Bài 4.5: Hàng đợi ưu tiên

- ✓ Định nghĩa và đặc trưng
- ✓ Ứng dụng của hàng đợi ưu tiên
- ✓ Các hành động đặc trưng
- ✓ Ví dụ minh họa
- ✓ Bài tập thực hành

Định nghĩa và đặc trưng

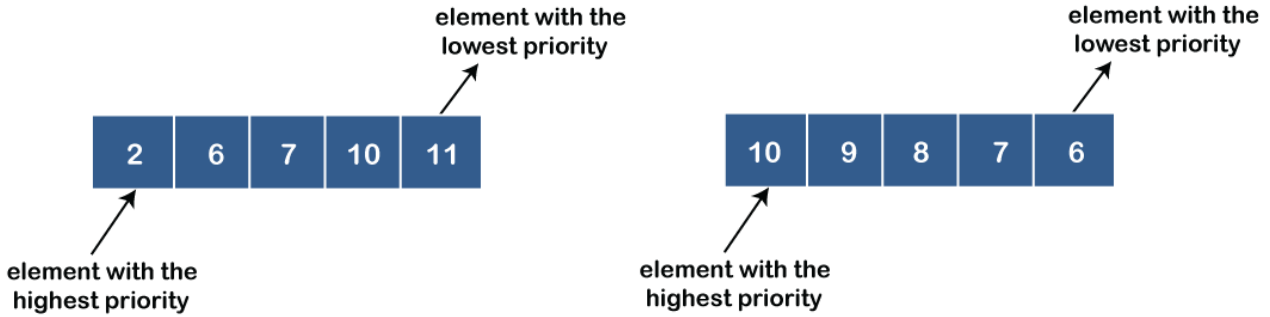
- Hàng đợi ưu tiên(priority queue) là biến thể của hàng đợi trong đó mỗi phần tử sẽ có mức ưu tiên khác nhau. Phần tử có mức ưu tiên cao nhất sẽ được đưa lên đầu queue và được lấy ra đầu tiên. Phần tử có mức ưu tiên nhỏ hơn sẽ được lấy ra sau.
- Hàng đợi ưu tiên chỉ hỗ trợ các phần tử có thể so sánh với nhau, tức là các phần tử trong hàng đợi sẽ được sắp xếp theo trật tự tăng dần hoặc giảm dần.
- Đặc trưng 1: mỗi phần tử sẽ có một mức ưu tiên gắn liền với nó.
- Đặc trưng 2: một phần tử có mức ưu tiên cao hơn sẽ bị xóa khỏi hàng đợi trước các phần tử có mức ưu tiên thấp hơn.
- Đặc trưng 3: nếu 2 phần tử có cùng mức ưu tiên chúng sẽ được sắp xếp theo thứ tự FIFO.

Ứng dụng

- Thuật toán tìm đường đi ngắn nhất: Dijkstra.
- Thuật toán Prim.
- Nén dữ liệu.
- Thuật toán tìm kiếm A* trong trí tuệ nhân tạo.
- Heap sort.
- Trong hệ điều hành: cân bằng tải, xử lý ngắt

Phân loại

- Có 2 loại hàng đợi ưu tiên:
- Loại 1: hàng đợi ưu tiên có các phần tử được sắp xếp theo thứ tự tăng dần giá trị các phần tử theo tiêu chí nào đó. Trong hàng đợi kiểu này, phần tử có giá trị nhỏ hơn sẽ có mức ưu tiên cao hơn.



- Loại 2: hàng đợi ưu tiên có các phần tử được sắp xếp theo thứ tự giảm dần giá trị các phần tử theo tiêu chí nào đó. Trong hàng đợi kiểu này, các phần tử có giá trị lớn hơn sẽ có mức ưu tiên cao hơn.

Các hành động

- add(e): thêm phần tử mới vào queue theo mức ưu tiên từ cao đến thấp.
- remove(): xóa và trả về phần tử có mức ưu tiên cao nhất ở đầu queue.
- peek(): lấy phần tử đầu queue nhưng không xóa.
- isEmpty(): kiểm tra queue có rỗng hay không.
- isFull(): kiểm tra queue có đầy không(áp dụng với triển khai bằng mảng).
- size(): trả về số lượng phần tử hiện có trong queue.

Triển khai hàng đợi ưu tiên

- Sử dụng mảng.
- Sử dụng danh sách liên kết.
- Sử dụng heap.
- Sử dụng cây nhị phân tìm kiếm.
- Trong bài này ta sẽ triển khai hàng đợi ưu tiên sử dụng danh sách liên kết.

Tạo lớp PriorityQueue với phương thức add()

```java
public class PriorityQueue<E> {
    private int currentSize;
    private LinkedList<E> data;

    PriorityQueue() {
        currentSize = 0;
        data = new LinkedList<>();
    }

    public void add(E e, int priority) {
        currentSize++;
        data.add(e, priority);
    }
```

Lấy phần tử đầu queue và xóa phần tử đầu queue

```
public E peek() {
    return data.front();
}

public E remove() {
    if (!isEmpty()) {
        currentSize--;
    }
    return data.remove();
}
```

Kiểm tra queue rỗng và lấy kích thước queue

```
public boolean isEmpty() {
    return currentSize == 0;
}

public int size() {
    return currentSize;
}
```

Danh sách liên kết và lớp Node<E>

```
public class LinkedList<E> {
    private Node<E> head;

    static class Node<E> {
        private E data;
        private int priority;
        private Node<E> next;

        public Node(E data, int priority) {
            this.data = data;
            this.priority = priority;
            this.next = null;
        }
    }
```

Phương thức add()

```
public void add(E e, int priority) {
    Node<E> p = new Node<>(e, priority);
    if (isEmpty()) {
        head = p;
    } else if (p.priority > head.priority) {
        p.next = head;
        head = p;
    } else {
        var r : LinkedList<...>.Node<...> = head;
        for (var q : LinkedList<...>.Node<...> = head.next; q != null; q = q.next) {
            if (q.priority < p.priority) {
                break;
            }
            r = q;
        }
        p.next = r.next;
        r.next = p;
    }
}
```

Phương thức remove(), front() và isEmpty()

```
public E remove() {
    if (!isEmpty()) {
        E tmp = head.data;
        head = head.next;
        return tmp;
    }
    return null;
}

public E front() {
    return isEmpty() ? null : head.data;
}

public boolean isEmpty() {
    return head == null;
}
```

Nội dung tiếp theo

Hàng đợi ưu tiên trong thư viện Java