

# University of Washington, Bothell

## CSS 501: Data Structures and Object-Oriented Programming I

### Example Program 4: Mergesorting Linked Lists

#### Purpose

You've probably learned about mergesort in the context of arrays. However, consider for a moment the core part of the mergesort algorithm: the merge operation. This operation scans through two sequences of items linearly; it doesn't require the random access capabilities of an array. So, mergesort is actually a good algorithm for sorting lists. In fact, it's even better for linked lists than arrays, since it can merge two parts of a linked list in place — without requiring extra storage.

#### Mergesort for Linked Lists

Mergesort takes an input list and treats it as though it were a collection of small sorted lists. It makes  $\log N$  passes along the list, and in each pass it combines each adjacent pair of small sorted lists into one larger sorted list. When a pass only needs to do this once, the whole output list is sorted.

**Require:**  $L$  is a singly linked list of length  $N$

**Ensure:** Upon return,  $L$  is sorted from low to high

```
repeat
   $k = 1$ 
  Set pointer  $p$  to point to the head of  $L$ 
  Let  $T$  be an empty temporary list
  Set number of merges to zero
  while  $p \neq \text{NULL}$  do
    Increment number of merges
    Set pointer  $q = p$ 
    Step  $q$  along the list  $k$  items (or until end of list)
    Set  $psize$  to number of items skipped
     $qsize = k$  {Merge a list of length  $psize$ , starting at  $p$ , with a list of at most  $qsize$ , starting at  $q$ }
    while  $psize > 0$  or ( $qsize > 0$  and  $q \neq \text{NULL}$ ) do
      if one list is empty then
        Set pointer  $e$  to the item from the non-empty list
      else
        Set pointer  $e$  to the smaller of the current items in the two lists
      end if
      Remove  $e$  from the list, advancing either  $p$  or  $q$  and decrementing either  $psize$  or  $qsize$ 
      Add  $e$  to the end of list  $T$ 
    end while {We have merged the  $p$  and  $q$  lists}
  end while
   $L = T$ 
   $k = k \times 2$  {We've merged all lists of length  $k$ }
until number of merges equals 1
```

As you can see, this is the same mergesort algorithm we learned in class, except that it is operating on a linked list and it is implemented iteratively (by starting with lists of size 1 and building up). Note that, while the array based mergesort requires  $\Theta(N)$  extra memory, the linked list based one only requires  $\Theta(1)$ , because moving items from one list to another in the merge also moves their storage.

## Statement of Work

Write a program to perform a mergesort on a linked list, using a linked list class of your own design with a `mergeSort()` method. To test your algorithm, your program should create a linked list object and initialize it by reading integers, one per line, from a file named `input.txt`. Some sample input files (that contain random sequences of unique integers) are provided. Please have your program output the full contents of the final sorted list to a file `output.txt`.