Câu hỏi này rất thực tế và thông minh, vì **học thuật toán là một chuyện**, còn **biết khi nào dùng chúng là tư duy của người kỹ sư phần mềm giỏi.**  
Thầy sẽ chia nhỏ theo từng loại tình huống **thường gặp trong thực tế/phỏng vấn kỹ sư fresher tại VNG, FPT, TMA, Shopee,...** và gợi ý **thuật toán sắp xếp hoặc tìm kiếm phù hợp nhất.**

**📌 1. Khi nào chọn thuật toán nào? (tổng hợp)**

**✅ 1. Mảng nhỏ (n ≤ 20)**

| **Chọn** | **Vì sao** |
| --- | --- |
| **Insertion Sort** | Chạy nhanh, dễ cài |
| **Bubble Sort** | Dễ kiểm tra, trực quan |
| **Selection Sort** | Dễ hiểu, ít swap |

✅ Dùng khi bài cho giới hạn nhỏ, yêu cầu minh họa sắp xếp từng bước.

**✅ 2. Mảng lớn (n > 10⁴) – cần hiệu năng**

| **Chọn** | **Vì sao** |
| --- | --- |
| **Quick Sort** | Thường chạy nhanh nhất trên thực tế |
| **Merge Sort** | Luôn O(n log n), ổn định |
| **Heap Sort** | Không dùng thêm bộ nhớ, O(n log n) |

🚫 Tránh dùng: Bubble, Insertion, Selection (quá chậm)

**✅ 3. Mảng gần sắp xếp (hầu hết đã đúng thứ tự)**

| **Chọn** | **Vì sao** |
| --- | --- |
| **Insertion Sort** | Chỉ cần chèn lại vài phần tử → O(n) trong best case |
| **Tối ưu hóa Bubble Sort** | Dừng sớm nếu không có swap |

🚫 Không cần dùng Merge hay Quick vì chi phí chia/merge không đáng.

**✅ 4. Cần giữ thứ tự phần tử bằng nhau (stable sort)**

| **Chọn** | **Vì sao** |
| --- | --- |
| **Merge Sort** | Ổn định, nhanh |
| **Insertion Sort** | Ổn định, dễ code |
| **Counting/Radix Sort** | Ổn định nếu cài đúng |

🚫 Không chọn Quick/Heap/Selection nếu yêu cầu ổn định.

**✅ 5. Dữ liệu có miền giá trị nhỏ (ví dụ 0 ≤ a[i] ≤ 100)**

| **Chọn** | **Vì sao** |
| --- | --- |
| **Counting Sort** | O(n + k), nhanh cực nhanh |
| **Radix Sort** | Nếu giá trị là số lớn nhiều chữ số |

🚫 Không cần dùng các thuật toán so sánh thông thường

**✅ 6. Không được dùng bộ nhớ phụ (in-place required)**

| **Chọn** | **Vì sao** |
| --- | --- |
| **Quick Sort** | In-place, hiệu năng tốt |
| **Heap Sort** | In-place, ổn định không cần |

🚫 Merge Sort cần bộ nhớ phụ → tránh

**✅ 7. Duyệt dữ liệu tuyến tính hoặc 1 lần**

| **Chọn** | **Vì sao** |
| --- | --- |
| **Linear Search** | Dữ liệu chưa sắp xếp |
| **Counting Sort** | Đếm, tần suất, histogram |
| **DFS/BFS** | Duyệt đồ thị, cây, hệ thống, mạng lưới |

**✅ 8. Tìm kiếm phần tử (giá trị x) trong mảng**

| **Tình huống** | **Chọn** |
| --- | --- |
| Mảng chưa sắp xếp | **Linear Search** |
| Mảng đã sắp xếp | **Binary Search** (O(log n)) |
| Dữ liệu cây | **BST search**, **Trie** |

**✅ 9. Dữ liệu đến theo luồng (không được lưu toàn bộ)**

| **Chọn** | **Vì sao** |
| --- | --- |
| **Heap (min/max)** | Lưu top-k, stream processing |
| **Counting Sort** | Khi chỉ cần đếm |

