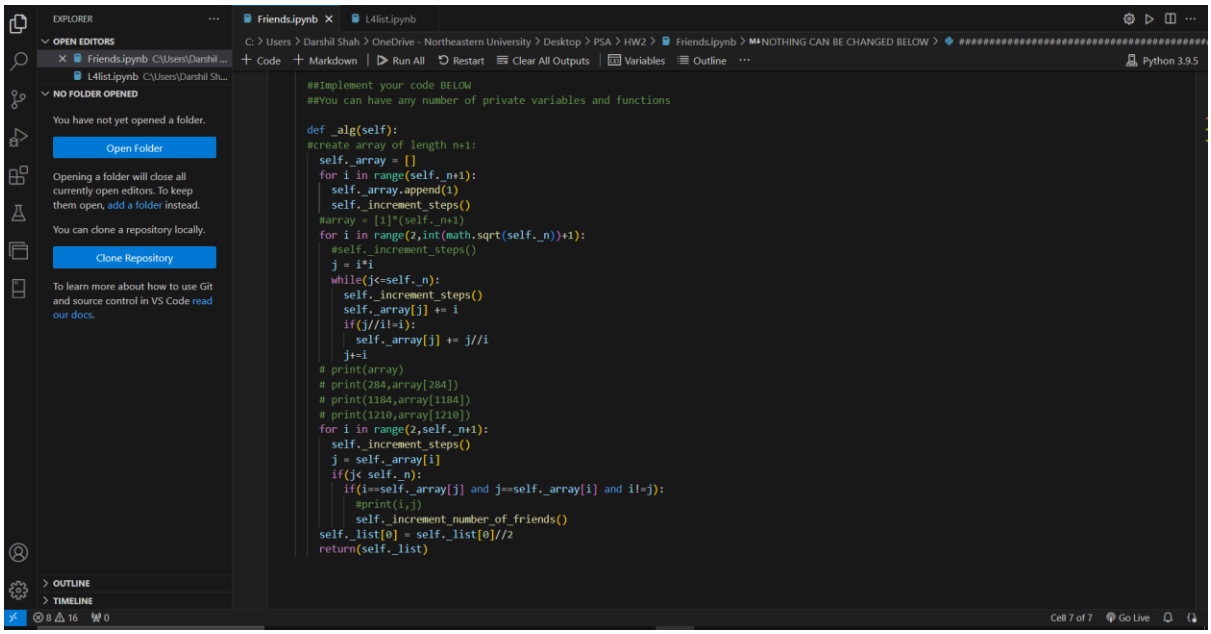
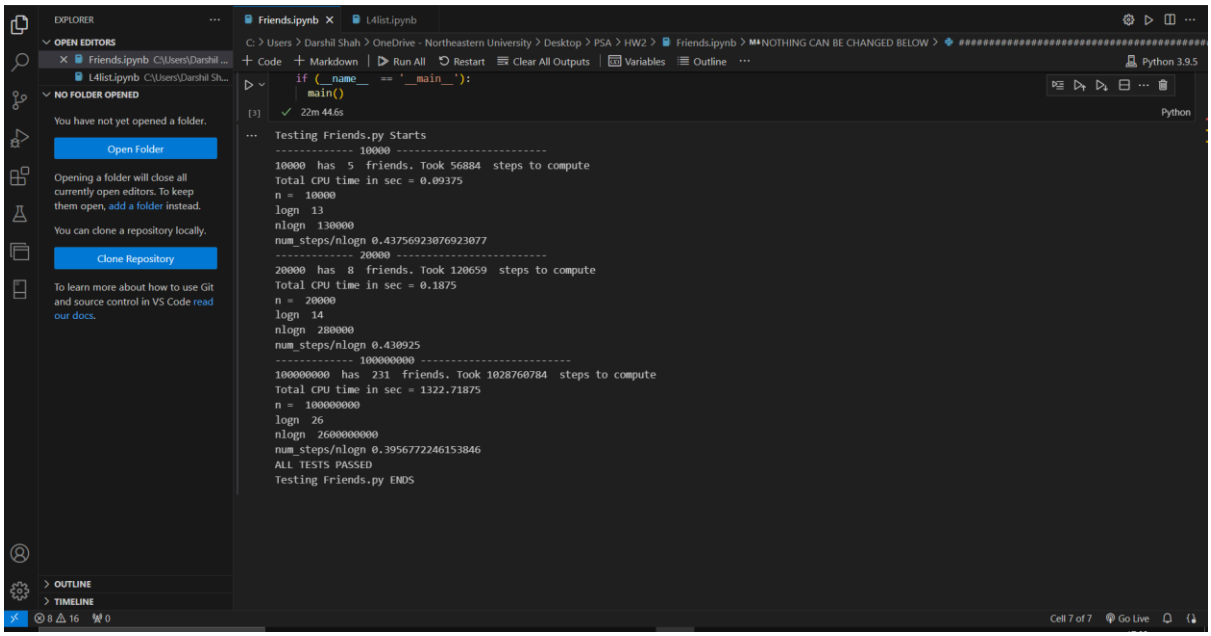
Program Structures and Algorithms

