UNIVERSITY OF CALGARY

CPSC 319
Data Structures, Algorithms, and Their Applications

Winter 2024

Introduction to PriorityQueue

- PriorityQueue is a class in Java that implements a priority queue, which orders elements based on their natural ordering or by a Comparator provided at queue construction time.

- It does not permit null elements and does not guarantee any specific ordering for equal elements.

Key Features

- Enqueue and dequeue elements efficiently according to their priority.
- Offers constant time for the retrieval of the highest priority element.
- Automatically maintains the elements' order based on their priority.
- Allows for customization of ordering using Comparator.

Declaration and Initialization

- To use PriorityQueue, import the java.util.PriorityQueue package.

- Declare and instantiate a PriorityQueue object specifying the type of elements it will contain.

Adding Elements

- Use add() or offer() method to add elements to the PriorityQueue.

- Elements will be inserted according to their natural order or the specified Comparator.

Retrieving Elements

- The poll() method removes and returns the highest priority element from the queue.

- Use peek() to retrieve the highest priority element without removing it.

Custom Comparators

- PriorityQueue allows the use of custom comparators to order elements based on criteria other than their natural ordering.

- Pass a Comparator object during PriorityQueue initialization to define custom ordering.

Iterating Through Elements

- PriorityQueue does not offer direct iteration methods like forEach() or traditional loops.

- To iterate, convert the PriorityQueue to an array or list and then iterate through it.

Complexity Analysis

- Enqueue (add/offer): O(log n)
- Dequeue (poll): O(log n)
- Retrieval (peek): O(1)
- Removing specified elements: O(n)

Use Cases

- Used in algorithms like Dijkstra's shortest path algorithm and Prim's minimum spanning tree algorithm.
- Task scheduling where tasks are executed based on priority.
- Simulating event-driven systems where events are processed based on their priority.

Limitations

- Not thread-safe. For concurrent access, consider using PriorityBlockingQueue.
- Elements must be comparable or a Comparator must be provided.
- Removal of specific elements has linear time complexity.