University of Waterloo
CS 234 - Data Types and Structures
Spring 2014
Assignment 3 - Solutions

Question 1: Testing Assumptions: Stock.py [25 marks]

In the file StockExchangeList.py find the worst case asymptotic time complexity of the following functions. In particular, identify any lines that have more than a O(1) worst case asymptotic time complexity and state their complexity. Also state the asymptotic worst case time complexity of the function as a whole.

a) __init__() [12 points]

Expensive Operations

`F = open(fname)` is O(1) but you didn't lose marks if you got this one wrong
`L = F.readlines()` is O(n)
`for line in L[2:]` is O(m)

Each column has at most 20 characters and each line has at most 12 x 20 = 240 characters. Both of these are constant sizes (i.e. they don't increase with n or m), so all the popping, conversions to ints and floats etc. is constant time. The only exception is `self._stocks.append(stock)` which is worst case O(m).
`self._stocks.append(stock)` is O(m)

Overall Cost

The overall cost is the max of the `F.readlines()` statement and the "`for line in L`" loop. All the statements in the for loop are O(1) except the `self._stocks.append()` command. But the cost of this command depends on the size of the `self._stocks` list so you cannot just multiply the cost of the loop times the number of times the loop iterates but must do a sum of the cost of each iteration (e.g. similar to the slide 102, Time Complexity of Loops: Example 3). This sum $1 + 2 + 4 + \ldots 2^k$ is in O(m) where k is the floor of log n. This analysis is similar to the one presented in the Amorized Cost slide.

So the overall cost is O(n+m) but O(n) = O(m) since the size of the file is bounded from above by m x 240 x the size of each character. So O(n+m), O(n) and O(m) are all acceptible answers.

b) add() [3 points]

Appending to a Python list of size m is worst case O(m) which is also the overall cost of the function.

c) remove() [3 points]

All the parameters of Stock are of constant size < 20 characters, so to create the stock is constant time. Remove is worst case O(m) and so the overall function is also O(m) .

d) getPrice() [4 points]

The asymptotic time complexity of getPrice() depends on the time complexity of _findStock() which in the worst case must search through the entire list and compare two strings, each with a size less than or equal to 20 characters. The asymptotic time complexity both for this step and the overall function is O(m).

e) sell() [3 points]

As in part d) _findStock() is O(m) and the rest of the function is testing and setting values of a fixed size to the overal cost is O(m).

Question 2: Course Project [50 marks]

For this part of the course project you will create a data structure that will track information that flows between your browser and websites you visit.

a) In a file called strCounts.py create a class called StringCounts. This class keep track of how many times a particular string has been "seen." [10 marks for clarity (i.e. well documented class and function headers, identified preconditions and postconditions, meaningful error message and variable names) as well as organization, simplicity and efficiency. code, 10 for passing our test cases].

   See the file strCounts.py in the Assignment section of Learn.

b) In a file called strCountsTest.py import the StringCounts class and test it thoroughly [5 marks].

   See the file strCountsTest.py in the Assignment section of Learn.

c) In a file called strCommands.py create a function called `getValue(anHTTPmsg, paramName)` where both the two parameters and the return type are strings. Create a second function called `hasParam(anHTTPmsg, paramName)` that given anHTTPmsg, as above, and a paramName, returns True if anHTTPmsg has paramName as a parameter name or false otherwise. [10 marks for clarity (i.e. well documented class and function headers, identified preconditions and postconditions, meaningful error message and variable names) as well as organization, simplicity and efficiency. code, 10 for passing our test cases]

   See the file strCommands.py in the Assignment section of Learn.

d) In a file called strCommandsTest.py import the functions `getValue()` and `hasParam()` and test them thoroughly. Use your programs from the previous two assignment to generate a few HTTP commands from your browser to get some "correct" HTTP commands. You can also edit these commands to create some "incorrect" HTTP commands. [5 marks]

   See the file strCommandsTest.py in the Assignment section of Learn.