MI545

Ozan Yetkin | 1908227

> Exporting distorted the document a bit. In order to view this document online, please visit:
>
> MI545

```
https://mysterious-snowman-10b.notion.site/MI545-1a56349211504e22970025b1b3b8c4ab
```

Question 1

Find the algorithmic complexity of each of the following code snippets in big-O notation, write the tight bound (Θ) if possible:

```
for i in range(n):
    k += 1
```

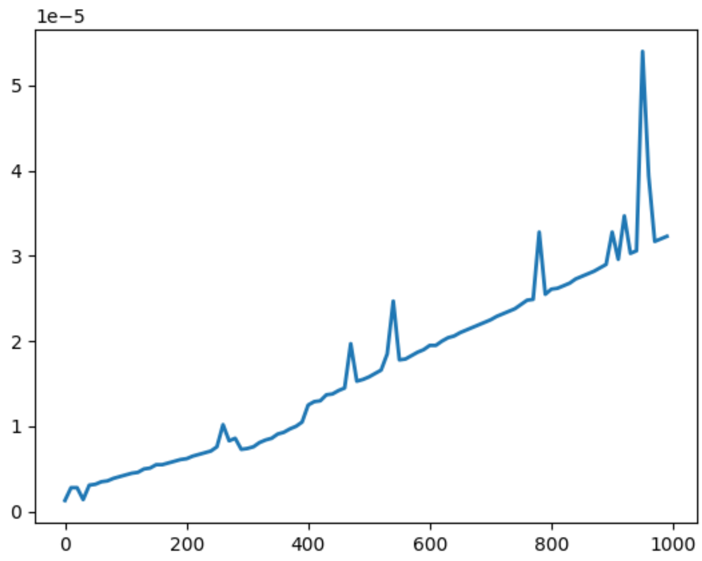
```
for i in range(n):
```

$$T_1 = c_1 \times n$$

```
    k += 1
```

$$T_2 = c_2 \times (n-1)$$

$$T_1 + T_2 = c_1 n + c_2(n-1) = \Theta(n)$$




```
i = 1
while i < n*n:
    i += 2
```

```
i = 1
```

$$T_1 = c_1$$

```
while i < n*n:
```

$$T_2 = c_2 \times \sum_{i=1}^{n} t_i$$

```
    i += 2
```

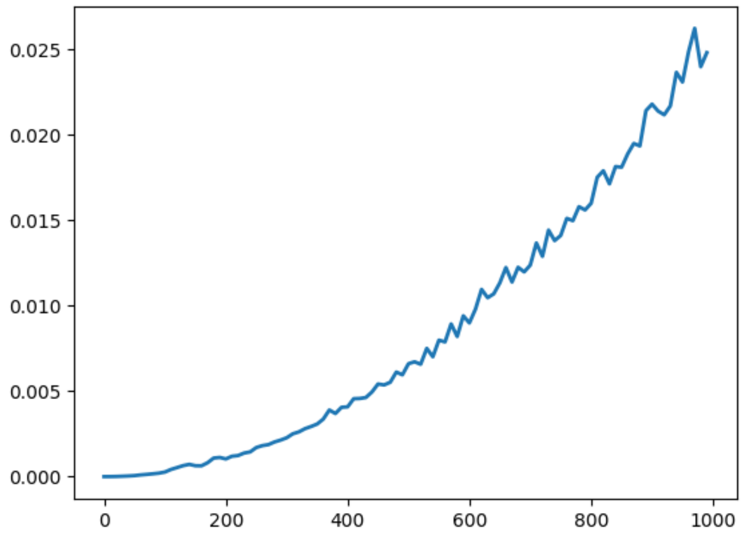
$$T_3 = c_3 \times \sum_{i=1}^{n} (t_i - 1)$$

$$T(n) = c_1 + c_2 \sum_{i=1}^{n} t_i + c_3 \sum_{i=1}^{n} (t_i - 1)$$

$$\text{Algorithm stops when } i = n^n \to log_n(i) = 2$$

$$\sum_{i=1}^{n} t_i = \sum_{i=1}^{n} (log_n(i) + 2) = log_n(n!) + 2n$$

$$T(n) = c_1 + c_2(log_n(n!) + 2n) + c_3(log_n(n!) + 2n - 1) = \Theta(log(n!))$$




```
k = 1
for i in range(n):
    for j in range(i+1,n):
        k += 1
```

```
k = 1
```

