Data Structures & Algorithms – Lab #1

Aim: Getting Familiar with Algorithms and Algorithm Analysis

Topics:

1. Algorithm Analysis
2. Running Time (Time Complexity) of an Algorithm
3. **time** Module in Python

Lab Questions –

Q1 – Write a function that takes ***n*** as a positive integer number and computes the sum of the first ***n*** integers. Calculate the running time of the function when you call it for n = 1000000.

Answer – We will write the code in three ways (using three functions) and will compare their running times using time module. sum_3 is the fastest algorithm and sum_2 is the slowest algorithm, because of one unnecessary statement in the loop (num = i).

```
import time
def sum_1(n):
    start = time.time()
    summation = 0
    for i in range(1, n+1):
        summation = summation + i
    end = time.time()
    return summation, end-start

result = sum_1(1000000)
print("First Function -> sum_1()")
print("Sum: %d, running time: %.7f" %result)
```

```
First Function -> sum_1()
Sum: 500000500000, running time: 0.7426772
```

```
import time
def sum_2(n):
    start = time.time()
    summation = 0
    for i in range(1, n+1):
        num = i
        summation = summation + num
    end = time.time()
    return summation, end-start

result = sum_2(1000000)
print("Second Function -> sum_2()")
print("Sum: %d, running time: %.7f" %result)
```

```
Second Function -> sum_2()
Sum: 500000500000, running time: 1.0119321
```

```python
import time
def sum_3(n):
    start = time.time()
    summation = n*(n+1)/2
    end = time.time()
    return summation, end-start

result = sum_3(1000000)
print("Third Function -> sum_3()")
print("Sum: %d, running time: %.7f" %result)
```

```
Third Function -> sum_3()
Sum: 500000500000, running time: 0.0000000
```

Q2 – Write a code to count even numbers between 1 and 1000000. Calculate the running time of the code.

Answer – We can write the code in three ways. In the last code, there is no for loop, that's why it is fastest one.

```python
import time
start = time.time()
counter = 0
for i in range(1, 1000001):
    if i%2 == 0:
        counter+=1
end = time.time()
print("Total number of even numbers:", counter)
print("Running time of the code:", end-start)
```

```
Total number of even numbers: 500000
Running time of the code: 1.0620496273040771
```

```python
import time
start = time.time()
counter = 0
for i in range(2, 1000001, 2):
    counter+=1
end = time.time()
print("Total number of even numbers:", counter)
print("Running time of the code:", end-start)
```

```
Total number of even numbers: 500000
Running time of the code: 0.43848299980163574
```

```python
import time
start = time.time()
a = list(range(2, 1000001, 2))
counter = len(a)
end = time.time()
print("Total number of even numbers:", counter)
print("Running time of the code:", end-start)
```

```
Total number of even numbers: 500000
Running time of the code: 0.009451866149902344
```

Q3 – Write a code that reverse a long text. Calculate the running time of the code.

Answer – We can first create a long text by repeating a short string. For example, **"Good Afternoon"*10000** is a long text in which the string "Good afternoon" is repeated 10000 times.

Now we can write the code in two ways. The second code uses a string slice that starts at the end of the string and moves backwards. This way is faster than the first code that uses for loop.

```
import time

start = time.time()

text = "Good Afternoon"*10000
reversed_text = ""
for ch in text:
    reversed_text = ch + reversed_text

end = time.time()

print("Running time: %.10f" %(end-start))
print("\n")
```

```
Running time: 0.5793070793
```

```
import time

start = time.time()

text = "Good Afternoon"*10000
reversed_text = text[::-1]

end = time.time()

print("Running time: %.10f" %(end-start))
print("\n")
```

```
Running time: 0.0000000000
```