Distribution Sort - Radix Sort

[**DATA STRUCTURE & ALGORITHM**](https://www.stdio.vn/articles/index/13/data-structure-algorithm)

[NGUYỄN THỊ TRÚC LINH](https://www.stdio.vn/users/index/545/nguyen-thi-truc-linh)Sắp xếp là quá trình biến đổi danh sách các đối tượng thành một danh sách thoả mãn một thứ tự nào đó. Thuật toán sắp xếp đóng vai trò quan trọng và được ứng dụng nhiều vào trong lập trình. Trong số đó, Radix Sort là thuật toán hay, độ phức tạp thấp và sử dụng nhiều trong thực tiễn.

Giới thiệu

Sắp xếp là quá trình biến đổi danh sách các đối tượng thành một danh sách thoả mãn một thứ tự nào đó. Thuật toán sắp xếp đóng vai trò quan trọng và được ứng dụng nhiều vào trong lập trình. Trong số đó, Radix Sort là thuật toán hay, độ phức tạp thấp và được sử dụng nhiều trong thực tiễn.

Tiền đề bài viết

Thuật toán sắp xếp Radix Sort là một kiến thức hay tôi học được từ anh [La Kiến Vinh](https://www.stdio.vn/users/index/1/la-kien-vinh). Tôi muốn chia sẻ kiến thức về thuật toán Radix Sort nhằm giúp các bạn có cái nhìn trực quan hơn và ứng dụng hiệu quả vào trong công việc.

Đối tượng hướng đến

Bài viết hướng đến các bạn có kiến thức cơ bản lập trình và muốn tìm hiểu về các giải thuật sắp xếp.

Khái niệm

Radix sort là một thuật toán sắp xếp tiếp cận theo một hướng hoàn toàn khác. Nếu như trong các thuật toán sắp xếp khác, cơ sở để sắp xếp luôn là việc so sánh giá trị của 2 phần tử thì Radix sort lại dựa theo nguyên tắc phân loại thư của bưu điện (Postman’s sort). Radix sort không hề quan tâm đến việc so sánh giá trị của 2 phần tử mà bản thân việc phân loại và thứ tự phân loại sẽ tạo ra thứ tự cho các phần tử.

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 |

Hiện thực

1. void RadixSort(int \*a, int n)
2. {
3. int i, b[MAX], m = a[0], exp = 1;
4. for (i = 0; i < n; i++)
5. {
6. if (a[i] > m)
7. m = a[i];
8. }
9. while (m / exp > 0)
10. {
11. int bucket[10] ={ 0 };
12. for (i = 0; i < n; i++)
13. bucket[a[i] / exp % 10]++;
14. for (i = 1; i < 10; i++)
15. bucket[i] += bucket[i - 1];
16. for (i = n - 1; i >= 0; i--)
17. b[--bucket[a[i] / exp % 10]] = a[i];
18. for (i = 0; i < n; i++)
19. a[i] = b[i];
20. exp \*= 10;
21. }
22. }