Lab 2 Writeup – Dean Maynard

Benchmark Results

All benchmarks were run with WARMUP_OPS = 15,000, MEASURE_OPS = 60,000, 7 trials, and a fixed seed of 42. Checksums matched across every trial which means the results are correct.

Stacks – W1: ArrayListStack 13 ns/op, DLinkedListStack 31 ns/op Stacks – W2: ArrayListStack 50 ns/op, DLinkedListStack 53 ns/op Queues – W1: ArrayListQueue 14 ns/op, DLinkedListQueue 31 ns/op Queues – W2: ArrayListQueue 14 ns/op, DLinkedListQueue 16 ns/op Priority Queues – W1: SortedArrayListPQ 22,560 ns/op, SortedDLinkedListPQ 56,268 ns/op, BinaryHeapPQ 54 ns/op Priority Queues – W2: SortedArrayListPQ 13,183 ns/op, SortedDLinkedListPQ 23,288 ns/op, BinaryHeapPQ 50 ns/op Priority Queues – W3: SortedArrayListPQ 22,511 ns/op, SortedDLinkedListPQ 56,668 ns/op, BinaryHeapPQ 45 ns/op

Stacks

ArrayListStack was faster in both workloads. I think this is because the ArrayList keeps everything in one block of memory so pushing and popping from the end is quick. With the DLinkedListStack, every push creates a new node object and those can end up all over the place in memory. So when you need to access them it takes a bit longer. The difference was smaller in W2 since the mixed operations don't give either one a clear advantage like W1 does.

Queues

Same story as stacks basically. The ArrayList version uses a circular buffer so there's no shifting and both operations stay O(1). The linked list version is also O(1) but it still has to create a new node every time something gets enqueued, and that adds up over 60,000 operations. The gap in W2 was really small though, only 14 vs 16 ns/op.

Priority Queues

This was the most interesting part. The heap was way faster than both sorted implementations, like 400 times faster in W1. The sorted structures have to find the right position to insert every time which is O(n), and SortedArrayListPQ also calls sort() after every insert which makes it even slower. The heap only needs O(log n) per insert so at large N it just runs away from the others. For W3 with skewed priorities I thought the sorted ones might do better since most items would land near the front of the list, but it

didn't really help much. The heap still won easily. I think the only time a sorted structure would be competitive is if the list stayed really small the whole time.

========== STACK BENCHMARKS ==========

[W1] ArrayListStack - fill then drain
  trial 0 checksum=1799970000
  trial 1 checksum=1799970000
  trial 2 checksum=1799970000
  trial 3 checksum=1799970000
  trial 4 checksum=1799970000
  trial 5 checksum=1799970000
  trial 6 checksum=1799970000
 >>> MEDIAN ns/op: 13

[W1] DLinkedListStack - fill then drain
  trial 0 checksum=1799970000
  trial 1 checksum=1799970000
  trial 2 checksum=1799970000
  trial 3 checksum=1799970000
  trial 4 checksum=1799970000
  trial 5 checksum=1799970000
  trial 6 checksum=1799970000
 >>> MEDIAN ns/op: 31

[W2] ArrayListStack - mixed steady state
  trial 0 checksum=725592181
  trial 1 checksum=725592181
  trial 2 checksum=725592181
  trial 3 checksum=725592181
  trial 4 checksum=725592181
  trial 5 checksum=725592181
  trial 6 checksum=725592181
 >>> MEDIAN ns/op: 50

