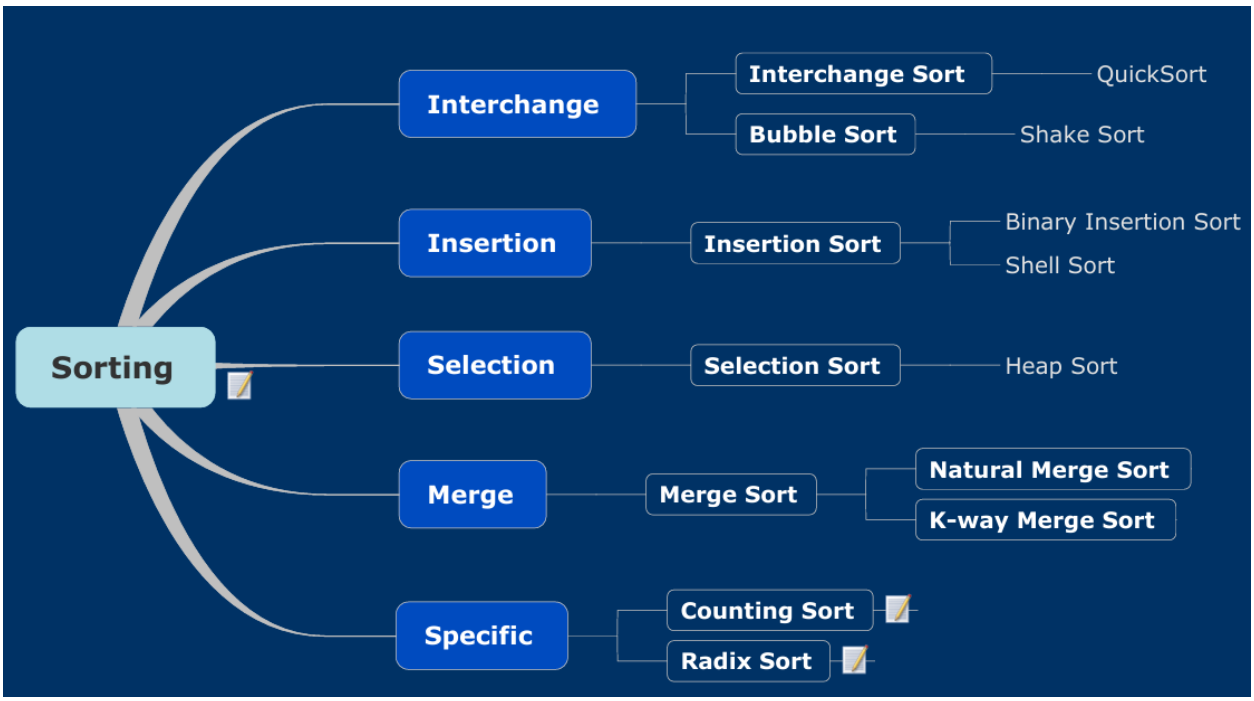
SORTING ALGORITHMS

Trong lập trình, bài toán sắp xếp và tìm kiếm là một vấn đề thú vị và không hề đơn giản vì thế nên ta sẽ cùng đi tìm hiểu và những thuật toán sắp xếp căn bản và thuật toán nâng cao hơn nhé. Thứ tự các thuật toán sắp xếp sẽ được liệt kê như sơ đồ sau:



Trong sơ đồ trên, có 4 phân loại là Interchange, Insertion, Selection, Merge là 4 kiểu thao tác trên mảng, từ đó ta sẽ được ra được các thuật toán căn bản nhất và tối ưu lên (những thuật toán ở cột cuối cùng).

Các thuật toán sắp xếp bắt nguồn từ Interchange

1. Thuật toán sắp xếp cơ bản Interchange và sự cải tiến Quick Sort

Interchange sort	Quick Sort
Nhận xét: Mảng chưa sắp xếp sẽ có **nghịch thế** nên để sắp xếp một dãy số, ta có thể xét các nghịch thế có trong dãy và làm triệt tiêu dần chúng đi. **Thuật toán:** 1.Xuất phát từ đầu dãy, tìm tất cả nghịch thế chứa phần tử này, triệt tiêu chúng bằng cách đổi chỗ phần tử này với phần tử tương ứng trong cặp nghịch thế 2.Lặp lại xử lý trên với các phần tử tiếp theo trong dãy. **Khuyết điểm:** Interchange Sort có thể thực hiện **rất nhiều lần hoán đổi không cần thiết**, làm giảm hiệu suất.	Để cải tiến thuật toán Interchange Sort, Quick Sort đã được áp dụng **kĩ thuật chia để trị** để cải tiến nhưng vẫn giữ được chính bản chất là hoán đổi vị trí. Thuật toán này chọn một phần tử làm **pivot** (điểm chốt), sau đó sắp xếp các phần tử nhỏ hơn pivot về bên trái và các phần tử lớn hơn pivot về bên phải. Tiếp tục áp dụng đệ quy cho hai phần này cho đến khi mảng được sắp xếp hoàn toàn. **Thuật toán:** 1. Chọn một phần tử trong mảng làm pivot. 2. Đưa các phần tử nhỏ hơn pivot về bên trái, lớn hơn pivot về bên phải. 3. Áp dụng Quick Sort cho hai nửa mảng (bên trái và bên phải của pivot)

