Bài 7.7: 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


11 9 21 8 17 19 13 1 24 12
Scatter
1
9 8
11 13 12
17 19
21 24
0-5
5-10
10-15
15-20
20-25
Sort
1
8 9
11 12 13
17 19
21 24
0-5
5-10
10-15
15-20
20-25
Gather
1 8 9 11 12 13 17 19 21 24

Mã giả

➢ Mã giả của thuật toán:

```
// thuật toán sắp xếp bucket sort
bucketSort(arr, k) {
    bucket <- khởi tạo mảng của k + 1 danh sách rỗng
    M <- giá trị lớn nhất trong mảng
    for(i = 0; i < arr.length; 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

```java
public static void bucketSort(double[] arr, int k) {
    if (k < 0) { // nếu n nhỏ hơn 0, kết thúc
        throw new RuntimeException("Số phân vùng không hợp lệ: " + k);
    }
    @SuppressWarnings("unchecked")
    ArrayList<Double>[] buckets = new ArrayList[k + 1];
    for (int i = 0; i <= k; i++) { // khởi tạo k danh sách rỗng
        buckets[i] = new ArrayList<>();
    }
    double max = findMax(arr); // tìm giá trị lớn nhất trong mảng
    // phân phối các phần tử vào các danh sách khác nhau
    for (int i = 0; i < arr.length; i++) {
        int bucketIndex = (int) (arr[i] * k / max);
        buckets[bucketIndex].add(arr[i]);
    }
    for (int i = 0; i <= k; i++) { // sắp xếp các danh sách
        buckets[i].sort( c: null); // sắp xếp tăng dần
    }
    // gộp các phần tử lại mảng gốc
    int index = 0; // khởi tạo vị trí index
    for (int i = 0; i <= k; i++) {
        for (int j = 0; j < buckets[i].size(); j++) {
            arr[index] = buckets[i].get(j);
            index++;
        }
    }
}
```

Nội dung tiếp theo

Tìm hiểu lớp Hashtable trong thư viện Java