

## Bài 6.6: Thuật toán sắp xếp heap sort

---

- ✓ Tổng quan thuật toán
- ✓ Mã giả và triển khai
- ✓ Ví dụ minh họa
- ✓ Bài tập thực hành

# Tổng quan thuật toán

- Thuật toán sắp xếp vun đống(heap sort) là một thuật toán sắp xếp dựa trên việc so sánh các cặp phần tử và sử dụng tính chất của cấu trúc dữ liệu heap(đống).
- Heap được sử dụng trong thuật toán là max heap nhị phân.
- Heap có thể được triển khai bằng mảng hoặc cây nhị phân, ta áp dụng heap triển khai bằng mảng trong thuật toán heap sort.
- Độ phức tạp của thuật toán này là  $O(n\log n)$ .

# Tổng quan thuật toán

Các bước thực hiện của thuật toán sắp xếp tăng dần:

- B1: tạo max heap từ mảng đầu vào.
- B2: trao đổi phần tử đầu và cuối heap; giảm kích thước của heap đi 1; sàng xuống node đầu heap.
- B3: lặp lại bước 2 đến khi heap chỉ còn 1 phần tử.

Ở đây ta lưu ý, trong mảng  $n$  phần tử, các phần tử từ vị trí  $n/2$  trở về sau là các node lá của cây nhị phân.

Thủ tục sàng(vun đống) chỉ có thể thực hiện được nếu các node con của nó đã là đống. Do đó khi tiến hành vun mảng thành đống ta vun từ dưới lên từ vị trí  $[n/2] - 1$ .

# Mã giả của thuật toán

➤ Mã giả của thuật toán heap sort:

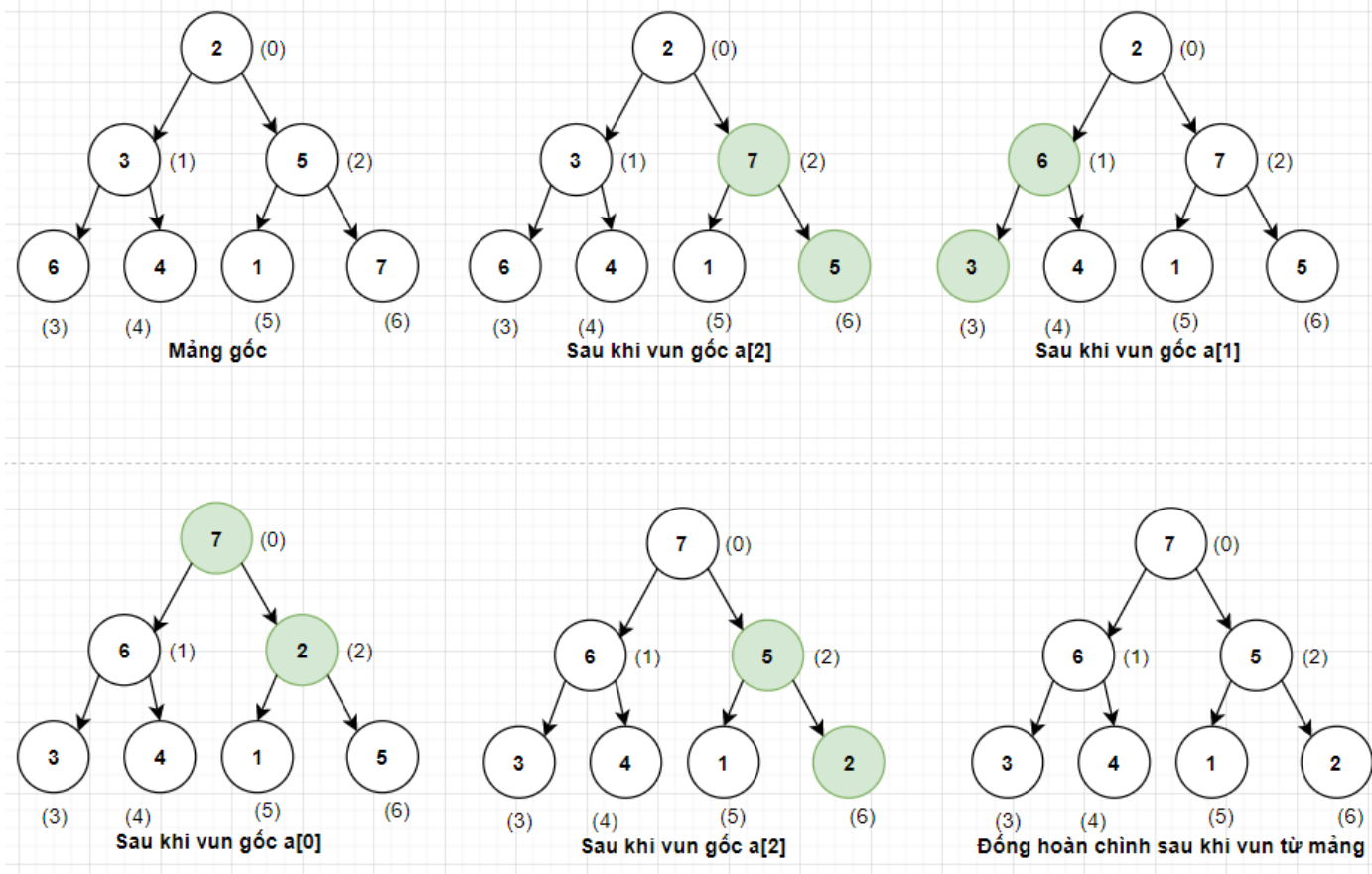
```
// thuật toán sắp xếp vun đống tăng dần
void heapSort(arr[]) {
    n = arr.length;
    // tạo max heap từ mảng đầu vào
    for(i = n/2 - 1; i >= 0; i--) {
        siftDown(arr, n, i); // sàng xuống từ vị trí i
    }
    // lần lượt đưa từng phần tử ở đầu heap về đúng vị trí
    for (i = n-1; i > 0; i--)
    { // trao đổi phần tử đầu và cuối heap
        tmp = arr[0];
        arr[0] = arr[i];
        arr[i] = tmp;
        // gọi phương thức sàng xuống để tái cân bằng heap mới
        siftDown(arr, i, 0);
    }
}
```