2. Bubble sort và Shake sort

Bubble sort	Shake sort
Thuật toán: 1.Xuất phát từ cuối dãy, đổi chỗ các cặp phần tử kế cận để đưa phần tử nhỏ hơn trong cặp phần tử đó về vị trí đầu dãy hiện hành, sau đó sẽ không xét đến nó ở bước tiếp theo 2.Ở lần xử lý thứ i có vị trí đầu dãy là i 3.Lặp lại xử lý trên cho đến khi không còn cặp phần tử nào để xét **Khuyết điểm:** Các phần tử nhỏ được đưa về vị trí đúng rất nhanh, trong khi các phần tử lớn lại được đưa về vị trí đúng rất chậm	Nhận thấy Bubble Sort chỉ di chuyển phần tử lớn về cuối dãy trong mỗi lượt quét. Nếu phần tử nhỏ đang ở cuối mảng, thuật toán phải đợi nhiều lần quét để đưa nó về đúng vị trí. Cải tiến: 1. Shake Sort quét hai chiều (từ trái → phải rồi từ phải → trái), giúp di chuyển cả phần tử lớn nhất về cuối và phần tử nhỏ nhất về đầu trong một lượt quét. 2. Nhờ vậy, Shake Sort giảm số lần quét không cần thiết và giúp phần tử nhỏ nhanh chóng về đúng vị trí. Tuy nhiên, Shake Sort vẫn chậm với mảng lớn vì độ phức tạp O(n^2).

Thuật toán sắp xếp bắt nguồn từ Insertion

Insertion Sort	
Ý tưởng chính: 1. Tìm cách chèn phần tử a[i] vào vị trí thích hợp của đoạn đã được sắp để có dãy mới a[0] , a[1] ,... , a[i-1] trở nên có thứ tự 2. Vị trí này chính là pos thỏa : a[pos-1] <= a[i]< a[pos] (1 <= pos <= i)	
Binary Insertion Sort	**Shell Sort**
Nhận thấy khi tìm kiếm vị trí chèn phải duyệt cả mảng. Cải tiến của Insertion Sort bằng cách sử dụng **tìm kiếm nhị phân** để xác định vị trí chèn, giúp giảm số lần so sánh từ O(n) xuống O(log n)	Nhận thấy hiệu suất kém khi xử lý các phần tử ở xa nhau trong mảng của Insertion sort Thay vì hoán đổi từng phần tử liền kề, Shell Sort sắp xếp các phần tử cách xa nhau trước, sau đó dần giảm khoảng cách về 1. Giúp phần tử di chuyển nhanh hơn về đúng vị trí → Giảm số lần hoán đổi không cần thiết.

Thuật toán sắp xếp bắt nguồn từ Merge

Merge sort	
Ý tưởng: 1. Chia mảng thành hai nửa liên tiếp đến khi mỗi phần chỉ còn 1 phần tử. 2. Sắp xếp từng nửa nhỏ bằng cách gọi đệ quy. 3. Gộp hai mảng con đã sắp xếp thành một mảng duy nhất theo thứ tự đúng. Natural Merge Sort và K-way Merge Sort là hai thuật toán này đều là **biến thể của Merge Sort**, nhưng chúng cải tiến theo những cách khác nhau để tối ưu hóa hiệu suất trong các tình huống cụ thể.	
Natural Merge Sort	**K-way Merge Sort**
Tận dụng các dãy con đã được sắp xếp tự nhiên trong mảng ban đầu thay vì chia mảng cố định như Merge Sort thông thường. -> Giảm số lần chia và trộn, giúp tăng tốc độ sắp xếp nếu dữ liệu đã có phần nào đó được sắp xếp.	Mở rộng Merge Sort truyền thống, thay vì chia thành 2 mảng con, ta chia thành K mảng con. Tận dụng thuật toán K-way merge để trộn các dãy con lại với nhau một cách hiệu quả. Hữu ích khi cần sắp xếp dữ liệu lớn không thể chứa trong bộ nhớ RAM (ví dụ: sắp xếp file trên đĩa).

Thuật toán sắp xếp bắt nguồn từ Selection

Selection sort	Heap sort
Tìm phần tử nhỏ nhất trong mảng và hoán đổi nó lên đầu, lặp lại cho phần còn lại. **Nhược điểm:** Mỗi bước tìm min/max mất O(n) → Toàn bộ thuật toán mất O(n^2) trong trường hợp xấu nhất.	Sử dụng Heap (cấu trúc dữ liệu cây nhị phân hoàn chỉnh) để tìm phần tử lớn nhất/nhỏ nhất nhanh hơn trong O(log n).

Specific

Counting Sort	Radix Sort
Counting Sort là một thuật toán sắp xếp không dựa trên so sánh. Nó hoạt động bằng cách đếm số lần xuất hiện của mỗi phần tử trong mảng đầu vào, sau đó sử dụng thông tin này để sắp xếp mảng. Vậy nên số cần sắp xếp có phạm vi giá trị nhỏ với tốc độ cực nhanh O(n+k) và k (phạm vi giá trị).	Radix Sort là một thuật toán sắp xếp hoạt động bằng cách nhóm các chữ số theo cùng một giá trị chữ số (ví dụ: hàng đơn vị, hàng chục, …) trước khi sắp xếp các phần tử theo thứ tự tăng dần hoặc giảm dần. Đầu tiên, thuật toán sẽ sắp xếp các phần tử dựa trên giá trị ở hàng đơn vị. Sau đó, nó tiếp tục sắp xếp dựa trên giá trị ở hàng chục, … Quá trình này được lặp lại cho đến khi tất cả các chữ số có ý nghĩa đã được xét hết.