Ben Pierce

EECS 233

Final Project- Progress Report

12/5/17

At this point in the project, the DynamicList class, which encapsulates my custom ArrayList, LinkedList, and HashTable classes is almost complete. Basic functionality (add,remove,get,etc..) is complete and tested. The most significant functionality still needed is a HashTable method that increases the size of the structure if it exceeds the HashTable capacity. Additionally, the methods that remove the largest element and search the data structure have not yet been rigorously tested. However, no difficulties are foreseen in this matter, as the code is fairly trivial.

I have also analyzed the results of the add in order method. I compared each data structure in the DynamicList to the standard library structures (ArrayList, LinkedList, Hashtable). For the ArrayList comparison, the results were as expected. The time of the custom data structure were slightly longer than the times for the built in structure. Interestingly, when complier optimization is disabled, the standard library takes much more time. This was also expected, as one of the advantages of using a well-known java.util library is these small optimizations. In contrast, the standard library linked list performed worse than my custom linked list. This is most probably due to the algorithm I used to add in place with the standard library, which needs to be changed to a more efficient version. Lastly, the Hashtable class and my custom hashtable behaved almost exactly the same (differing by 6ms, which may be due to poor randomization of the numbers inserted).

The files included are pretty self-explanatory. BasicArrayList.java is my custom ArrayList class, which accepts a generic type *T*. BasicHashTable.java is my custom hashtable, which was adapted from the textbook to accept a generic type. Likewise, BasicLinkedList is fairly similar in design to the textbook’s version, but again modified to accept generics. Both BasicArrayList and BasicLinkedList have a addInOrder() method, which adds in element to maintain a least-greatest sorted list. This method is omitted in BasicHashTable because the functionality is meaningless for a hash table. All three classes have a removeLargest() method, which does exactly what the title implies. Likewise, all three have a search() method, which conducts a linear search to find a generic target element. DynamicListTest is where I compare the custom structure to the standard structure.

One issue I am having is with the chooseStructure() method, because I am no longer convinced that the data structure should be in charge of switching its own state when arbitrary conditions are met. As a programmer, I would rather have manual control over the structure, with the knowledge that I can change it at any time, but it won’t decide to change on its own. Therefore, I think I’m going to scrap the idea of the structure changing on its own, and focus on manual control over the structure.