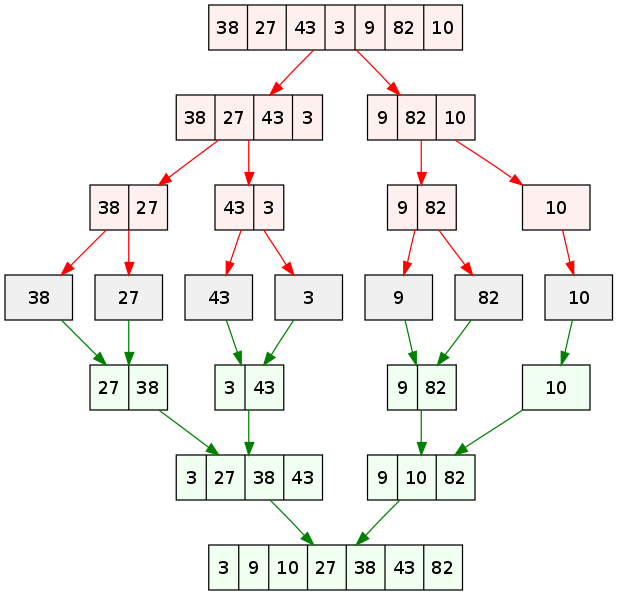
**Câu 3: Ngoài các phương pháp xếp thứ tự đã học, hãy tìm hiểu thêm 1 phương pháp xếp thứ tự khác, giới thiệu sơ và giải thích**

**Merge Sort – Sắp xếp trộn**

**Ý tưởng của thuật toán merge sort**

Giống như Quick sort, Merge sort là một thuật toán chia để trị. Thuật toán này chia mảng cần sắp xếp thành 2 nửa. Tiếp tục lặp lại việc này ở các nửa mảng đã chia. Sau cùng gộp các nửa đó thành mảng đã sắp xếp. Hàm merge() được sử dụng để gộp hai nửa mảng. Hàm merge(arr, l, m, r) là tiến trình quan trọng nhất sẽ gộp hai nửa mảng thành 1 mảng sắp xếp, các nửa mảng là arr[l…m] và arr[m+1…r] sau khi gộp sẽ thành một mảng duy nhất đã sắp xếp.



Hình ảnh trên đây từ wikipedia sẽ hiển thị cho bạn toàn bộ sơ đồ tiến trình của thuật toán merge sort cho mảng {38, 27, 43, 3, 9, 82, 10}. Nếu nhìn kỹ hơn vào sơ đồ này, chúng ta có thể thấy mảng ban đầu được lặp lại hành động chia cho tới khi kích thước các mảng sau chia là 1. Khi kích thước các mảng con là 1, tiến trình gộp sẽ bắt đầu thực hiện gộp lại các mảng này cho tới khi hoàn thành và chỉ còn một mảng đã sắp xếp.

**Đánh giá thuật toán sắp xếp merge sort**

Độ  phức tạp thuật toán

* Trường hợp tốt: O(nlog(n))
* Trung bình: O(nlog(n))
* Trường hợp xấu: O(nlog(n))

Không gian bộ nhớ sử dụng: O(n)

**Distribution Sort - Radix Sort**

**Ý tưởng**

Giả sử mỗi phần tử ai trong dãy a0, a1, …, an-1 là một số nguyên có tối đa m chữ số. Phân loại các phần tử này lần lượt theo các chữ số hàng đơn vị, hàng chục, hàng trăm, …, tương tự việc phân loại thư theo tỉnh thành, quận huyện, phường xã, …

**Ví dụ**

Ta có mảng A cần sắp xếp gồm các phần tử sau:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 964 | 354 | 368 | 128 | 495 | 121 |

Phân lô theo hàng đơn vị:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 12**1** |  |  |  |  |  |  |  |  |  |
| 49**5** |  |  |  |  |  |  |  |  |  |
| 12**8** |  |  |  |  |  |  |  |  |  |
| 36**8** |  |  |  |  |  |  |  |  |  |
| 35**4** |  |  |  | 35**4** |  |  |  | 12**8** |  |
| 96**4** | 12**1** |  |  | 96**4** | 49**5** |  |  | 36**8** |  |
| A | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

