**Prompt 1**

**“So sánh Bubble Sort, Merge Sort và Quick Sort theo tiêu chí: độ phức tạp thời gian, độ phức tạp bộ nhớ, và tình huống áp dụng.”**

AI sẽ thường trả lời dạng **đoạn văn**, ví dụ:

**Kết quả dạng văn bản:**

* **Bubble Sort** có độ phức tạp thời gian O(n²) trong hầu hết các trường hợp vì phải so sánh từng cặp phần tử, bộ nhớ O(1), thường chỉ dùng để dạy học và với dữ liệu rất nhỏ.
* **Merge Sort** có độ phức tạp thời gian ổn định O(n log n) kể cả trường hợp xấu nhất, nhưng tốn bộ nhớ O(n) vì cần tạo mảng phụ, phù hợp với dữ liệu lớn cần ổn định.
* **Quick Sort** trung bình O(n log n) nhưng trường hợp xấu nhất O(n²), bộ nhớ O(log n), nhanh và hiệu quả trong thực tế, phù hợp dữ liệu lớn nhưng không đảm bảo ổn định.

**Prompt 2 (yêu cầu cải thiện)**

**“Trình bày kết quả dưới dạng bảng.”**

**Bảng so sánh :**

| **Thuật toán** | **Độ phức tạp thời gian (Time Complexity)** | **Độ phức tạp bộ nhớ (Space Complexity)** | **Khi nào nên dùng** |
| --- | --- | --- | --- |
| **Bubble Sort** | Best: O(n) (dữ liệu gần như sẵn sàng) Avg/Worst: O(n²) | O(1) – In-place | • Dữ liệu rất nhỏ hoặc dùng để dạy/làm ví dụ • Khi cần code đơn giản, dễ hiểu |
| **Merge Sort** | Best/Avg/Worst: O(n log n) – ổn định | O(n) – dùng bộ nhớ phụ để trộn | • Dữ liệu lớn • Cần thuật toán **ổn định** • Trong xử lý file/danh sách liên kết |
| **Quick Sort** | Best/Avg: O(n log n) Worst: O(n²) nếu chọn pivot xấu | O(log n) – In-place (gọi đệ quy) | • Dữ liệu lớn • Cần chạy thực tế nhanh và sử dụng ít bộ nhớ • Thường dùng mặc định trong thư viện chuẩn |