

CSCI-SHU 210 Data Structures

Recitation 2 Complexity

You have a series of tasks in front of you. Complete them! Everyone should code on their own computer, but you are encouraged to talk to others, and seek help from each other and from the TA/LAs.

Question 1 Permutation

Suppose you need to generate a *random* permutation from 0 to N-1. For example, {4, 3, 1, 0, 2} and {3, 1, 4, 2, 0} are legal permutations, but {0, 4, 1, 2, 1} is not, because one number (1) is duplicated and another (3) is missing. This routine is often used in simulation of algorithms. We assume the existence of a random number generator, *r*, with method `randInt(i,j)`, that generates integers between *i* and *j* with equal probability. Here are three algorithms:

1. Fill the array *a* from *a*[0] to *a*[N-1] as follows: To fill *a*[*i*], generate random numbers until you get one that is not already in *a*[0], *a*[1], . . . , *a*[*i*-1].
2. Same as algorithm (1), but keep an extra array called the *used* array. When a random number, *ran*, is first put in the array *a*, set *used*[*ran*] = *true*. This means that when filling *a*[*i*] with a random number, you can test in one step to see whether the random number has been used, instead of the (possibly) *i* steps in the first algorithm.
3. Fill the array such that *a*[*i*] = *i*. Then

```
for i in range(len(array)):

    swap( a[ i ], a[ randint( 0, i ) ] );
```

Your task:

Code random permutation algorithm in three given ways.
They do the same job, but different implementation.

Example:

Input:	[0,1,2,3,4,5,6]
Possible Output using permutation1:	[4, 5, 2, 0, 1, 6, 3]
Possible Output using permutation2:	[0, 4, 3, 2, 1, 5, 6]
Possible Output using permutation3:	[5, 6, 0, 4, 1, 3, 2]

Question 2 Plotting run time

We are going to plot three versions Prefix Averages algorithm, using python matplotlib library.

If you are using Anaconda python, you can skip all installations!!!!

1. You may not have matplotlib library, so we are going to install matplotlib module first.

```
Python 3.6.2 (v3.6.2:5fd33b5926, Jul 16 2017, 20:11:06)
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import matplotlib
>>> █
```

No Error

Windows:

1. Add python path to your command line prompt
2. Install pip
3. Use pip to install python modules.

Step 1: Add python path to your command line prompt

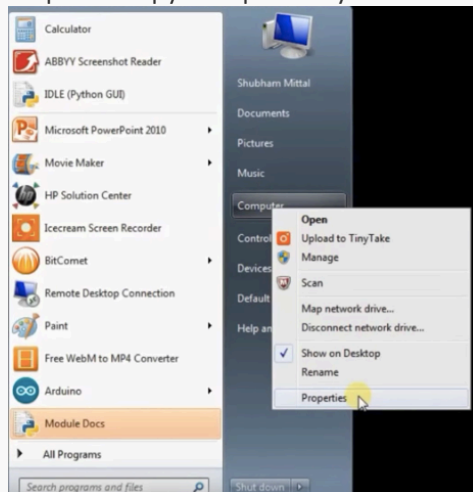


Figure 1: Right Click Computer, go to Properties.

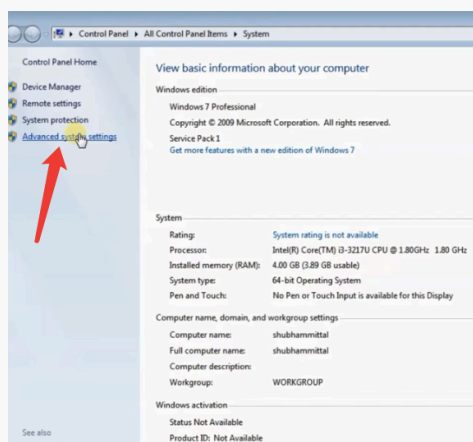


Figure 2: Select "Advanced System Settings"

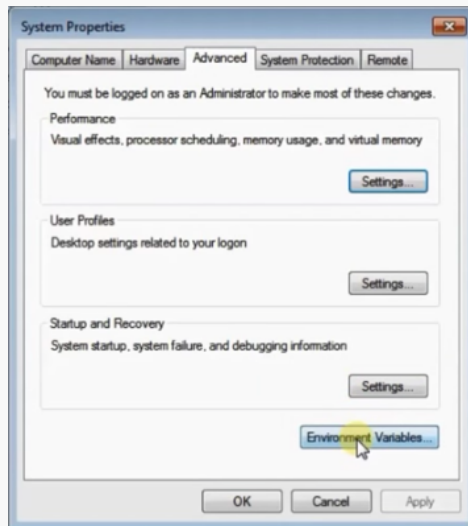


Figure 3: Select “Environment Variables”

