Homework 3

Assigned: 10/12/2023

Due: 10/26/2023 at 11:59 PM

WARNING. Students may not work together. Students may discuss the problems with each other, but do not give any other student your solutions.

Aside: This file is written using markdown. markdown renders reasonably well inside pycharm. If you use pandoc, markdown can be converted to a pdf file.

Place all files containing your answers to homework 3 in a directory named `hw3_last_first` where `last_first` is the student's last and first names separated by an underscore. For me, the directory would be `hw3_harrison_david`.

Place each answer in its own file or directory. When you are done your directory structure should look like.

```
$ ls -F
hw3_harrison_david/
$ cd hw3_harrison_david
$ ls -F
p1.txt  p2/  p3.txt  p4/  p5.txt  p6/ p7/
```

Zip or tar the directory `hw3_last_first` and submit them to blackboard in the same manner as was done for homework 1. On Mac OS or linux, it would look like this:

```
$ ls -F
hw3_harrison_david/
$ tar -czf hw3_harrison_david.tgz hw3_harrison_david
$ ls -F
hw3_harrison_david/ hw3_harrison_david.tgz
```

Submit `hw3_harrisond_david.tgz` to blackboard. If you are on windows, you may use zip, in which case the file submitted would be `hw3_harrison_david.zip`.

Problems 1 (1 point each) is Problem 1 in Discussion Questions in 4.26. of *Problem Solving with Algorithms and Data Structures using Python*. Put the answer in a text file named `hw3_last_first/p1.txt`.

Problem 2: (2 points)

(a) Create a `LinkedList` class. It must pass the `hw3/p2/test_linked_list.py`. The class MUST not use any Python built-in or standard library collection class, i.e., do not wrap a list or deque.

(b) Copy the Stack implementation found in the repository

https://git.cs.olemiss.edu/harrison/csci-356

in `lecture13and14/stack.py` into your homework directory `hw3_last_first/p2`, rename the file to `linked_list_stack.py`. In the file `linked_list_stack.py` rename the class `Stack` to `LinkedListStack`, and modify it so that it is implemented using your `LinkedList`. It must pass the unit tests in the repository in the directory `hw3/p2/test_linked_list_stack.py`. It MUST use your `LinkedList`. the new `LinkedListStack` class MUST NOT use any Python built-in or standard library collection class, i.e., the LinkedListStack class MUST not wrap a `list` or `deque`.

Problem 3 (1 point each) is Problem 3 in Discussion Questions in 4.26. of *Problem Solving with Algorithms and Data Structures using Python*.

Problem 4: (2 points) Use the `Queue` found in the source code repository in `hw3/p4/queue.py`, which is based on the code in the book in Listing 1 of Section 4.12

(a) Create file `perftest_list_queue_a.py` in `hw3_last_first/p4` whose `main` function enqueues n random integers into m Queue objects according to the following pseudocode:

```
create m empty `Queue` objects and put them in a list named queues.
for n in some range:
    start timer
    for x in queues:
        enqueue the nth random integer into queue x
    stop timer
    divide the elapsed time by m to get an average
    append the average time for an execution of enqueue to a list of times.
```

Using `matplotlib` have your code plot the average execution time for a call to `enqueue()` as a function of n. Vary n at least to 10,000. You may skip n by increments of 10, but if you do then adjust the x-axis accordingly.

(b) Analyze the performance of the `enqueue()` method using big-O notation. Put your analysis in a file named `hw3_last_first/p4/b.txt`.

(c) In a file `perftest_list_queue_c.py` Create a variant of the code created for (a) that starts by creating m Queues of the largest length (e.g., $n = 10000$) and then dequeues one item from each list while recording the average time for a dequeue. Using `matplotlib` plot the average execution time for a call to `dequeue()` as a function of n. If it runs too slowly you can start with $n = 1000$. You may skip n by increments of 10, but if you do then adjust the x-axis accordingly.

(d) Analyze the performance of the `dequeue()` method using big-O notation and put your analysis in a file named `hw3_last_first/p4/d.txt`.

Problem 5 (1 point) is Problem 5 in Discussion Questions in 4.26. of *Problem Solving with Algorithms and Data Structures using Python*.

Problem 6: (2 points)

Repeat problem 4, but implement a `Queue` using a Python `dequeue`. Generate the plots and analyze the `enqueue` and `dequeue` methods using big-O notation in the same manner. In the plots for (a) and (c) include the plot for the same scenario but using the list implementation of the `Queue`. This way we can visually compare the performance of the `list` and `deque` implementations.

Problem 7 (1 point) is Problem 11 in Programming Exercises in 4.27. of *Problem Solving with Algorithms and Data Structures using Python*. It must pass the unit tests for `p7/test_html_balance.py`.