Bài 8.7: Thuật toán sắp xếp nhanh

- ✓ Các đặc điểm
- ✓ Thuật toán sắp xếp nhanh
- ✓ Ví dụ minh họa
- ✓ Bài tập thực hành

Các đặc điểm

- Thuật toán sắp xếp nhanh có độ phức tạp O(nlogn).
- Đây là thuật toán thường được áp dụng trong các chức năng sắp xếp.
- Tuy vậy, hiệu quả của một thuật toán còn phụ thuộc vào vấn đề cần giải quyết và bộ dữ liệu đầu vào mà nó vận hành.
- Quicksort là một thuật toán thuộc dạng chia để trị. Nó chọn một phần tử làm mốc và chia tập hợp cần sắp xếp thành 3 phần:
  - Phần 1: chứa các phần tử nhỏ hơn phần tử chọn làm mốc.
  - Phần 2: gồm phần tử chọn làm mốc.
  - Phần 3: gồm các phần tử lớn hơn phần tử chọn làm mốc.
- Có nhiều cách chọn phần tử làm mốc: ở đầu, cuối, giữa hoặc ngẫu nhiên trong tập hợp.
- Thuật toán minh họa trong bài học này chọn phần tử cuối tập hợp làm mốc.

Thuật toán sắp xếp nhanh

- Mã giả của thuật toán quick sort:

```
// thuật toán quicksort:
void quicksort(arr[], leftIndex, rightIndex) {
    if (leftIndex < rightIndex) { // nếu biên trái < biên phải
        p = partition(arr, leftIndex, rightIndex)
        quicksort(arr, leftIndex, p - 1)
        quicksort(arr, p + 1, rightIndex)
    }
}

// thuật toán phân mảnh và sắp xếp các phần tử
int partition(arr[], left, right) {
    pivot = arr[right] // giá trị phần tử được chọn làm mốc
    i = left // khởi tạo biến chạy i bắt đầu từ biên trái
    for (j from left to right) { // chạy j từ biên trái đến biên phải
        if (arr[j] < pivot) {
            tmp = arr[i];
            arr[i] = arr[j];
            arr[j] = tmp;
            i = i + 1
        }
    }
    swap(arr[i], arr[right]) // đổi chỗ arr[i] và arr[right]
    return i // trả về vị trí phần tử chọn làm mốc kế tiếp
}
```

Thuật toán sắp xếp nhanh

- Mã thật của thuật toán quick sort:

```
// thuật toán sắp xếp nhanh trên mảng
public static <T extends Comparable<T>> void quicksort(T[] arr, int leftIndex, int rightIndex) {
    if (leftIndex < rightIndex) {
        int p = partition(arr, leftIndex, rightIndex);
        quicksort(arr, leftIndex, rightIndex: p - 1);
        quicksort(arr, leftIndex: p + 1, rightIndex);
    }
}

// thuật toán phân mảnh và sắp xếp các phần tử
public static <T extends Comparable<T>> int partition(T[] arr, int left, int right) {
    T pivot = arr[right];
    int i = left;
    for (int j = left; j <= right; j++) {
        if (arr[j].compareTo(pivot) < 0) {
            T tmp = arr[i];
            arr[i] = arr[j];
            arr[j] = tmp;
            i = i + 1;
        }
    }
    T tmp = arr[i];
    arr[i] = arr[right];
    arr[right] = tmp;
    return i;
}
```

Nội dung tiếp theo

Thuật toán sắp xếp counting sort