Priority Queue

This is a queue designed to handle the queuing of objects in a priority order based on a given priority value.

Requirements

- Able to queue an element with a given priority
    - Method signature:

```
void Enqueue(T value, int priority)
```

- Able to dequeue the element with the minimum priority
    - Method signature:

```
T DequeueMin()
```

- Able to dequeue the element with the maximum priority
    - Method signature:

```
T DequeueMax()
```

- Support multiple entries with the same priority
- Able to peek at the front and back of the queue

Future Requirements

- Can dequeue multiple entries with the same priority value
    - If multiple entries can be made at the same priority level it makes sense to be able to retrieve all of them at once

Assumptions

- Items inserted at the same priority level will be handled First In First Out

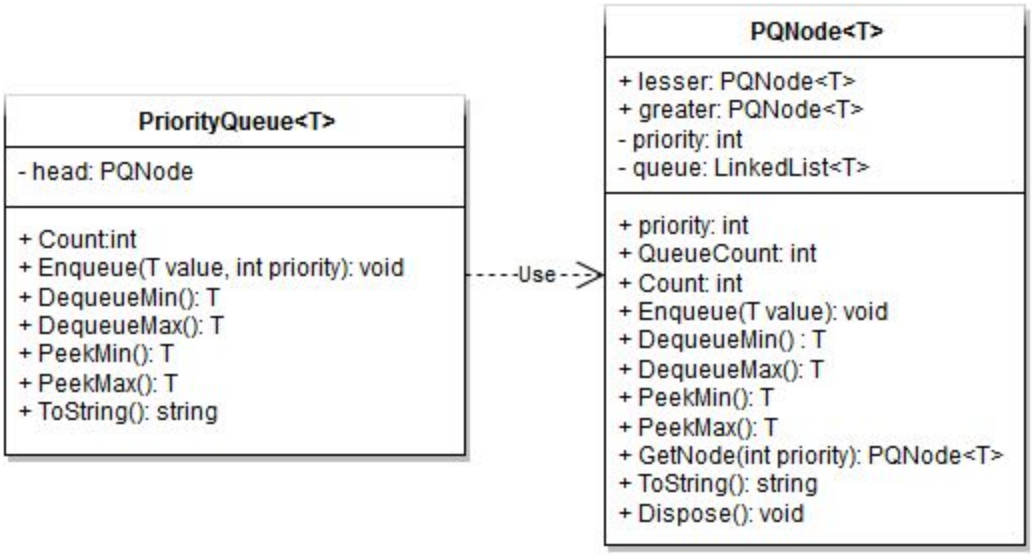
Overview

There are a few challenges to overcome with creating this Priority Queue:

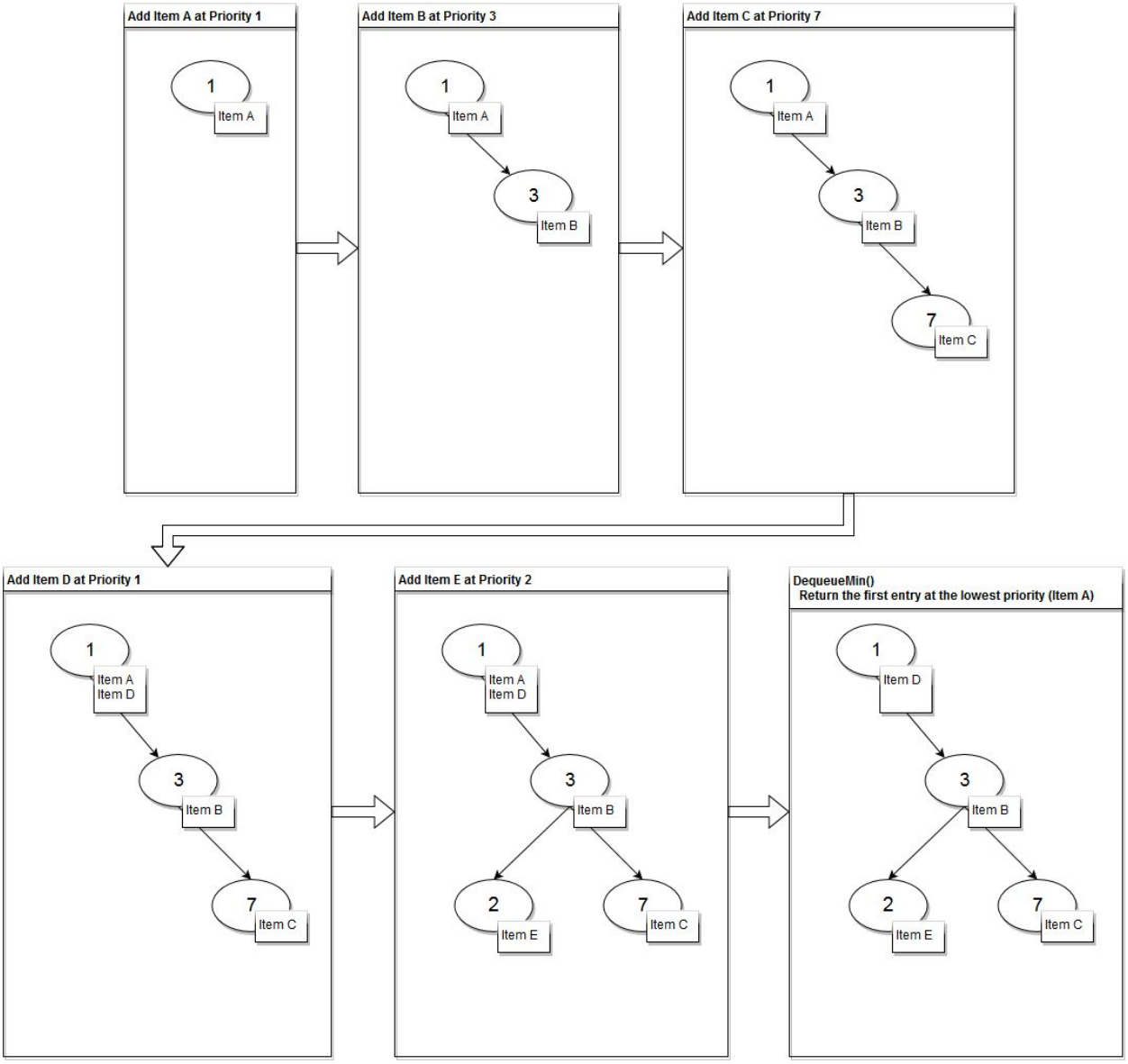
- Adding and removing elements needs to be handled quickly.
- We cannot guarantee that priority values will be relative to one another or added in sequence.
    - We may add a priority 1 item followed by a priority 3 item with a priority 2 item being added last or not at all.
    - We may add a priority 1 item followed by a priority 12 item with nothing else falling between the two.
- We need to be able to remove items from the front or back of the queue.

The challenges outlined can be handled by implementing a Binary Search Tree with nodes that contain a LinkedList allowing for retrieval from the front or back of a list of items with a similar priority.

Class Diagram



Binary Search Tree Example



Notes

- This implementation does not support removing items from the queue outside of the Dequeue methods.
    - Adding support for this would require a re-evaluation of the implementation as building it into the current solution would result in a very slow method
- What should happen if an item of the same value is added to the queue at a different priority? Should both co-exist or should the original be modified?
    - Like removing items, a solution for this within the current implementation would be very slow