Java PriorityQueue Class

Anders Jacobsen, Dima Karaush

May 6, 2021

Abstract

Java's PriorityQueue class have a bottleneck when you need to update values in the queue. Depending on the use and implementation, Java's PriorityQueue can add serious performance issues when accessing the data using the poll() method. When updating a value in the queue, performance can be improved by more than 50% compared to Java's PriorityQueue implementation. Implementing your own version might remove this bottleneck from your software.

1 Introduction

Why did we choose this subject This article will discuss Java's "Build-in" class `PriorityQueue`. The interest for this topic has grown from an implmentation of a weighted graph in Java, which we found had a possible shortcomming for our use-case. The shortcomming was that to update a value in the queue, we had to use a linear search (loop) through the queue to update a value within. That is fine on a small scale, but what if need to draw a graph of all the cities in the world? We believe that this has space for optimization and that is why this article explores this subject.

We are going to use a previously developed `Timer` class.

How will we work with the subject

2 Scope

What will be in this article What will not be included in this article

3 Problem

3.1 Background

3.2 Problem Statement

The questions we will try to solve Problemfomulering

4 Analysis

4.1 Benchmark Method

4.2 Benchmark of Java's priorityQueue

To understand the example that we are using we have to take a look at how we use our queue.

```
1     PriorityQueue<Node> pQueue = new PriorityQueue<>(
      QUEUE_LENGTH, nodeComp);
2
3     // Filling the queue with Nodes
4     for (int i = 0; i < QUEUE_LENGTH; i++) {
5         pQueue.add(new Node(i, i + 1, i));
6     }
```

Listing 1: Timer implementation

We create a simple priorityQueue that takes nodes as an object. Then we add some nodes that fill up the queue, the nodes do not need to have any special values. But we do add a unique values so that we are able to distinguish the nodes.

When were looking at updating a node in Java's priority queue. There is no method to retrieve the wanted node from the list. Therefore we use a Iterator to list through every node in our list until we find a match.

```
1     Timer timer = new Timer();
2     long time = 0L;
3     int count = 0;
4     Node n = null;
5
6     timer.start();
7     while (it.hasNext()) {
8         n = it.next();
9         if (count == QUEUE_LENGTH - 1) {
10            n.verticeTo = 80085;
11            time = timer.step();
12            break;
13        }
14        count++;
15    }
```

Listing 2: Timer implementation

Notice that we are listing through the queue in a linear way. This is also the

4.3 Update Method

4.4 Benchmark of updateable PriorityQueue

4.5 Comparisson of PriorityQueues

5 Conclusion

Answer our questions and possibly introduction