[W2] DLinkedListStack - mixed steady state
  trial 0 checksum=725592181
  trial 1 checksum=725592181
  trial 2 checksum=725592181
  trial 3 checksum=725592181

```
    trial 4  checksum=725592181
    trial 5  checksum=725592181
    trial 6  checksum=725592181
  >>> MEDIAN ns/op: 53

========== QUEUE BENCHMARKS ==========

[W1] ArrayListQueue - fill then drain
    trial 0  checksum=1799970000
    trial 1  checksum=1799970000
    trial 2  checksum=1799970000
    trial 3  checksum=1799970000
    trial 4  checksum=1799970000
    trial 5  checksum=1799970000
    trial 6  checksum=1799970000
  >>> MEDIAN ns/op: 14

[W1] DLinkedListQueue - fill then drain
    trial 0  checksum=1799970000
    trial 1  checksum=1799970000
    trial 2  checksum=1799970000
    trial 3  checksum=1799970000
    trial 4  checksum=1799970000
    trial 5  checksum=1799970000
    trial 6  checksum=1799970000
  >>> MEDIAN ns/op: 31

[W2] ArrayListQueue - mixed steady state
    trial 0  checksum=178087585
    trial 1  checksum=178087585
    trial 2  checksum=178087585
    trial 3  checksum=178087585
    trial 4  checksum=178087585
    trial 5  checksum=178087585
    trial 6  checksum=178087585
  >>> MEDIAN ns/op: 14

[W2] DLinkedListQueue - mixed steady state
    trial 0  checksum=178087585
    trial 1  checksum=178087585
```

```
    trial 2  checksum=178087585
    trial 3  checksum=178087585
    trial 4  checksum=178087585
    trial 5  checksum=178087585
    trial 6  checksum=178087585
  >>> MEDIAN ns/op: 16

========== PRIORITY QUEUE BENCHMARKS ==========

[W1] SortedArrayListPQ - fill then drain
    trial 0  checksum=1799970000
    trial 1  checksum=1799970000
    trial 2  checksum=1799970000
    trial 3  checksum=1799970000
    trial 4  checksum=1799970000
    trial 5  checksum=1799970000
    trial 6  checksum=1799970000
  >>> MEDIAN ns/op: 22560

[W1] SortedDLinkedListPQ - fill then drain
    trial 0  checksum=1799970000
    trial 1  checksum=1799970000
    trial 2  checksum=1799970000
    trial 3  checksum=1799970000
    trial 4  checksum=1799970000
    trial 5  checksum=1799970000
    trial 6  checksum=1799970000
  >>> MEDIAN ns/op: 56268

[W1] BinaryHeapPQ - fill then drain
    trial 0  checksum=1799970000
    trial 1  checksum=1799970000
    trial 2  checksum=1799970000
    trial 3  checksum=1799970000
    trial 4  checksum=1799970000
    trial 5  checksum=1799970000
    trial 6  checksum=1799970000
  >>> MEDIAN ns/op: 54

[W2] SortedArrayListPQ - mixed steady state
```

```
    trial 0  checksum=589865148
    trial 1  checksum=589865148
    trial 2  checksum=589865148
    trial 3  checksum=589865148
    trial 4  checksum=589865148
    trial 5  checksum=589865148
    trial 6  checksum=589865148
  >>> MEDIAN ns/op: 13183

[W2] SortedDLinkedListPQ - mixed steady state
    trial 0  checksum=589865148
    trial 1  checksum=589865148
    trial 2  checksum=589865148
    trial 3  checksum=589865148
    trial 4  checksum=589865148
    trial 5  checksum=589865148
    trial 6  checksum=589865148
  >>> MEDIAN ns/op: 23288

[W2] BinaryHeapPQ - mixed steady state
    trial 0  checksum=593976295
    trial 1  checksum=593976295
    trial 2  checksum=593976295
    trial 3  checksum=593976295
    trial 4  checksum=593976295
    trial 5  checksum=593976295
    trial 6  checksum=593976295
  >>> MEDIAN ns/op: 50

[W3] SortedArrayListPQ - skewed priorities
    trial 0  checksum=1799970000
    trial 1  checksum=1799970000
    trial 2  checksum=1799970000
    trial 3  checksum=1799970000
    trial 4  checksum=1799970000
    trial 5  checksum=1799970000
    trial 6  checksum=1799970000
  >>> MEDIAN ns/op: 22511

[W3] SortedDLinkedListPQ - skewed priorities
```

```
    trial 0  checksum=1799970000
    trial 1  checksum=1799970000
    trial 2  checksum=1799970000
    trial 3  checksum=1799970000
    trial 4  checksum=1799970000
    trial 5  checksum=1799970000
    trial 6  checksum=1799970000
  >>> MEDIAN ns/op: 56668

[W3] BinaryHeapPQ - skewed priorities
    trial 0  checksum=1799970000
    trial 1  checksum=1799970000
    trial 2  checksum=1799970000
    trial 3  checksum=1799970000
    trial 4  checksum=1799970000
    trial 5  checksum=1799970000
    trial 6  checksum=1799970000
  >>> MEDIAN ns/op: 45

========== DONE ==========
```