Phân lô theo hàng chục:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1**2**8 |  |  |  |  |  |  |  |  |  |
| 3**6**8 |  |  |  |  |  |  |  |  |  |
| 4**9**5 |  |  |  |  |  |  |  |  |  |
| 3**5**4 |  |  |  |  |  |  |  |  |  |
| 9**6**4 |  | 1**2**8 |  |  |  | 3**6**8 |  |  |  |
| 1**2**1 |  | 1**2**1 |  |  | 3**5**4 | 9**6**4 |  |  | 4**9**5 |
| A | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

Phân lô theo hàng trăm:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **4**95 |  |  |  |  |  |  |  |  |  |
| **3**68 |  |  |  |  |  |  |  |  |  |
| **9**64 |  |  |  |  |  |  |  |  |  |
| **3**54 |  |  |  |  |  |  |  |  |  |
| **1**28 | **1**28 |  | **3**68 |  |  |  |  |  |  |
| **1**21 | **1**21 |  | **3**54 | **4**95 |  |  |  |  | **9**64 |
| A | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

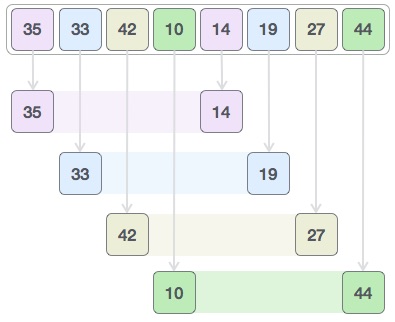
Mảng sau khi đươc sắp xếp:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 121 | 128 | 354 | 368 | 495 | 964 |

**Độ phức tạp**  
*O(k.(n+B))* với k là số pha radix sort, B là hệ cơ số

**Cell Sort**

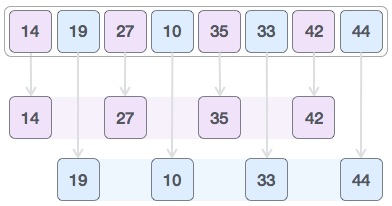
Để dễ tìm hiểu hơn, dưới đây mình cung cấp các hình minh họa cho cách Shell Sort làm việc. Chúng ta sử dụng một mảng gồm các giá trị như dưới đây. Giả sử ban đầu giá trị Khoảng (interval) là 4. Ví dụ, với phần tử 35 thì với khoảng là 4 thì phần tử còn lại sẽ là 14. Do đó ta sẽ có các cặp giá trị {35, 14}, {33, 19}, {42, 27}, và {10, 14}.



So sánh các giá trị này với nhau trong các danh sách con và tráo đổi chúng (nếu cần) trong mảng ban đầu. Sau bước này, mảng mới sẽ trống như sau:

Description: Shell Sort trong cấu trúc dữ liệu và giải thuật

Sau đó, lấy giá trị Khoảng (interval) là 2 và với khoảng cách này sẽ cho hai danh sách con: {14, 27, 35, 42}, {19, 10, 33, 44}.



Tiếp tục so sánh và tráo đổi các giá trị (nếu cần) trong mảng ban đầu. Sau bước này, mảng sẽ trông như sau:

Description: Shell Sort trong cấu trúc dữ liệu và giải thuật

Cuối cùng, chúng ta sắp xếp phần mảng còn lại này với Khoảng (interval) bằng 1. Shell Sort sử dụng giải thuật sắp xếp chèn để sắp xếp mảng. Dưới đây là hình minh họa cho từng bước.



Như trên các hình trên, bạn thấy rằng chúng ta chỉ cần 4 lần tráo đổi để sắp xếp phần mảng còn lại nà

***Độ phức tạp***

o   Yếu tố quyết định chính của thuật toán chính là cách chọn khoảng cách h trong từng bước sắp xếp và số bước sắp xếp k. Nhưng phải thỏa 2 điều kiện sau: hi > hi + 1 và hk= 1.

o   Các phần tử h không được là bội số của nhau nhằm tránh hiện tượng mỗi bước sắp thứ tự phải tổ hợp 2 nhóm mà bước trước chúng không hề có ảnh hưởng lẫn nhau. Điều mong muốn là ảnh hưởng giữa các nhóm khác nhau càng nhiều càng tốt.

o   Việc đánh giá giải thuật Shell sort hiện nay rất phức tạp, thậm chí 1 số chưa được chứng minh. Nhưng có 1 điều chắc chắn là hiệu quả của thuật toán phụ thuộc vào dãy các độ dài được chọn. Trong trường hợp chọn dãy độ dài theo công thức hi= (hi – 1- 1)/2 và hk= 1, k = log2- 1 thì giải thuật có độ phức tạp tương đương n1,2 << n2.