# Mã thật của thuật toán

➤ Sau đây là mã thật của thuật toán heap sort:

```
// thuật toán sắp xếp vun đống tăng dần
public static <E extends Comparable<E>> void heapSort(E[] arr) {
    var n :int = arr.length;
    // tạo max heap từ mảng đầu vào
    for (int i = n / 2 - 1; i >= 0; i--) {
        siftDown(arr, n, i); // sàng xuống từ vị trí i
    }
    // Lần lượt đưa từng phần tử ở đầu heap về đúng vị trí
    for (int i = n - 1; i > 0; i--) { // trao đổi phần tử đầu và cuối heap
        E tmp = arr[0];
        arr[0] = arr[i];
        arr[i] = tmp;
        // gọi phương thức sàng xuống để tái cân bằng heap mới
        siftDown(arr, i, index: 0);
    }
}
```

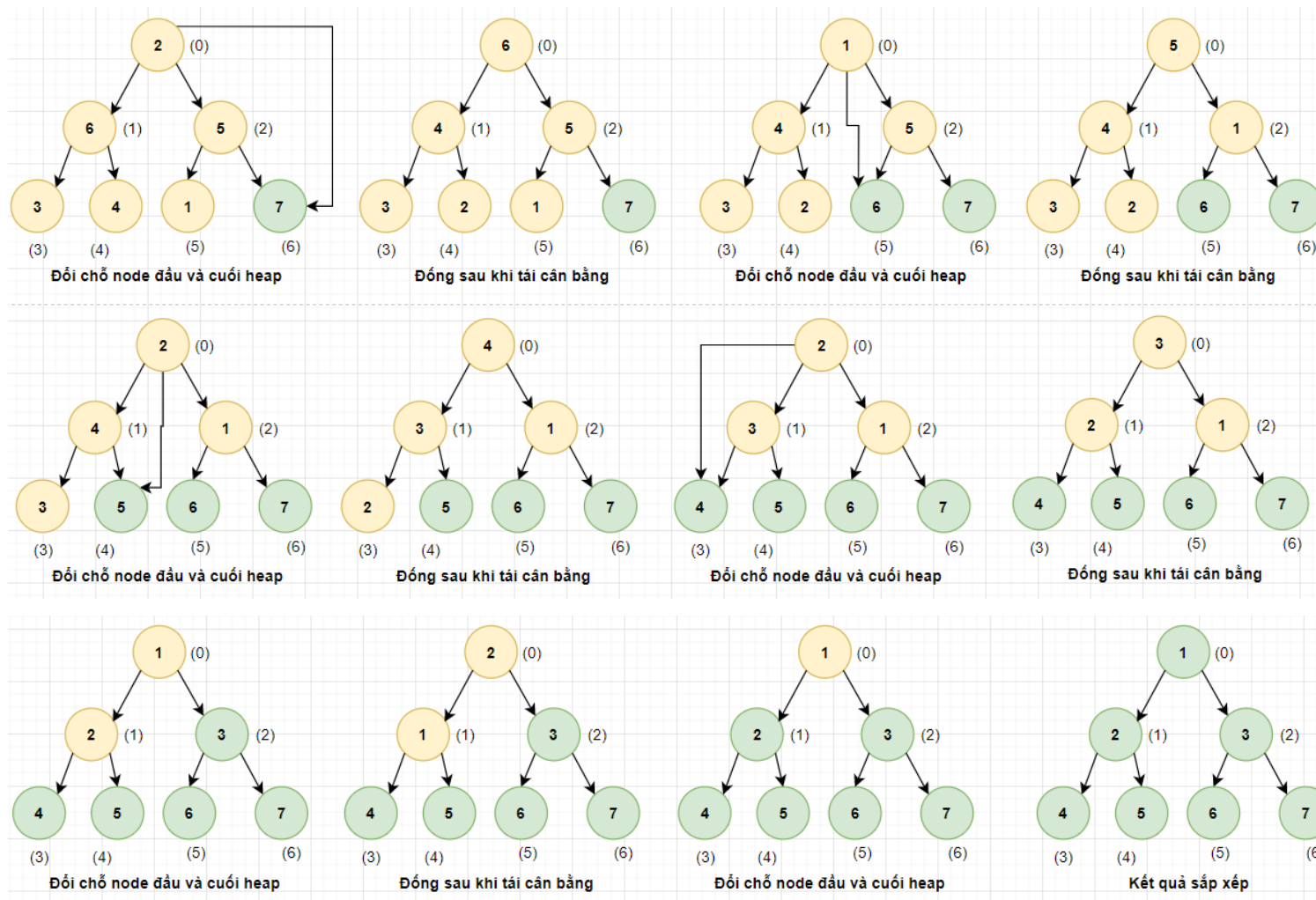
- Cho mảng a gồm 7 phần tử [2, 3, 5, 6, 4, 1, 7]. Các phần tử từ vị trí 3 đến 6 là node lá.
- Quá trình vun gốc tạo đống từ mảng đầu vào:



Ví dụ minh họa

➤ Sau đây là quá trình trao đổi phần tử:

# Quá trình trao đổi đổi phần tử



# Nội dung tiếp theo

**Fibonacci heap**