Lab 4 – Self Organizing List

# Saturday 11-07-2020

## Goals

* Review textbook material about self-organizing list
* Read and understand the lab description
* Start to plan implementation
* Start to build the count self-organizing list

## Notes

* I’ve read the lab description and my understanding is that I need to build a self-organizing list, in addition to keeping track of certain metrics for output.
* For my implementation, I don’t want to use the list.h file but instead I want to implement my own doubly linked list.
* I need to modify the link.h file to make it doubly linked.
* I need to review the author’s doubly linked list
  + I decided to use his double-linked link implementation, including the freelist
* I am going to implement the count heuristic first as that seems the most complex
* I encountered a problem that should have been simple to debug, but I wasn’t paying the close attention that was needed:
  + In order to implement the count function, I have a private updateList function that steps back through the list as long as the updated element has a greater count. In order to do this, I need to swap many pointers, and I thought I had done so correctly, but it turns out that I had been using a pointer that had already been updated to update another one, and this led to a very confusing bug where reading forward, the list updated properly, but in reverse it became a jumbled mess.
  + I used 4 full pieces of paper tracing through the swap algorithm several times until I finally saw what was happening.
  + After all the debugging I found a simpler solution to what I was doing

## Results

* My count sol works as expected and I’ve begun to write the test file for it.
* I still need to implement the other two heuristics and write the rest of main.cpp, but I don’t anticipate any major setbacks (of course, who does?) so I can work on those on Monday.

# Monday 11-09-2020

## Goals

* Finish other SOL heuristics
* Write main.cpp with file input tests
* Write any other tests to demonstrate my classes work correctly

## Notes

* I was originally going to make a superclass that implements the majority of the functionality and then each heuristic derives from that bass class (which derives from the ADT) and overloads the updateList method. Unfortunately, I got a weird error about abstract classes and such, so each heuristic will inherit from the ADT independently and just reuse the same functions.
* I’ve built the move to front heuristic. I had a fun little period of time where if the link being moved to the front was the tail, I didn’t unlink the previous link so it created an infinite loop.
* I need to start thinking about how I’m going to keep track of frequency to print out the strings
* I’ve finished the transpose heuristic and written main.cpp for the three heuristics using the char list.