Algorithms and Data Structures

Summer Semester 2025

For discussion on Wednesday, May 28, 2025

1. (GTG Exercise R-6.5) Implement a function that reverses a list of elements by pushing them onto a stack in one order, and writing them back to the list in reversed order.

2. (GTG Exercise R-6.8) Suppose an initially empty queue Q has executed a total of 32 enqueue operations, 10 first operations, and 15 dequeue operations, 5 of which raised `Empty` errors that were caught and ignored. What is the current size of Q?

3. (GTG Exercise C-6.20) Describe a nonrecursive algorithm for enumerating all permutations of the numbers $\{1, 2, \ldots, n\}$ using an explicit stack.

4. (GTG Exercise C-6.21) Show how to use a stack S and a queue Q to generate all possible subsets of an n-element set T nonrecursively.

5. (GTG Exercise R-7.4) Describe in detail how to swap two nodes x and y (and not just their contents) in a singly linked list L given references only to x and y. Repeat this exercise for the case when L is a doubly linked list. Which algorithm takes more time?

6. (from GTG Exercise R-7.5) A circularly linked list is a list where the next pointer of the last element of a regular singly linked list points back to the starting node. Thus, the starting pointer should be seen as a "current" marker while the list itself is a loop.

   Implement a function that counts the number of nodes in a circularly linked list.

7. In the "move-to-front heuristic", each time an element in a linked list is accessed, it is moved to the first position. The idea is that it might be likely to be accessed again, and any direct subsequent access can be done in $O(1)$ time.

   (GTG Exercise R-7.20) Let L be a list of n items maintained according to the move-to-front heuristic. Describe a series of $O(n)$ accesses that will reverse L.

8. (GTG Exercise R-7.21) Suppose we have an n-element list L maintained according to the move-to-front heuristic. Describe a sequence of n^2 accesses that is guaranteed to take $\Omega(n^3)$ time to perform on L.

9. (GTG Exercise C-7.29) Describe in detail an algorithm for reversing a singly linked list L using only a constant amount of additional space and not using any recursion.