CS2028C Lab 7

Nicholas McClorey Daniel Wood, Kyle Van Blaricom February 25, 2019

Overview

The purpose of this lab was to explore different methods of implementing a class and observing the efficiency of different approaches. It's important to keep this in mind when working with data on a larger scale. This could affect the amount of time or memory a program uses to complete its task.

Task 3

The first class searches from the beginning to find an empty spot and does the same when removing an item. This is a straightforward approach that's easy to program.

The second class searches from the end to find an empty spot. This doesn't affect the number of "operations" because once it finds the empty spot, it does the same insertion as its base class. The time it takes to find the insertion spot may be affected, but this only involves checking for null pointers and doesn't affect the number of "operations."

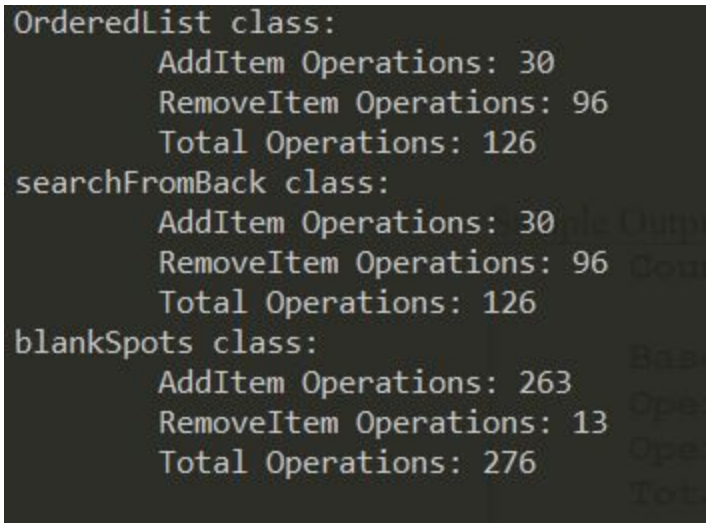
The final class adds items in the middle of the array. It has an efficient AddItem function but an inefficient RemoveItem function. When adding multiple items, each item has to be shifted each time. When removing an item, nothing has to be shifted. It'll be interesting to see whether the gains in the RemoveItem outway the losses in the AddItem

Task 4

The first two classes performed similarly. They differ only in how they find the spot to add the item and thus, have identical operation counts. The final class that added items to the middle was very inefficient. It's RemoveItem function was very efficient but this didn't make up for the woeful inefficiency of the AddItem method. The idea of adding items in the middle, shifting all the others and wrapping around from the end to the beginning is silly. It is unnecessarily confusing and doesn't provide any benefit.

Testing

To run the program 100 times, I created a batch file to run it 100 times.



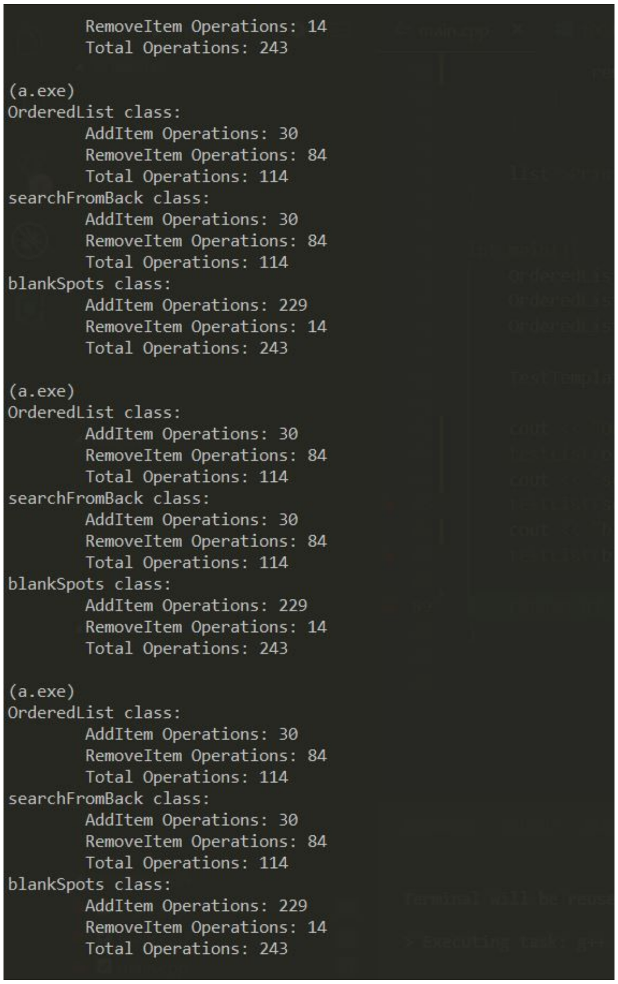
MAX_ITEMS = 20

```
OrderedList class:
        AddItem Operations: 30
        RemoveItem Operations: 59
        Total Operations: 89
searchFromBack class:
        AddItem Operations: 30
        RemoveItem Operations: 59
        Total Operations: 89
blankSpots class:
        AddItem Operations: 364
        RemoveItem Operations: 6
        Total Operations: 370
```

MAX_ITEMS = 50

```
OrderedList class:
        AddItem Operations: 30
        RemoveItem Operations: 44
        Total Operations: 74
searchFromBack class:
        AddItem Operations: 30
        RemoveItem Operations: 44
        Total Operations: 74
blankSpots class:
        AddItem Operations: 421
        RemoveItem Operations: 2
        Total Operations: 423
```

MAX_ITEMS = 100



```
        RemoveItem Operations: 14
        Total Operations: 243

(a.exe)
OrderedList class:
        AddItem Operations: 30
        RemoveItem Operations: 84
        Total Operations: 114
searchFromBack class:
        AddItem Operations: 30
        RemoveItem Operations: 84
        Total Operations: 114
blankSpots class:
        AddItem Operations: 229
        RemoveItem Operations: 14
        Total Operations: 243

(a.exe)
OrderedList class:
        AddItem Operations: 30
        RemoveItem Operations: 84
        Total Operations: 114
searchFromBack class:
        AddItem Operations: 30
        RemoveItem Operations: 84
        Total Operations: 114
blankSpots class:
        AddItem Operations: 229
        RemoveItem Operations: 14
        Total Operations: 243

(a.exe)
OrderedList class:
        AddItem Operations: 30
        RemoveItem Operations: 84
        Total Operations: 114
searchFromBack class:
        AddItem Operations: 30
        RemoveItem Operations: 84
        Total Operations: 114
blankSpots class:
        AddItem Operations: 229
        RemoveItem Operations: 14
        Total Operations: 243
```

Running the program 100 times

All the results point to the class inserting values in the middle as being the least efficient class. The other classes were identical in behavior because there is no case where they would perform identical operations and not have identical operation counts.

Compiling

This program should be compiled with the gcc or tdm compiler. The command to compile this program is “g++ main.cpp -std=c++14”. This program was compiled on Windows 10.