Bài 9.6: Thuật toán sắp xếp trộn

- ✓ Các đặc điểm của thuật toán
- ✓ Mô tả thuật toán
- ✓ Ví dụ minh họa
- ✓ Bài tập thực hành

Các đặc điểm

- Giống như thuật toán sắp xếp nhanh, sắp xếp trộn là một thuật toán chia để trị.
- Thuật toán sắp xếp trộn có độ phức tạp O(nlogn).
- Thuật toán chia mảng cần sắp xếp ra thành hai nửa qua chỉ số phần tử giữa.
- Sau đó tiếp tục đệ quy việc chia tách trên hai nửa vừa tách được đến khi mảng chỉ còn 1 phần tử thì dừng chia tách.
- Lúc này thuật toán tiến hành trộn hai mảng đã sắp xếp lại đến khi nhận được mảng đã sắp xếp hoàn chỉnh.

Mô tả thuật toán sắp xếp

- Mã giả và triển khai của thuật toán merge sort:

```
// thuật toán sắp xếp trộn đệ quy
// arr: mảng đầu vào
// first: chỉ số phần tử đầu
// last: chỉ số phần tử cuối
function mergeSort(arr[], first, last):
    if(first < last): // nếu chỉ số first nhỏ hơn last
        middle = (first + last) / 2 // tìm chỉ số phần tử giữa
        mergeSort(arr, first, middle) // đệ quy nửa trái mảng
        mergeSort(arr, middle + 1, last) // đệ quy nửa phải mảng
        merge(arr, first, middle, last) // trộng mảng kết quả
```

```
// thuật toán sắp xếp trộn
template<class T> void mergeSort(T* arr, int first, int last) {
    if (first < last) {
        int middle = (first + last) / 2;
        mergeSort(arr, first, middle);
        mergeSort(arr, middle + 1, last);
        merge(arr, first, middle, last);
    }
}
```

Mô tả thuật toán trộn

➢ Mã giả của thuật toán merge:

```
// thuật toán trộn hai mảng đã sắp xếp
// arr: mảng đầu vào
// first: chỉ số phần tử đầu
// last: chỉ số phần tử cuối
// middle: chỉ số phần tử giữa
function merge(arr[], first, middle, last):
    size1 = middle - first + 1 // kích thước mảng con trái
    size2 = last - middle // kích thước mảng con phải
    leftArr[] = new T[size1] // tạo mảng con trái
    rightArr[] = new T[size2] // tạo mảng con phải
    for(i từ 0 tới size1 - 1): // chép dữ liệu vào mảng con trái
        leftArr[i] = arr[l + i]
    for(j từ 0 tới size2 - 1): // chép dữ liệu vào mảng con phải
        rightArr[j] = arr[middle + j];
    i = 0, j = 0, k = first // các biến chạy
    while(i < size1 && j < size2): // tiến hành trộn
        if(leftArr[i] <= rightArr[j]):
            arr[k++] = leftArr[i++];
        else:
            arr[k++] = rightArr[j++]
    while(i < size1): // chép các phần tử còn lại của mảng con trái
        arr[k++] = leftArr[i++]
    while(j < size2): // chép các phần tử còn lại của mảng con phải
        arr[k++] = rightArr[j++]
```

Mã thật thuật toán trộn

➢ Mã thật của thuật toán merge:

```
template<class T> void merge(T* arr, int first, int middle, int last) {
    int size1 = middle - first + 1;
    int size2 = last - middle;
    T* leftArr = new T[size1];
    T* rightArr = new T[size2];
    for (int i = 0; i < size1; i++)
    {
        leftArr[i] = arr[first + i];
    }
    for (int j = 0; j < size2; j++)
    {
        rightArr[j] = arr[middle + j + 1];
    }
    int i = 0, j = 0, k = first;
    while (i < size1 && j < size2) {
        if (leftArr[i] <= rightArr[j]) {
            arr[k++] = leftArr[i++];
        }
        else {
            arr[k++] = rightArr[j++];
        }
    }
    while (i < size1) {
        arr[k++] = leftArr[i++];
    }
    while (j < size2) {
        arr[k++] = rightArr[j++];
    }
    delete[] leftArr;
    delete[] rightArr;
}
```

Nội dung tiếp theo

Thuật toán sắp xếp nhanh