**📌 2. Tổng kết: bảng chọn nhanh**

| **Nhu cầu** | **Thuật toán gợi ý** |
| --- | --- |
| Dễ cài, mảng nhỏ | Insertion, Bubble |
| Tốc độ cao, mảng lớn | Quick, Merge, Heap |
| Ổn định | Merge, Insertion, Counting |
| Ít bộ nhớ phụ | Quick, Heap |
| Miền giá trị nhỏ | Counting, Radix |
| Mảng gần sắp xếp | Insertion |
| Cần tìm kiếm nhanh trong mảng đã sort | Binary Search |
| Duyệt đồ thị | DFS, BFS |

**🎯 Thầy chốt lại:**

💡 Khi chọn thuật toán, đừng chọn theo "cái nào hay hơn", mà hãy hỏi:

* Dữ liệu lớn hay nhỏ?
* Có sắp xếp sẵn không?
* Có yêu cầu ổn định không?
* Có cần tối ưu bộ nhớ không?

Nếu em muốn thầy **ra 5 bài tập thực tế**, yêu cầu em chọn đúng thuật toán để giải, thầy có thể làm ngay.

Em có muốn luyện chọn thuật toán qua bài tập tình huống không?

Tuyệt vời! Dưới đây là **10 bài tập tình huống thực tế**, mỗi bài yêu cầu em **chọn đúng thuật toán** (hoặc nhóm thuật toán) để giải quyết hiệu quả, **giải thích tại sao chọn** và nếu muốn, em có thể viết luôn code sau.

**🧪 10 BÀI TẬP CHỌN THUẬT TOÁN**

**✅ Bài 1:**

Mảng a[] gồm 1 triệu số nguyên đã sắp xếp. Hãy kiểm tra xem số x có tồn tại trong mảng hay không.

➡ **Yêu cầu:** Tìm kiếm nhanh, dữ liệu đã sort.

**✅ Bài 2:**

Mảng a[] gồm 100 phần tử, nhưng đã gần như sắp xếp rồi (chỉ vài phần tử bị lệch). Hãy sắp xếp lại mảng.

➡ **Yêu cầu:** Mảng nhỏ, gần đúng thứ tự.

**✅ Bài 3:**

Cần sắp xếp 100 triệu số nguyên có giá trị từ 0 đến 10^6.

➡ **Yêu cầu:** Số lượng lớn, giá trị nhỏ.

**✅ Bài 4:**

Sắp xếp một mảng các đối tượng nhân viên (gồm ID và tên) **theo ID**, **giữ nguyên thứ tự nếu ID trùng**.

➡ **Yêu cầu:** Sắp xếp ổn định.

**✅ Bài 5:**

Một danh sách gồm 10 phần tử ngẫu nhiên. Viết code sắp xếp để minh họa từng bước chạy ra màn hình.

➡ **Yêu cầu:** Hiển thị từng bước, dễ nhìn.

**✅ Bài 6:**

Một đồ thị có n đỉnh và m cạnh. Hãy duyệt qua toàn bộ đỉnh theo chiều sâu.

➡ **Yêu cầu:** Duyệt đồ thị theo DFS.

**✅ Bài 7:**

Tìm phần tử lớn thứ k trong một mảng a[] rất lớn (n = 10^6), không cần sắp xếp toàn bộ.

➡ **Yêu cầu:** Tối ưu thời gian, không cần toàn bộ sort.

**✅ Bài 8:**

Mảng a[] gồm các số thực có 5 chữ số, cần sắp xếp nhanh nhất có thể.

➡ **Yêu cầu:** Dữ liệu không ổn định, cần hiệu suất cao.

**✅ Bài 9:**

Mỗi lần nhập một số nguyên từ bàn phím (stream), cần duy trì 10 phần tử nhỏ nhất từ trước đến giờ.

➡ **Yêu cầu:** Stream processing, top-k nhỏ nhất.

**✅ Bài 10:**

Một mảng có tối đa 500 phần tử, yêu cầu viết thuật toán sắp xếp ổn định, dễ cài, không cần tối ưu cao.

➡ **Yêu cầu:** Ưu tiên ổn định, đơn giản.

**📌 Cách em luyện:**

* Em có thể trả lời từng bài: chọn thuật toán nào + giải thích.
* Sau đó thầy sẽ nhận xét, sửa, và nếu cần thì minh họa luôn bằng code hoặc phân tích thời gian chạy.

Em muốn làm từng bài một hay trả lời cả 10 bài luôn? Thầy sẵn sàng chấm và sửa chi tiết!