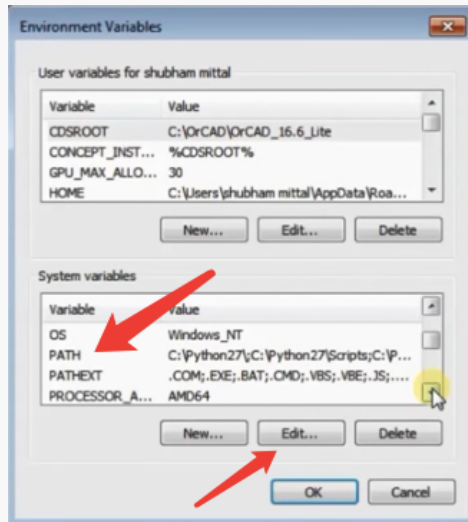


Figure 4: Search for Path variable, select it, then click “Edit”



Figure 5: Add your python path.

For example, if your python is installed at `D:\Python36`

You should append `;D:\Python36; D:\Python36\Scripts` at the end.

Step 2. Install pip

Download get-pip.py, save as .py file.

<https://bootstrap.pypa.io/get-pip.py>

then in command line prompt, execute

\$ python get-pip.py

3. Use pip to install python modules.

```
$ pip install matplotlib
```

or

```
$ python -m pip install matplotlib
```

MacOS:

1. Install pip

```
$ sudo easy_install pip
```

2. Use pip to install python modules.

```
$ python -m pip install matplotlib
```

Your task:

Plot three Prefix_Average() functions' runtime, with input size equal to:

[1000, 2000, 3000, 4000, 5000, 6000, 7000, 8000, 9000, 10000]

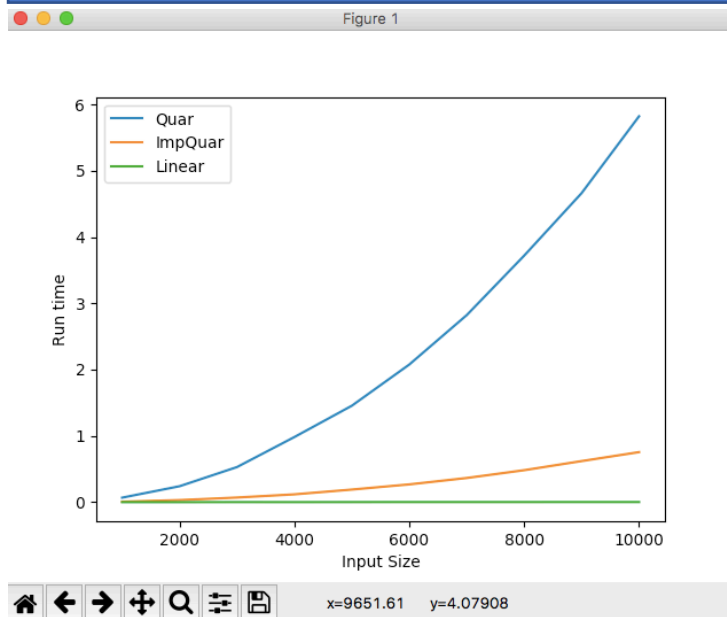


Figure 6. Your result should looks like this:

matplotlib functions explained:

```

import matplotlib.pyplot as plt


line1 = plt.plot(x, y, label = "Linear")
# plot a line in the graph.
# takes two lists of values, first list contains x values, second list contains y values
# returns the line you just plotted

plt.xlabel("Input size")
# Gives x axis label name.

plt.ylabel("Run time")
# Gives y axis label name.

plt.legend(handles=[line1,line2,line3])
# Adds legend, like this:

```



```

plt.show()
# Shows graph on the screen, usually called at the end.

```

So what am I supposed to do?

1. For each input size x, create an empty list
2. Fill the list with random values
3. Run prefix_average1, prefix_average2, prefix_average3
4. Use timeFunction() to monitor runtime
5. Save this runtime to y_values list.