Homework Assignment -2:



```
##Implement your code BELOW
##You can have any number of private variables and functions

def _alg(self):
#create array of length n+1:
  self._array = []
  for i in range(self._n+1):
    self._array.append(1)
    self._increment_steps()
  #array = [1]*(self._n+1)
  for i in range(2,int(math.sqrt(self._n))+1):
    #self._increment_steps()
    j = i*i
    while(j<=self._n):
      self._increment_steps()
      self._array[j] += i
      if(j//i!=1):
        self._array[j] += j//i
      j+=i
  # print(array)
  # print(284,array[284])
  # print(1184,array[1184])
  # print(1210,array[1210])
  for i in range(2,self._n+1):
    self._increment_steps()
    j = self._array[i]
    if(j< self._n):
      if(i==self._array[j] and j==self._array[i] and i!=j):
        #print(i,j)
        self._increment_number_of_friends()
  self._list[0] = self._list[0]//2
  return(self._list)
```



```
if (__name__ == '__main__'):
  main()
```

```
Testing Friends.py Starts
------------- 10000 -------------------------
10000  has  5  friends. Took 56884  steps to compute
Total CPU time in sec = 0.09375
n =  10000
logn  13
nlogn  130000
num_steps/nlogn 0.43756923076923077
------------- 20000 -------------------------
20000  has  8  friends. Took 120659  steps to compute
Total CPU time in sec = 0.1875
n =  20000
logn  14
nlogn  280000
num_steps/nlogn 0.430925
------------- 100000000 -------------------------
100000000  has  231  friends. Took 1028760784  steps to compute
Total CPU time in sec = 1322.71875
n =  100000000
logn  26
nlogn  2600000000
num_steps/nlogn 0.3956772246153846
ALL TESTS PASSED
Testing Friends.py ENDS
```

Approach used to solve the above problem is similar to solve prime number problem using Sieve of Eratosthenes.

Steps:

1. Create an array of value 1 and of length n+1 where n is the number of integers present in the list.
2. Use the outer loop till the square root of n and use the inner loop to start over the square of the index of the outer loop till the last index of the list. Now at every multiple of the outer index add the index value and find the divisor value.
   For example:
   Outer loop value : 2

Inner loop value : start: 4 , last loop value: n
Divide 4 with 2 and find the another factor which is 4//2 = 2 and since 2 is already added to index 4 don't add 2 again.
As the outer loop overs till square root of n and inner loop over the multiples of that number we get complexity of nlog(log(n)) (As explained by professor in last class)

1728

Euler

$$\frac{n}{2} + \frac{n}{3} + \frac{n}{5} + \frac{n}{7} + \cdots + \frac{n}{\sqrt{n}}$$

$$n\left[\frac{1}{2} + \frac{1}{3} + \frac{1}{5} + \frac{1}{7} + \cdots + \frac{1}{\sqrt{n}}\right]$$

$\log\log n$

$n \log n$

$n \log(\log n)$

3. After outer loop is completed, we would get array with all the factors for the number present in the array.
   Code for above part:

```
self._array = []
for i in range(self._n+1):
    self._array.append(1)
    self._increment_steps()
#array = [1]*(self._n+1)
for i in range(2,int(math.sqrt(self._n))+1):
    #self._increment_steps()
    j = i*i
    while(j<=self._n):
        self._increment_steps()
        self._array[j] += i
        if(j//i!=i):
            self._array[j] += j//i
        j+=i
# print(array)
# print(284,array[284])
# print(1184,array[1184])
# print(1210,array[1210])
```

4. Now loop over the array and check for each index and index value and for the corresponding index value check in array if at the index we can find the "Friend".
   Code for the above part:

```
    for i in range(2,self._n+1):
        self._increment_steps()
        j = self._array[i]
        if(j< self._n):
            if(i==self._array[j] and j==self._array[i] and i!=j):
                #print(i,j)
                self._increment_number_of_friends()
    self._list[0] = self._list[0]//2
    return(self._list)
```