Rồi, thầy hiểu rồi – ta tiếp tục **giảng sâu về Insertion Sort** nhé! Đây là thuật toán cực hay, tuy đơn giản nhưng **thể hiện tư duy sắp xếp "giống con người làm ngoài đời thật"**. Nếu hiểu kỹ Insertion Sort, em sẽ rất dễ hiểu về các thuật toán tối ưu hơn sau này như Shell Sort, Binary Insertion Sort,…

**🎓 CHUYÊN SÂU VỀ INSERTION SORT**

**✅ 1. Ý tưởng cơ bản**

"Lấy từng phần tử trong mảng, chèn nó vào đúng vị trí trong dãy đã sắp xếp bên trái."

**📦 Cách hình dung:**

* Giống như **xếp bài trên tay**: mỗi lần rút 1 lá bài mới, ta đưa nó vào đúng vị trí trong nhóm bài đã xếp bên tay trái.

**🔁 Ví dụ: [5, 2, 4, 6, 1]**

* Lần 1: xếp 2 vào dãy [5] → [2,5]
* Lần 2: xếp 4 vào [2,5] → [2,4,5]
* Lần 3: xếp 6 vào [2,4,5] → [2,4,5,6]
* Lần 4: xếp 1 vào [2,4,5,6] → [1,2,4,5,6]

**🧠 2. Code Insertion Sort**

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

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

int key = a[i];

int j = i - 1;

// Dịch các phần tử lớn hơn key sang phải

while (j >= 0 && a[j] > key) {

a[j + 1] = a[j];

j--;

}

a[j + 1] = key; // Chèn key vào đúng vị trí

}

}

**📊 3. Độ phức tạp**

| **Trường hợp** | **Số bước** | **Độ phức tạp** |
| --- | --- | --- |
| Best Case | Không phải dịch | O(n) ✅ |
| Average Case | Dịch trung bình | O(n²) |
| Worst Case | Dịch toàn bộ mỗi lần | O(n²) ❌ |

**🔎 Giải thích:**

* Best case khi mảng **đã sắp xếp tăng dần** → không phải dịch gì cả.
* Worst case khi mảng **giảm dần hoàn toàn** → mỗi lần chèn phải dịch hết dãy bên trái.

**⚖️ 4. So sánh với Selection Sort**

| **Thuật toán** | **Số lần so sánh** | **Số lần hoán vị** | **Ổn định** |
| --- | --- | --- | --- |
| Selection Sort | O(n²) | Tối đa n-1 | ❌ Không |
| Insertion Sort | O(n²) | Có thể nhiều | ✅ Có |

👉 Insertion Sort tuy vẫn là O(n²) nhưng **thực tế chạy nhanh hơn Selection** trong nhiều TH, đặc biệt là khi mảng **gần sắp xếp rồi**.

**✅ 5. Tính ổn định**

**Stable Sort**: Giữ nguyên thứ tự phần tử bằng nhau.  
**Insertion Sort là ổn định** vì chỉ **dịch** chứ không hoán đổi lung tung.

**📦 6. Ưu – nhược điểm**

**✅ ƯU ĐIỂM:**

* Dễ cài, dễ hiểu.
* Tốt với mảng **gần sắp xếp**.
* Ổn định (giữ thứ tự phần tử bằng nhau).
* Không cần bộ nhớ phụ (O(1)).

**❌ NHƯỢC ĐIỂM:**

* O(n²) với mảng đảo ngược.
* Không dùng cho dữ liệu lớn.

**🧠 7. Khi nào nên dùng Insertion Sort?**

| **Tình huống** | **Vì sao** |
| --- | --- |
| Mảng nhỏ (10–20 phần tử) | Code đơn giản, đủ nhanh |
| Mảng gần sắp xếp | Vì chỉ cần ít bước dịch |
| Cần giữ thứ tự gốc | Vì ổn định |

**💡 8. Mẹo phỏng vấn**

“Khi nào Insertion Sort chạy O(n)?”

→ Khi mảng **đã sắp xếp sẵn**

“Khi nào dùng Insertion Sort thay vì Merge Sort?”

→ Khi mảng **nhỏ** hoặc **gần sắp xếp**.

“Insertion Sort có ổn định không?”

→ ✅ Có. Vì chỉ dịch chứ không hoán vị.

**✍️ Thầy đề xuất em luyện 1 bài đơn giản như sau:**

**🔧 Đề:**

Viết hàm sắp xếp mảng a[] có tối đa 100 phần tử bằng Insertion Sort. Sau mỗi bước chèn, in ra trạng thái mảng.

Nếu em cần, thầy có thể ra thêm bài kiểm tra nhỏ, hoặc giải thích cặn kẽ từng dòng code chạy như thế nào.

Em muốn thầy tiếp tục với **Merge Sort**, hay luyện 1–2 bài code về Insertion Sort và Selection Sort trước?