PA - 1
Suggested Due Date: Sunday Dec 4, 2022
Playing with Stacks!

Description:

For this programming assignment, you will implement a **Stack ADT** whose **size can grow** as elements are inserted into the stack. You will **build four different implementations** of the Stack ADT. The **first two implementations** will be **array-based** (list in Python). Each of these two should take an **initial capacity** for the stack in the **constructor**. The only difference between these implementations will be what happens when the Stack is full.

- For the first implementation, **ArrayStack**, you should increase the size of the array by a **constant amount**.
- For the second implementation, **DoublingArrayStack**, you should **double the size** of the array.
- For the third implementation, **LinkedListStack**, you will build a Stack using **Linked List**.
- For comparison, you will also include a fourth implementation that uses the **deque structure from the collections module** in Python to see an optimized pre-built implementation.

You will **time each of your implementations**. For each version of your stack, push a large amount of random numbers onto the stack (e.g. 1,000,000). Time how long this takes using the **time module** in Python which you may import (an example of the usage is provided in the starter code). However, you should measure the total time at **fixed intervals** (e.g. at every 10000 push operations). For the array-based implementations, you should start with some small capacity significantly smaller than the maximum number of push operations. For the ArrayStack, your increment amount should also be significantly smaller than the number of push operations.

You will then **graph the runtimes for the four implementations**. This process will allow you to visualize how your choice of implementation can affect performance.

Coding Portion (50 Points):

- You will be using the concepts of Object Oriented Programming to execute the implementations through classes for each of the versions (except the fourth).
- Start with the following template: *Stack.ipynb*, and fill in all of the member functions. Use the **ArrayStack** class as a template for both **ArrayStack** and **DoublingArrayStack.**
- For the **Linked List** implementation, you will need to create another class for the individual nodes in the Linked List.
- Be sure to test the correctness of your algorithms and implementations. Ensure all stack operations of stack ADT (push, pop, top, size, isEmpty etc.) are implemented and tested to be working as expected.
- Your code will be graded based on whether or not it runs (10 points), produces correct output on test cases (30 points), and your coding style (does the code follow proper practices/style and comments, 10 points).

Report (50 Points):
You will write a brief report that includes theoretical analysis, a description of your experiments, and discussion of your results. At a minimum, your report should include the following sections:

1. *Introduction*. In this section, summarize the objective of this assignment in your own words.
2. *Theoretical Analysis*. In this section, provide an analysis of the expected performance and growth rate based on each of your implementations and based on what has been covered in the lectures so far about Stack and Linked List. Describe the effect of a push operation and the advantages and disadvantages of the two strategies.
3. *Experimental Setup*. In this section, provide a description of your experimental setup, which includes but is not limited to:
    a. Machine specification
    b. How did you generate the test inputs? What input sizes did you test? Why?
    c. How many times did you repeat each experiment?
4. *Experimental Results*. In this section, compare the performance (runtime) of the push() operation in the four different implementations to one another and to your theoretical analysis.
    a. Make a plot showing the runtime (y-axis) vs. the number of push operations (x-axis). You must use **matplotlib** or similar visualization tools to create the plot. **Hand-made plots will NOT be accepted.** Make sure it is appropriately labeled.
    b. Provide a discussion of your results, which includes but is not limited to:
        i. Which of the three Stack implementations performs the best? Does it depend on the input? Provide some reasoning behind your observation.
        ii. To what extent does the theoretical analysis agree with the experimental results? Attempt to understand and explain any discrepancies you note.

Submission:

Once you have completed the assignment, you should upload the python script to the **PA-1** codePost portal (this portal will be made accessible over the weekend, you will receive an invitation by email). Please ensure the following while submitting:

- Once satisfied with your code, you should download the file as a python script (.py file), by going to **File** > **Download** > **Download .py**
- The name of the file should be **PA1.py**
- Upload the python script file to codePost under the PA-1 assignment
- You can run the test cases on your script up to a limit of 50 times
- Once satisfied with the test runs, complete your submission
- The reports should be submitted to the PA-1 assignment portal on BrightSpace