Giải thích các thuật toán

1. Bubble Sort (Sắp xếp nổi bọt)

Bubble Sort là giải thuật sắp xếp cơ bản, ý tưởng là dựa trên việc so sánh các cặp phần tử kề nhau và tráo đổi thứ tự nếu chúng không theo thứ tự yêu cầu

Giải thuật này có tốc độ chậm nhất trong các giải thuật sắp xếp do chúng chỉ đổi chỗ các phần tử kề nhau nên số lần đổi chỗ sẽ rất nhiều. Không phù hợp với các tập dữ liệu lớn

Ví dụ: cho dãy a = [4, 1, 2, 5]

|  |  |  |  |
| --- | --- | --- | --- |
| 4 | 1 | 2 | 5 |

0 1 2 3

So sánh a[0] > a[1] => đổi chỗ

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 4 | 2 | 5 |

0 1 2 3

So sánh a[1] > a[2] => đổi chỗ

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 4 | 5 |

So sánh a[2] < a[3] => Giữ nguyên

Ở đây, ta thấy dãy đã được sắp xếp. Tuy nhiên thuật toán sẽ tiếp tục chạy. Phần tử lớn nhất đã ở đúng vị trí nên ta sẽ loại ra. Tiếp tục xét dãy a = [1, 2, 4]

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 4 | 5 |

Tương tự như thế. Ta thu được dãy đã sắp xếp. => Tốn nhiều thời gian, không thể dừng giữa chừng dù vào trường hợp tốt.

1. Selection Sort (Sắp xếp lựa chọn)

Selection Sort là thuật toán đơn giản dựa trên việc so sánh từng phần tử, dãy được chia làm hai phần đã sắp xếp và chưa được sắp xếp. Ban đầu phần được sắp xếp là trống và phần chưa được sắp xếp là toàn bộ danh sách ban đầu.

Phần tử nhỏ nhất được lựa chọn từ mảng chưa sắp xếp và tráo đổi với mảng bên trái nhất của mảng chưa sắp xếp. Phần tử đó trở thành phần tử của mảng đã sắp xếp. Tiến trình này lặp lại cho đến khi toàn bộ phần tử của mảng chưa sắp xếp chuyển sang mảng đã sắp xếp.

Giải thuật này cũng không phù hợp với tập dữ liệu lớn khi ta phải quét toàn bộ mảng để tìm phần tử nhỏ nhất.

Ví dụ: Cho dãy a[ 5, 8, 13, 6, 7, 2]

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 5 | 8 | 13 | 6 | 7 | 2 |

Xét dãy chưa sắp xếp là dãy ban đầu, từ phần tử thứ 0 -> phần tử thứ n-1, so sánh từng cặp phần tử ta tìm được phần tử nhỏ nhất là phần tử a[5]. Đổi vị trí a[5] và a[0].

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 2 | 8 | 13 | 6 | 7 | 5 |

Xét dãy chưa sắp xếp là dãy từ phần tử 1 -> phần tử thứ n-1, so sánh từng cặp tìm được phần tử nhỏ nhất là a[5]. Đổi chỗ a[5] và a[1].

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 2 | 5 | 13 | 6 | 7 | 8 |

Tiếp tục như vậy đến khi dãy được sắp xếp.

…

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 2 | 5 | 6 | 7 | 8 | 13 |

Dãy đã được sắp xếp.

1. Insertion Sort

Sắp xếp chèn là giải thuật sắp xuất dựa trên ý tưởng chia danh sách ra làm 2, trong đó 1 danh sách con luôn duy trì dưới dạng đã sắp xếp. Sắp xếp chèn là chèn thêm một phần tử vào vị trí thích hợp sao cho đảm bảo rằng danh sách con luôn được sắp xếp theo thứ tự.

Giải thuật này cũng không thích hợp để sử dụng với các tập dữ liệu lớn do việc tìm kiếm tuần tự mất nhiều thời gian.

Dãy con ban đầu sẽ có một phần tử đầu tiên. Ta xét lần lượt các phần tử có chỉ số từ 1 -> n-1

Ví dụ dãy a = [4, 3, 1, 5 ,7 ,8]

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 4 | 3 | 1 | 5 | 7 | 8 |

Do a[0] > a[1]. Chúng ta chèn a[1] = 3 vào trước a[0] :

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 3 | 4 | 1 | 5 | 7 | 8 |

Xét tiếp a[2] = 1. Ta thấy a[2] < a[0] và a[1] nên chèn a[2] = 1 vào trước a[1] và a[0]

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 3 | 4 | 5 | 7 | 8 |

Xét a[3] = 5, do a[3] lớn hơn các giá trị phía trước. Ta chèn a[5] vào cuối dãy => vị trí không đổi nhưng thực tế có hành động ở đây.

