

Bài 8.7: Bucket sort

- ✓ Khái niệm và đặc điểm
- ✓ Mã giả và triển khai
- ✓ Ví dụ minh họa
- ✓ Bài tập thực hành

Khái niệm và đặc điểm

- Bucket sort hay còn gọi là bin sort là một thuật toán sắp xếp dựa trên sự phân bố các phần tử của mảng vào các vùng chứa khác nhau.
- Sau đó mỗi vùng chứa sẽ tiếp tục sắp xếp bằng các thuật toán khác hoặc tiếp tục đệ quy với bucket sort.
- Bucket sort là một thuật toán sắp xếp phân tán. Là anh em của thuật toán radix sort.
- Thuật toán này có thể được triển khai bằng cách so sánh các cặp phần tử.
- Độ phức tạp của thuật toán phụ thuộc vào thuật toán dùng để sắp xếp mỗi vùng chứa, vào số lượng vùng chứa được sử dụng và phụ thuộc vào việc dữ liệu đầu vào có được phân phối đồng đều không.

Khái niệm và đặc điểm

- Độ phức tạp thuật toán:
 - Tệ nhất $O(n^2)$
 - Tốt nhất: $O(n + k)$
 - Trung bình: $O(n)$
- Sử dụng thuật toán sắp xếp này khi dữ liệu đầu vào phân bố đồng đều và có các giá trị số thực.

Các bước thực hiện

Thuật toán bucket sort hoạt động như sau:

- B1: khởi tạo một mảng của các vùng chứa rỗng.
- B2: phân phối các phần tử của mảng gốc vào trong vùng chứa phù hợp.
- B3: sắp xếp các vùng chứa không rỗng.
- B4: tập hợp các phần tử trong các vùng chứa theo thứ tự và đưa các phần tử của chúng trở lại mảng ban đầu.

Ví dụ



Mã giả

➤ Mã giả của thuật toán:

```
// thuật toán sắp xếp bucket sort
// arr: mảng đầu vào
// n: số phần tử của mảng
// k: số phân vùng được chỉ định
bucketSort(arr, n, k) {
    bucket <- khởi tạo mảng của k + 1 danh sách rỗng
    M <- giá trị lớn nhất trong mảng
    for(i = 0; i < n; i++) {
        bucket[floor(k*arr[i]/M)] = arr[i];
    }
    for(i = 0; i <= k; i++) {
        sắp xếp danh sách thứ i
    }
    index = 0;
    for(i = 0; i <= k; i++) {
        lấy từng phần tử của danh sách thứ i gán vào mảng arr[index]
        index++;
    }
}
```


Triển khai

```
// thuật toán bucket sort
template<class T> void bucketSort(T* arr, int n, int k) {
    if (k < 0) { // nếu số vùng chứa không hợp lệ, kết thúc chương trình
        cout << "Giá trị vùng chứa không hợp lệ: " << k << endl;
        return;
    }
    // tạo danh sách vùng chứa
    vector<T>* buckets = new vector<T>[k + 1];
    // phân bổ các phần tử vào vùng chứa
    T max = findMax(arr, n);
    for (int i = 0; i < n; i++)
    {
        int bucketIndex = (int)(arr[i] * k / max);
        buckets[bucketIndex].push_back(arr[i]);
    }
    // sắp xếp các vùng chứa không rỗng
    for (int i = 0; i <= k; i++)
    {
        if (buckets[i].size() > 0) {
            sort(buckets[i].begin(), buckets[i].end());
        }
    }
    // gộp các phần tử lại mảng gốc
    int index = 0;
    for (int i = 0; i <= k; i++)
    {
        int bSize = (int)buckets[i].size();
        for (int j = 0; j < bSize; j++)
        {
            arr[index++] = buckets[i][j];
        }
    }
}
```



Nội dung tiếp theo

Tìm hiểu lớp set trong thư viện C++