LAB03: Estimate the order of growth
CSE2050 Fall 2017
Instructor: Jeffrey Meunier & Wei Wei
TA: Jenny Blessing, Param Bidja, Yamuna Rajan & Zigeng Wang

1. Introduction

In this exercise, we will use the time() function in time module to estimate the order of growth for running time of three different functions.

2. Objectives

The purpose of this assignment is to help you:

- Refresh knowledge on order of growth of running time and big-O notation.
- Learn how to write simple code to sort a list.
- Do a little bit study on different modules in python.

3. Background

3.1. Time complexity

As computer science students, writing code is something you do every day. It is important to make sure we write efficient code. How do you know the code you have written is good enough? A good way is to find out the time complexity using the big-O notation. The big-O notation describes the order of growth of running times when input size increases. Note in the big-O notation, the running time considered is always the worst-case running time.

GROWTH RATE	NAME
O(1)	Constant time
O(n)	Linear time
O(n^2)	Quadratic time
O(n^3)	Cubic time

4. Assignment

In the skeleton zip file, you can find a skeleton code, named *timesumksquare.py*. All you need to do is to complete the skeleton code based on the following instructions and submit the *timesumksquare.py* to the Mimir system.

4.1. Selection Sort

Now, open the skeleton code. There are three functions for you to complete. The first function is the selectionsort function where unsorted elements are sorted in ascending or descending order. The function's skeleton is as the following:

```
def selectionsort(L):          # L is the list to be sorted
    start = time.time()        # record the start of the function
    ### Add your code here
    end = time.time()          #record the end of the function
    return L, end - start      #return the sorted list and the total time taken to sort the list
```

In the selectionsort function, we first store the start time of the function in the start variable by using the time() function.

We then implement a simple selection sort algorithm. The algorithm works like the following.

We loop the index i from 0 to len(L) – 2. In this loop, we compare L[i] to each item to the right. If a item is less than L[i], we switch these two. When this is done for all the i's, we should have a sorted list.

Once the sorting has completed, the time is once again recorded in the end variable. This allows us to calculate the time taken by the function.

4.2. Sum of squares of k numbers without formula

The second function we need to implement is the sumksquare. In this function, we simply calculate the sum of squares of the first k positive integers. Just as we did in selectionsort, the start time and end time are recorded and the total time taken by the function is returned along with the calculated sum.

4.3. Sum of squares of k numbers with formula

The third function that we need to implement is the sumksquare2. Just like in the previous function, this function also calculates the sum of squares of the first k positive integers. But this function uses a formula to calculate the sum. The formula is as follows:

```
sum = k*(k + 1)*(2*k + 1)/6
```

5. Estimating the order of growth

When you are done with the coding of each function above, count the number of atomic operations and predict the order of growth of the running time use the big-O notation. When you run your code, you can check whether your prediction is the same as the measured order of growth.

The following line is used to estimate the order of growth of each function:

```
int(statistics.mean([math.log(T[i+1]/T[i], 10) for i in range(len(T)-1)])+ 0.5)
```

Four different input sizes (where each input size is greater than the previous one by ten fold) are provided to the three functions and a number of trials are performed. The average running time of a function is the total time used divided by the number of trials. The order of growth is calculated by taking the mean of the log of the ratio of the time required for an input size to the time required by the previous input size. For example, if the ratio of running time is 100, then we have $\log_{10}(100) = 2$. This means that the estimated order of growth is n^2. On the hand, if the ratio of running time is 10, then $\log_{10}(10) = 1$. This implies that the estimated order of growth in running time is linear, because the input size increased by 10 fold, and the running time also increased by 10 fold. We obtain a number of such estimates and use mean() function from the statistics module to get the average.

In order to obtain an order of growth that takes an integer value, we add this average by 0.5 and then use the int() function. This way, we can round the order of growth instead of getting the floor of it.

6. Submit your work to Mimir

Submit your code timesumksquare.py to Mimir after you complete your code. Before you submit the code, **make sure to comment out the lines apart from the three functions** (selectionsort, sumksquare, sumksquare2). This is because the program takes too long to run, which is not accepted by the Mimir system. The output displaying the order of growth is for your understanding. The Mimir will automatically grade your submission based on different unit tests. You can submit your code to Mimir up to **30 times** to refresh your existing score before the submission deadline.

7. Due date

This lab assignment is worth **2 points** in the final grade. It will be due by **11:59pm on Wednesday, Sep 27th 2017**. A penalty of **10% per day** will be deducted from your grade, starting at 12:00am.

8. Getting help

Start your project early, because you will probably not be able to get timely help in the last few hours before the assignment is due.

- Go to the office hours of instructors and TAs.
    - Prof. Wei Wei: Mon. 2:30 - 3:15pm, Wed. 2:30 - 3:15pm, Thurs. 2:30 - 3:15pm, Fri. 2:30 - 3:15pm @ITE258
    - Jenny Blessing: Fri. 12pm - 2pm @ITE140
    - Param Bidja: Tues. 2pm - 3pm @ITE140
    - Yamuna Rajan: Tues. 11am - 12pm, Wed. 9:30am – 10:30am @ITE140
    - Zigeng Wang: Mon. 3pm - 4pm @ITE140
- Post your questions on Piazza. TAs and many of your classmates may answer your questions.

- Search the answer of your unknown questions on the Internet. Many questions asked by you might have been asked and answered many times online already.