```
PS C:\Users\Lawson_PC\Desktop\Lawson Fuller>  & 'C:\Program
Files\Java\jre1.8.0_481\bin\java.exe'
'-agentlib:jdwp=transport=dt_socket,server=n,suspend=y,address=localhost:62445' '-cp'
'C:\Users\Lawson_PC\AppData\Roaming\Code\User\workspaceStorage\1eaac59dca09e75c771
2a38164b7caf5\redhat.java\jdt_ws\Lawson Fuller_d18ca5ad\bin' 'benchmark.BenchmarkDriver'
Java ADT Benchmark (nanoTime).
Warmup ops: 15000, Measure ops: 60000, Trials: 7

Sanity checks: OK

== Stack: ArrayListStack ==
  Workload1 bulk push+pop              median:      10.88 ns/op   checksum: 449985000
  Workload2 mixed steady-state          median:      25.12 ns/op   checksum: -1055495428

== Stack: DLinkedListStack ==
  Workload1 bulk push+pop              median:      10.53 ns/op   checksum: 449985000
  Workload2 mixed steady-state          median:      22.57 ns/op   checksum: -1055495428

== Queue: ArrayListQueue ==
  Workload1 bulk enq+deq               median:      13.44 ns/op   checksum: 449985000
  Workload2 mixed steady-state          median:      26.01 ns/op   checksum: 8738648310

== Queue: DLinkedListQueue ==
  Workload1 bulk enq+deq               median:      10.80 ns/op   checksum: 449985000
  Workload2 mixed steady-state          median:      25.52 ns/op   checksum: 8738648310

== PriorityQueue: SortedArrayListPQ ==
  Workload1 bulk enq+deq (uniform priorities)  median:    2850.90 ns/op   checksum: 449985000
  Workload2 mixed steady-state (uniform priorities)  median:    2967.74 ns/op   checksum:
-101944034770
  Workload3 skewed priorities (bulk)      median:    3066.90 ns/op   checksum: 449985000

== PriorityQueue: SortedDLinkedListPQ ==
  Workload1 bulk enq+deq (uniform priorities)  median:   20595.21 ns/op   checksum:
449985000
  Workload2 mixed steady-state (uniform priorities)  median:   19495.89 ns/op   checksum:
-101944034770
  Workload3 skewed priorities (bulk)      median:   17926.00 ns/op   checksum: 449985000

== PriorityQueue: BinaryHeapPQ ==
  Workload1 bulk enq+deq (uniform priorities)  median:      58.59 ns/op   checksum: 449985000
  Workload2 mixed steady-state (uniform priorities)  median:      65.76 ns/op   checksum:
-106337492798
  Workload3 skewed priorities (bulk)      median:      44.13 ns/op   checksum: 449985000
```

1) Stack

stack implementations are O(1) for push and pop, so performance was very similar. In the bulk test, they were the same (around 10–11 ns/op). In the mixed workload, the `DLinkedListStack` was faster (22.57 vs 25.12 ns/op). The difference is small because both versions do constant time operations. The small performance changes are probably due to how memory is handled and small details rather than big differences.

2) Queue

The `DLinkedListQueue` was faster in both workloads (10.80 vs 13.44 ns/op in bulk and 25.52 vs 26.01 ns/op in mixed). The ArrayList version uses a buffer, which adds a extra work each time. The linked list version updates pointers, which may explain why it performed better. Overall, the difference is small because both implementations are constant time.

3) PriorityQueue

The difference was in the PriorityQueue results. The `BinaryHeapPriorityQueue` was faster (around 44 to 66 ns/op) compared to the sorted ArrayList (around 2800–3000 ns/op) and the sorted linked list (around 17,000–20,000 ns/op).

This is because the heap uses log n operations, whilesorted versions use O(n) insertion. When the num elements gets large (around 30,000), O(n) become much slower than O(log n). That is why the heap wins.

Between the two sorted versions, the ArrayList was much faster than the linked list. Even though both are O(n), the ArrayList stores elements next to each other in memory, which is faster for the computer to access. The linked list has to follow pointers from node to node, which slows it down.

Overall, the results match what we would expect from Big-O complexity, especially for large data sizes.