Bài 7.3: Xóa node khỏi heap

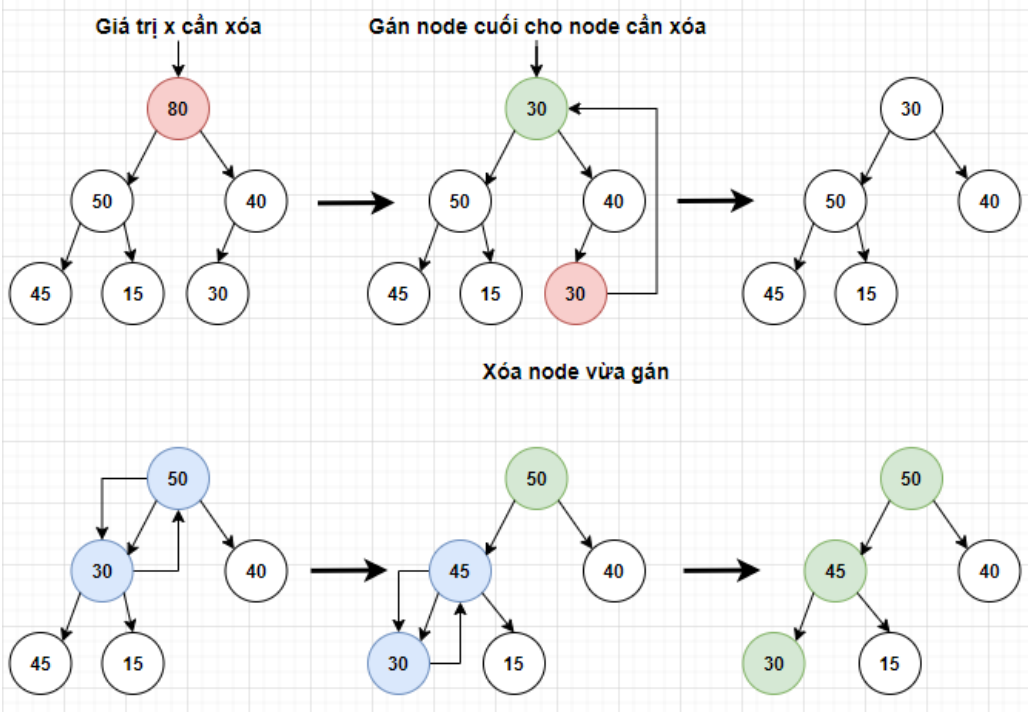
- ✓ Các bước thực hiện
- ✓ Mã giả và triển khai
- ✓ Ví dụ minh họa
- ✓ Bài tập thực hành

Các bước thực hiện

- Yêu cầu: xóa node có giá trị x khỏi max heap.
- Các bước thực hiện:
  - B1: xác định vị trí node cần xóa gọi là index.
  - B2: nếu index hợp lệ, gán phần tử cuối cho phần tử tại vị trí index.
  - B3: xóa bỏ phần tử cuối vừa sử dụng ở bước 2.
  - B4: sàng xuống để tái cân bằng lại các phần tử trong heap tại nhánh có node cha ở vị trí index.

Các bước thực hiện

➢ Hình ảnh minh họa:



Xóa node khỏi heap

- Sau đây là mã giả thao tác xóa node khỏi heap.

```
// thao tác xóa giá trị value trong heap
// value: giá trị cần xóa
function remove(value):
    index = findNodeIndex(value) // lấy vị trí của value trong heap
    if(index >= 0):     // nếu tìm thấy
        data[index] = data[currentSize - 1] // sử dụng phần tử cuối
        currentSize--   // giảm số phần tử trong heap
        siftDown(index) // sàng xuống
        return true     // xóa thành công
    else:               // ngược lại
        return false    // xóa thất bại
```

Xóa node khỏi heap

- Sau đây là mã thật thao tác xóa node khỏi heap.

```
// hàm xóa giá trị value trong heap
bool remove(T value) {
    int index = findNodeIndex(value);
    if (index >= 0) {
        data[index] = data[currentSize - 1];
        currentSize--;
        siftDown(index);
        return true;
    }
    else {
        return false;
    }
}
```

Sàng xuống

- Sau đây là mã giả thao tác sàng xuống.

```
// thao tác sàng xuống
// index: vị trí phần tử cần xem xét
function siftDown(index):
    largest = index // giả sử vị trí phần tử lớn nhất là index
    left = 2 * index + 1 // tìm vị trí phần tử con bên trái
    right = 2 * index + 2 // tìm vị trí phần tử con bên phải
    if(left < currentSize và data[left] > data[largest]):
        largest = left // cập nhật left
    if(right < curentSize và data[right] > data[largest]):
        largest = right // cập nhật right
    if(largest != index): // nếu hai vị trí là khác nhau
        swap(data[index], data[largest]) // tráo đổi phần tử
        siftDown(largest) // tiếp tục sàng xuống
```

Sàng xuống

- Sau đây là mã thật thao tác sàng xuống.

```
// hàm sàng xuống
void siftDown(int index) {
    int largest = index;
    int left = 2 * index + 1;
    int right = 2 * index + 2;
    if (left < currentSize && data[left] > data[largest]) {
        largest = left;
    }
    if (right < currentSize && data[right] > data[largest]) {
        largest = right;
    }
    if (largest != index) {
        swap(data[index], data[largest]);
        siftDown(largest);
    }
}
```

Nội dung tiếp theo

Cập nhật dữ liệu cho một node trong heap