Bài 9.9: Thuật toán radix sort

- ✓ Tổng quan về thuật toán
- ✓ Thuật toán sắp xếp radix sort
- ✓ Ví dụ minh họa
- ✓ Bài tập thực hành

Tổng quan

- Thuật toán radix sort là thuật toán sắp xếp không dựa trên việc so sánh các phần tử mảng.
- Nó tránh việc so sánh bằng cách tạo và phân phối các phần tử vào các nhóm theo cơ số của chúng.
- Thuật toán radix sort dựa trên thuật toán counting sort.
- Radix sort có thể được triển khai theo hướng tiếp cận MSD(chữ số quan trọng nhất) hoặc LSD(chữ số ít quan trọng nhất).
- Hướng tiếp cận LSD thường tuân thủ quy tắc: các khóa ngắn đứng trước khóa dài. Các khóa có cùng độ dài sẽ được sắp xếp theo thứ tự từ điển.

Tổng quan

- Có thể áp dụng cho các tập dữ liệu có thể sắp xếp theo thứ tự từ điển, các số nguyên, các từ hoặc email…
- Độ phức tạp của thuật toán này là O(nw) với n là lượng các khóa trong mảng và w là số chữ số cấu thành nên khóa.

Thuật toán sắp xếp đếm

➢ Mã giả của thuật toán counting sort:

```
// thuật toán sắp xếp đếm
// input: mảng đầu vào
// n: kích thước mảng input
// exp: giá trị 10^i đang xét
function countingSort(input[], n, int exp):
    k = 9 // số chữ số tối đa của 1 giá trị kiểu int(~2.1 tỉ)
    count[] = new int[k + 1] // tạo mảng count có k + 1 phần tử
    output = new int[n] // tạo mảng output có n phần tử
    for(i từ 0 tới n - 1): // đếm số lần xuất hiện của phần tử
        j = (input[i] / exp) % 10; // tại vị trí i trong mảng input
        count[j] += 1; // tăng biến đếm tại vị trí j lên 1 đơn vị
    for(i từ 1 tới k): // tìm tổng tiền tố cho phần tử tại vị trí i
        count[i] += count[i - 1]
    for(i từ n - 1 tới 0): // đưa các phần tử vào đúng vị trí của nó
        j = (input[i] / exp) % 10 // lấy phần tử tại vị trí i mảng input
        count[j] = count[j] - 1 // giảm biến đếm của nó đi 1
        output[count[j]] = input[i] // gán phần tử vào đúng vị trí
    for(i từ 0 tới n - 1): // chép các phần tử vào mảng gốc
        input[i] = output[i]
    delete[] count, output // thu hồi bộ nhớ đã cấp phát động
```

Thuật toán sắp xếp đếm

➢ Mã thật của thuật toán counting sort:

```
// thuật toán sắp xếp đếm
void countingSort(int* input, int size, int exp) {
    int k = 9; // số chữ số tối đa của giá trị lớn nhất kiểu int
    int* output = new int[size]; // cấp phát mảng output
    int* count = new int[k + 1](); // cấp phát, khởi tạo giá trị mặc định cho mảng
    for (int i = 0; i < size; i++) // tìm số lần xuất hiện của từng phần tử
    {
        int j = (input[i] / exp) % 10;
        count[j]++;
    }
    for (int i = 1; i <= k; i++) // cộng dồn tổng tiền tố
    {
        count[i] += count[i - 1];
    }
    for (int i = size - 1; i >= 0; i--) // đưa các phần tử vào đúng vị trí của nó
    {
        int j = (input[i] / exp) % 10;
        count[j] = count[j] - 1;
        output[count[j]] = input[i];
    }
    for (int i = 0; i < size; i++) // chép kết quả sang mảng gốc input
    {
        input[i] = output[i];
    }
    delete[] count; // thu hồi bộ nhớ cấp phát động
    delete[] output; // thu hồi bộ nhớ cấp phát động
}
```

Thuật toán sắp xếp đếm

- Mã giả của thuật toán radix sort:

```
// thuật toán radix sort
// arr: mảng đầu vào
// n: kích thước mảng input
function radixSort(arr[], int n):
    max = findMax(arr)
    for(i = 1; max / i > 0; i = i * 10):
        countingSort(arr, i)

// tìm giá trị lớn nhất trong mảng
// arr: mảng đầu vào
// n: kích thước mảng input
function findMax(arr[], n):
    max = arr[0]
    for(i từ 1 đến n - 1):
        if(arr[i] > max):
            max = arr[i]
    return max
```

