: phần tử bằng nhau không đổi vị trí tương đối.

📌 Vì swap chỉ xảy ra khi a[j] > a[j+1], chứ không dùng >=, nên không làm đảo thứ tự phần tử bằng nhau.

**📦 6. Ưu và nhược điểm**

**✅ ƯU:**

* Cài đặt cực kỳ dễ.
* Ổn định.
* Có thể tối ưu với swapped == false.

**❌ NHƯỢC:**

* Chạy chậm với dữ liệu lớn (O(n²)).
* Rất nhiều lần so sánh không cần thiết.

**💼 7. Khi nào nên dùng Bubble Sort?**

| **Trường hợp** | **Vì sao** |
| --- | --- |
| Mảng cực nhỏ (dưới 10 phần tử) | Dễ code, dễ hiểu |
| Dạy học thuật toán | Hiển thị từng bước dễ |
| Mảng gần sắp xếp | Có thể đạt O(n) nhờ swapped == false |

**🧠 8. So sánh với Selection và Insertion Sort**

| **Thuật toán** | **Best Case** | **Average** | **Worst** | **Stable?** | **Swap nhiều?** |
| --- | --- | --- | --- | --- | --- |
| **Bubble Sort** | O(n) | O(n²) | O(n²) | ✅ Có | ❗Rất nhiều |
| Selection | O(n²) | O(n²) | O(n²) | ❌ Không | Ít swap |
| Insertion | O(n) | O(n²) | O(n²) | ✅ Có | Trung bình |

**🧪 9. Câu hỏi phỏng vấn liên quan Bubble Sort**

“Tại sao Bubble Sort có O(n) trong best case?”

→ Vì nếu mảng đã sắp xếp sẵn, biến swapped sẽ luôn là false → chỉ duyệt 1 vòng.

“Bubble Sort có ổn định không?”  
→ ✅ Có.

“Khi nào chọn Bubble Sort?”  
→ Khi cần sort gấp mảng cực nhỏ hoặc trong giảng dạy/biểu diễn bước sắp xếp.

**🎯 10. Bài tập gợi ý**

**Bài 1:**

Viết Bubble Sort nhưng in ra mảng sau mỗi lần hoán vị.

**Bài 2:**

Đếm tổng số lần swap cần để sắp xếp mảng (bằng Bubble Sort).

**✅ Tổng kết nhanh:**

| **Đặc điểm** | **Bubble Sort** |
| --- | --- |
| Độ phức tạp | O(n²) thường, O(n) best case |
| Ổn định | ✅ Có |
| In-place | ✅ Có |
| Khi dùng | Mảng nhỏ, biểu diễn trực quan |
| Ưu điểm chính | Đơn giản, dễ học |
| Nhược điểm lớn | Chậm, nhiều swap |

Em muốn thầy minh họa cụ thể từng bước chạy Bubble Sort với 1 mảng ví dụ, hay chuyển sang học **Heap Sort / Counting Sort / Radix Sort** tiếp theo?