

CHƯƠNG 4: COUNTING SORT

GVGD: Ths NGUYỄN CHÍ THANH

Email: thanh.nc@ou.edu.com

NỘI DUNG

1. Ý tưởng

2. Thuật toán

3. Ví dụ

4. Cài đặt

5. Độ phức tạp

1. PHÁT BIỂU BÀI TOÁN



Cho một tập các số nguyên gồm n phần tử:


$$a_0, a_2, a_3, \dots, a_{n-1}$$


Yêu cầu: hãy thực hiện sắp xếp n phần tử này theo thứ tự tăng dần như sau:

$$a_0, a_2, a_3, \dots, a_{n-1}$$

Với $a_0 \leq a_2 \leq a_3 \leq \dots \leq a_{n-1}$

MÔ HÌNH HÓA BÀI TOÁN

 **Đầu vào:** một danh sách đặc (các số nguyên) gồm có n phần tử $a_0, a_2, a_3, \dots, a_{n-1}$.

 **Đầu ra:** một danh sách đặc (các số nguyên) gồm có n phần tử: $a_0, a_2, a_3, \dots, a_{n-1}$ ($a_0 \leq a_2 \leq a_3 \leq \dots \leq a_{n-1}$)

BIỂU DIỄN BÀI TOÁN TRÊN MÁY TÍNH

```
# define MAX 100
```

```
int a[MAX];
```

```
int n; // n là tổng số phần tử hiện có trong danh sách,  $0 \leq n < \text{MAX}$ 
```

Counting sort – Ý tưởng

Với một mảng danh sách đặc $a[]$ có n phần tử từ $a[0]$ đến $a[n-1]$ như sau:

$a[0], a[1], a[2], a[3], \dots, a[n-1]$

Phần tử:	$a[0]$	$a[1]$	$a[2]$	$a[3]$		$a[n-1]$
Vị trí:	0	1	2	3	$n-1$

Counting sort – Ý tưởng

Thực hiện việc ánh xạ giá trị của các phần tử cần sắp xếp thành chỉ mục, thống kê số lần xuất hiện của phần tử sau đó dựa vào thông tin đã có này để chuyển các phần tử từ tập đầu vào sang tập kết quả với vị trí đã sắp xếp.

Dữ Liệu Vào:

- **A[0 .. n-1]**: tập đầu vào có n phần tử với A[i] là số nguyên có giá trị $0 \leq A[i] \leq K$; $0 \leq i \leq n-1$;
- **B[0 .. n-1]**: tập kết quả có n phần tử.
- **C[0 .. K]**: tập chứa thông tin thống kê và chỉ mục tạm

Counting sort – Thuật toán

❖ **Bước 1.** Khởi tạo $c[0 .. K]$ có giá trị 0 ở mọi $c[i]$;

$0 \leq i \leq K$;

❖ **Bước 2.** Đếm số lần xuất hiện của mỗi phần tử trong a và gán vào chỉ mục có giá trị tương ứng với giá trị đó trong c :

$c[a[i]] = \text{số phần tử có giá trị } a[i];$

Counting sort – Thuật toán

- ❖ **Bước 3.** Thực hiện tính tích lũy giá trị các phần tử trong c sao cho $c[i] = \text{sum}(c[0 .. i])$; Về bản chất quá trình này chính là quá trình tính vị trí cuối mới cho loạt các phần tử giống nhau của a trong b.
- ❖ **Bước 4.** Chuyển các phần tử từ tập a sang b (đi từ cuối tập hợp) dựa vào vị trí đã tính trong c, mỗi lần chuyển xong một phần tử thì giá trị của c tương ứng phải giảm 1.

Counting Sort – Ví dụ

Cho mảng danh sách đặc $a[]$ như sau:

Vị trí	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$a[]$	3	1	0	3	0	4	3	3	1	0	2	3	5	4	1	5

$K = 5, n = 16$

❖ **Bước 1:** Khởi tạo mảng $c[]$

Vị trí	0	1	2	3	4	5
$c[]$	0	0	0	0	0	0

Counting Sort – Ví dụ

❖ **Bước 2:** Đếm số lần xuất hiện của mỗi phần tử trong a và $c[a[i]] =$ số phần tử có giá trị a[i];

