

In class Merge Sort was discussed for an array. The original array and the subarrays that are created are accessed only sequentially. That means Merge Sort can be easily implemented for a collection that only allows sequential access (such as a linked list or even a queue). **The random access capability provided by an array is not required to implement Merge Sort.**

In this assignment, we will extend the IntQueue class to create a class that has additional methods, so that a client can implement Merge Sort and selection sort for the collection.

Examine the existing code and fill in the missing pieces.

The client should be modified so that it shows that sorting works for a small random queue. Then it should create random queues of large sizes (say 10,000 to 100,000 in steps of 10, 000). The client should produce a table of queue sizes and corresponding sorting times in milliseconds. The client should also get the times for sorting random queues using selection sort for the same sizes and include that for comparison.

Quite a bit of startup code is provided which was discussed in class.

These are the results I got from my program. Different runs do tend to give different results.

```
Output - Asgn04_KEY (run) - Editor
Output - Asgn04_KEY (run)  >>

run:
CPS 151 Assignment 4 by KEY

Demonstrate Selection sort on a Linked Queue

Before sorting
[538, 790, 602, 927, 975, 372, 451, 227, 705, 220, 100, 520, 64, 29, 401, 157, 582, 990, 458, 435]

After sorting
[29, 64, 100, 157, 220, 227, 372, 401, 435, 451, 458, 520, 538, 582, 602, 705, 790, 927, 975, 990]

Demonstrate Merge sort on a Linked Queue

Before sorting
[989, 455, 813, 830, 789, 647, 666, 745, 471, 588, 917, 330, 5, 511, 49, 744, 564, 421, 593, 940]

After sorting
[5, 49, 330, 421, 455, 471, 511, 564, 588, 593, 647, 666, 744, 745, 789, 813, 830, 917, 940, 989]

Comparison of Selection and Merge sort timing

Sorting times in milliseconds
-----
  Size  Merge  Select
  ----  ----
 10000     8    163
 20000    13    917
 30000    20   1614
 40000    31   3170
 50000    40   5415
 60000    52   6974
 70000    50   9550
 80000    62  11588
 90000    97 12991
100000   116 20166

Assignment 4 complete
BUILD SUCCESSFUL (total time: 1 minute 14 seconds)
```