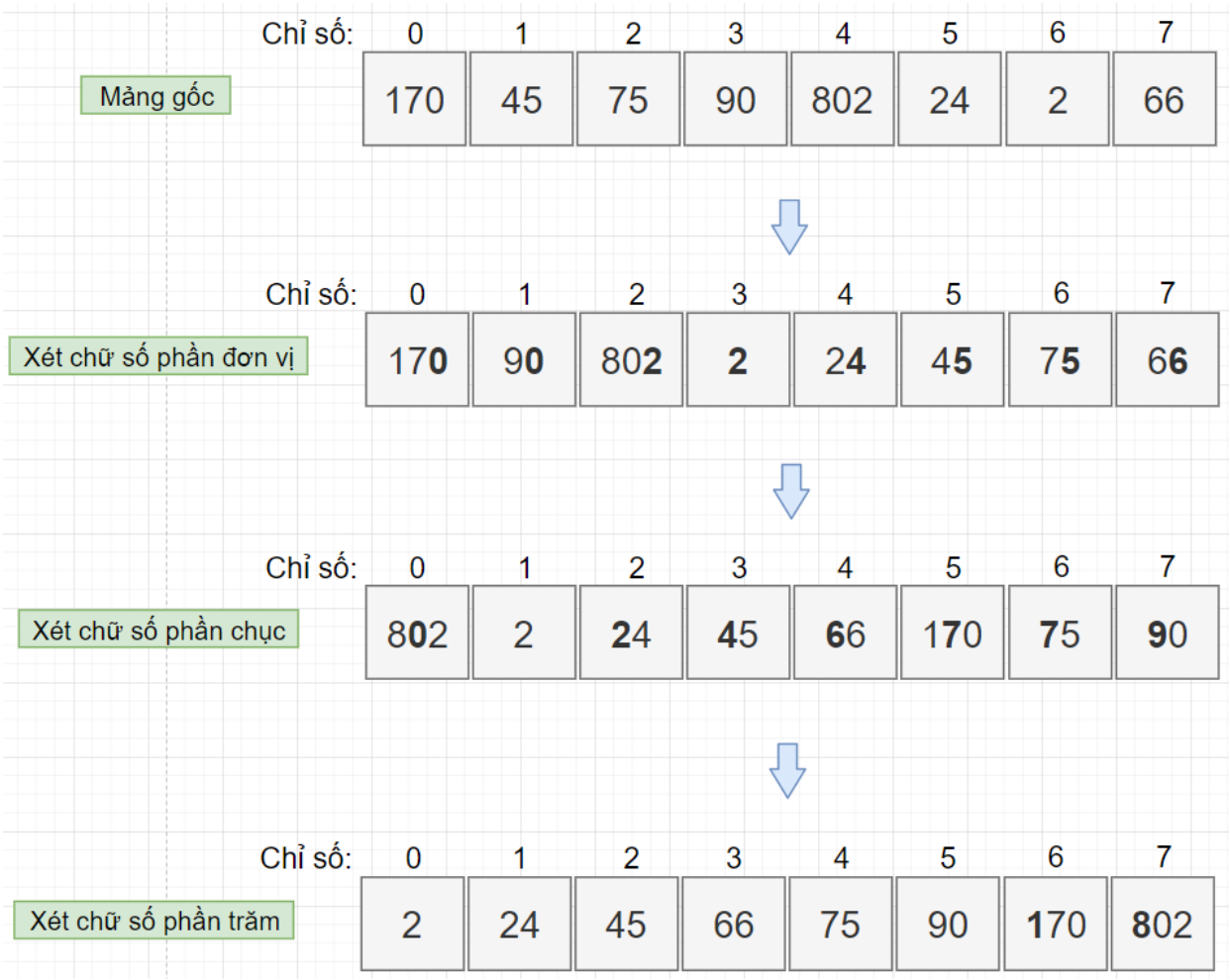
Thuật toán sắp xếp đếm

➢ Mã thật của thuật toán radix sort:

```
// hàm tìm giá trị lớn nhất trong mảng
int findMax(int* arr, int n) {
    int max = arr[0];
    for (int i = 1; i < n; i++)
    {
        if (arr[i] > max) {
            max = arr[i];
        }
    }
    return max;
}
// thuật toán sắp xếp radix sort
void radixSort(int* arr, int n) {
    int max = findMax(arr, n);
    for (int i = 1; max / i > 0; i *= 10) {
        countingSort(arr, n, i);
    }
}
```

Minh họa

➢ Ví dụ sắp xếp bằng radix sort:



Nội dung tiếp theo

Một số hàm sắp xếp trong thư viện C++