Vị trí	0	1	2	3	4	5
0.c[]	0	0	0	1	0	0

Vị trí	0	1	2	3	4	5
1.c[]	0	1	0	1	0	0

Vị trí	0	1	2	3	4	5
2.c[]	1	1	0	1	0	0

Counting Sort – Ví dụ

Vị trí	0	1	2	3	4	5
3.c[]	1	1	0	2	0	0

Vị trí	0	1	2	3	4	5
4.c[]	2	1	0	2	0	0

Vị trí	0	1	2	3	4	5
5.c[]	2	1	0	2	1	0

Counting Sort – Ví dụ

Vị trí	0	1	2	3	4	5
6.c[]	2	1	0	3	1	0

Vị trí	0	1	2	3	4	5
7.c[]	2	1	0	4	1	0

Vị trí	0	1	2	3	4	5
8.c[]	2	2	0	4	1	0

Counting Sort – Ví dụ

Vị trí	0	1	2	3	4	5
9.c[]	3	2	0	4	1	0

Vị trí	0	1	2	3	4	5
10.c[]	3	2	1	4	1	0

Vị trí	0	1	2	3	4	5
11.c[]	3	2	1	5	1	0

Counting Sort – Ví dụ

Vị trí	0	1	2	3	4	5
12.c[]	3	2	1	5	1	1

Vị trí	0	1	2	3	4	5
13.c[]	3	2	1	5	2	1

Vị trí	0	1	2	3	4	5
14.c[]	3	3	1	5	2	1

Vị trí	0	1	2	3	4	5
15.c[]	3	3	1	5	2	2

Counting Sort – Ví dụ

❖ **Bước 3:** $c[i] = \text{sum}(c[0 \dots i])$

Vị trí	0	1	2	3	4	5
c[]	3	6	7	12	14	16

❖ **Bước 4:** Chuyển các phần tử từ tập a sang b dựa vào vị trí đã tính trong c, mỗi lần chuyển xong một phần tử thì giá trị của c tương ứng phải giảm 1.

Vị trí	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
b[]	0	0	0	1	1	1	2	3	3	3	3	3	4	4	5	5

Counting sort – Cài đặt

```
void countingSort(int a[], int b[], int k, int n)
{
    int i, j;
    for(i = 0; i <= k; i++) { c[i] = 0; }
    for(j = 1; j <= n; j++) { c[a[j]]++; }
    for(i = 1; i <= k; i++) { c[i] = c[i] + c[i - 1]; }
    for(j = n; j >= 1; j--)
    { b[c[a[j]]] = a[j]; c[a[j]]--; }
}
```

Counting sort – Độ phức tạp

Độ phức tạp thuật toán:

Trường hợp	Độ phức tạp
Tốt nhất	$O(n)$
Trung bình	$O(n)$
Xấu nhất	$O(n)$

TỔNG KẾT

- **Trình bày ý tưởng thuật toán Counting Sort**
- **Thuật toán Counting Sort**
- **Cài đặt thuật toán Counting Sort**
- **Đánh giá độ phức tạp**





TÀI LIỆU THAM KHẢO

1. **Thomas H.Cormen, Charles E.Leiserson, Ronald L. Rivest, Clifford Stein**, (Chapter 2, 3) *Introduction to Algorithms*, Third Edition, 2009.
2. **Adam Drozdek**, (Chapter 9) *Data Structures and Algorithms in C++*, Fourth Edition, CENGAGE Learning, 2013.
3. **Lê Xuân Trường**, (Chapter 2) *Cấu trúc dữ liệu*, NXB Trường Đại học Mở TP-HCM, 2016.

BÀI TẬP CHƯƠNG 4

Bài 1: Thực hiện mô tả từng bước quá trình sắp xếp thứ tự dãy số nguyên bằng thuật toán Counting Sort

Bài 2: Quản lý danh sách đặc 100 phần tử kiểu số nguyên (int)

-  2.1. Khai báo cấu trúc danh sách.
-  2.2. Viết thủ tục nhập danh sách.
-  2.3. Viết thủ tục xuất danh sách
-  2.4. Viết thủ tục sắp xếp danh sách theo thứ tự tăng dần bằng thuật toán Counting Sort

Bài tập thêm

Bài 3: Thực hiện đánh giá độ phức tạp của thuật toán CountingSort

Bài 4: Quản lý danh sách liên kết đơn (các phần tử kiểu số nguyên)

-  Viết thủ tục sắp xếp danh sách theo thứ tự tăng dần bằng thuật toán Counting Sort