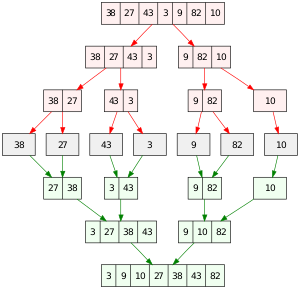
Tiếp tục như vậy ta thu được dãy đã sắp xếp.

1. Merge Sort (sắp xếp gộp)

Merge Sort là giải thuật sắp xếp dựa trên nguyên lý Divide and Conquer (Chia để trị). Đây là giải thuật mạnh hoạt động tốt với các tập dữ liệu lớn, những tập dữ liệu vào train theo External Sorting (Sắp xếp trên bộ nhớ ngoài).

Ý tưởng thuật toán :

Giải thuật sẽ liên tục chia mảng thành 2 nửa sử dụng thuật toán đệ quy cho đến khi chỉ còn lại các mảng chứa 1 phần tử, sau đó sẽ sắp xếp dần chúng lại thành một mảng cuối cùng đã được sắp xếp.



Minh họa về thuật toán Merge Sort - Ảnh : Wikipedia

1. Quick Sort (Sắp xếp nhanh)

Sắp xếp nhanh (Quick Sort) là giải thuật hiệu quả cao cũng dựa trên nguyên tắc Divine and conquer. Quick sort chiaa mảng thành các mảng dữ liệu nhỏ hơn, chia mảng thành hai phần bằng cách so sánh từng phần tử với một phần tử được chọn gọi là phần tử chốt (Pivot), một mảng sẽ bao gồm các phần tử lớn hơn phần tử chốt, một mảng bao gồm các phần tử nhỏ hơn hoặc bằng phần tử chốt. Tiếp tục chia như vậy cho tới khi mỗi mảng con chỉ còn 1 phần tử.   
Giải thuật Quick sort có hiệu quả đối với các tập dữ liệu lớn tương tự như giải thuật merge sort.

Cách hoạt động: Sử dụng biến i và biến jđể chia dãy thành 2 phần. Biến i sẽ chạy từ L đến R và biến j sẽ chạy từ R về L. Nếu phát hiện a[i] >= pivot và a[j] <= pivot thì dừng lại và tráo đổi ví trí của chúng.

VD: cho mảng a = [6, 8, 3, 1, 5, 2, 1, 4]

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 6 | 8 | 3 | 1 | 5 | 2 | 1 | 4 |
| L |  |  | pivot |  |  |  | R |

Đầu tiên ta chọn phần tử chốt. Có thể chọn phần tử chốt là bất cứ phần tử nào nhưng thông thường ta chọn ở đầu hoặc ở chính giữa mảng. Ở đây ta chọn phần tử chính giữa mảng là phẩn tử a[(L+R)/2] = a[3] .

Xét i = L =0 và j = R = 7.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 6 | 8 | 3 | 1 | 5 | 2 | 1 | 4 |

i = 0, a[i] = 6 > pivot; j = 7, a[j] = 4 > pivot => j - 1. I < j => Swap a[i] và a[j]. i +1 = 1, j – 1 = 6.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 8 | 3 | 1 | 5 | 2 | 6 | 4 |

Tương tự, xét a[1] = 8 > pivot. A[6], a[5], a[4] > pivot nên giữ nguyên, j – 3. Do i = 1 < j = 3 => Swap a[1] và a[3], i + 1 = 2, j – 1 = 2. Kết thúc vòng đầu, mảng được chia ra thành 2 mảng, 1 mảng trước pivot chỉ gồm phần tử 1, mảng sau pivot bao gồm a[2:7]. Sắp xếp tiếp a[2:7]

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 1 | 3 | 8 | 5 | 2 | 6 | 4 |

Vòng 2.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 1 | 3 | 8 | 5 | 2 | 6 | 4 |
|  |  | L |  | pivot |  |  | R |

pivot = 4 => a[4] = 5.

Xét tương tự, ta thu được:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 1 | 3 | 4 | 2 | 5 | 6 | 8 |
|  |  | L |  |  | pivot |  | R |

Vòng 3. Với 2 mảng

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 1 | 3 | 4 | 2 | 5 | 6 | 8 |
|  |  | L | pivot | R |  | L | R |

Tương tự, phần tử pivot và phần tử R sẽ đổi chỗ cho nhau. Tiếp tục chạy cho đến khi mỗi mảng chỉ còn 1 phần tử, ta thu được dãy đã sắp xếp.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 1 | 2 | 3 | 4 | 5 | 6 | 8 |