$$T_1 = c_1$$

```
for i in range(n):
```

$$T_2 = c_2 \times n$$

```
    for j in range(i+1,n):
```

$$T_3 = c_3 \times (n-1) \times \sum_{i=1}^{n} t_i$$

```
k += 1
```

$$T_4 = c_4 \times (n-1) \times \sum_{i=1}^{n}(t_i - 1)$$

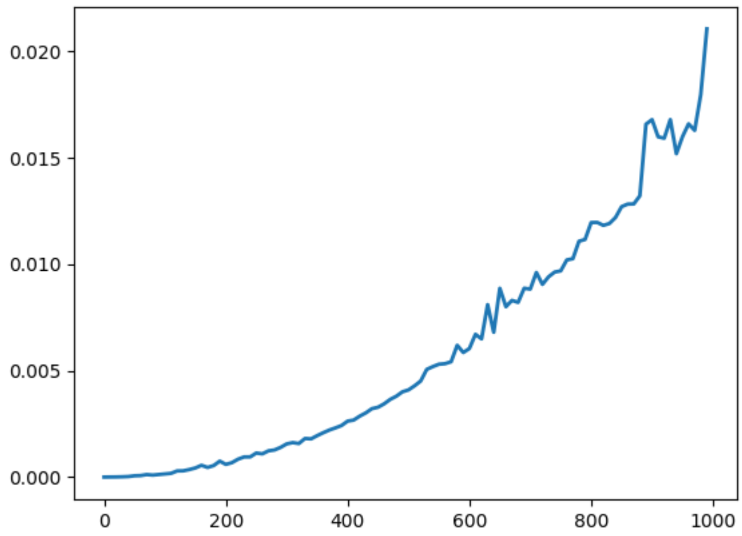
$$T(n) = c_1 + c_2 n + c_3(n-1)\sum_{i=1}^{n} t_i + c_4(n-1)\sum_{i=1}^{n}(t_i - 1)$$

$$\text{Algorithm stops when i = n} \rightarrow n - i = 0$$

$$\sum_{i=1}^{n} t_i = \sum_{i=1}^{n}(n-i) = n^2 - \frac{n(n+1)}{2} = \frac{n^2 - n}{2}$$

$$T(n) = c_1 + c_2 n + c_3(n-1)\frac{(n^2-n)}{2} + c_4(n-1)(\frac{n^2-n}{2} - n)$$

$$T(n) = c_1 + c_2 n + c_3\frac{n^3 - 2n^2 + n}{2} + c_4\frac{n^3 - 4n^2 + 3n}{2} = \Theta(n^3)$$



```
i = 1
while i < n*n*n:
    i = i + (2*n)
```

```
i = 1
```

$$T_1 = c_1$$

```
while i < n*n*n:
```

$$T_2 = c_2 \times \sum_{i=1}^{n} t_i$$

```
    i = i + (2*n)
```

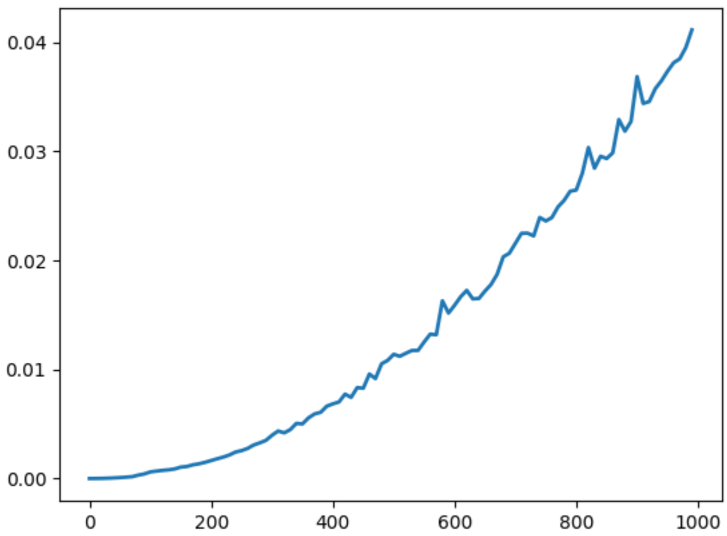
$$T_3 = c_3 \times \sum_{i=1}^{n}(t_i - 1)$$

$$T(n) = c_1 + c_2 \sum_{i=1}^{n} t_i + c_3 \sum_{i=1}^{n} (t_i - 1)$$

$$\text{Algorithm stops when } i = n^3 \rightarrow log_n(i) = 3$$

$$\sum_{i=1}^{n} t_i = \sum_{i=1}^{n} log_n(i) + \sum_{i=1}^{n} 2n = log_n(n!) + 2n^2$$

$$T(n) = c_1 + c_2(log_n(n!) + 2n^2) + c_3(log_n(n!) + 2n^2 - n) = \Theta(n^2)$$



```
i = 1
while i < n*n:
    i = i + (n // 2)
```

```
i = 1
```

$$T_1 = c_1$$

```
while i < n*n:
```

$$T_2 = c_2 \times \sum_{i=1}^{n} t_i$$

```
    i = i + (n // 2)
```

$$T_3 = c_3 \times \sum_{i=1}^{n} (t_i - 1)$$

$$T(n) = c_1 + c_2 \sum_{i=1}^{n} t_i + c_3 \sum_{i=1}^{n} (t_i - 1)$$

$$\text{Algorithm stops when } i = n^2 \rightarrow log_n(i) = 2$$

$$\sum_{i=1}^{n} t_i = \sum_{i=1}^{n} log_n(i + \frac{n}{2}) = \frac{log((\frac{n+2}{2})_n)}{log(n)}$$

💡 Little help from WolframAlpha for the equation above

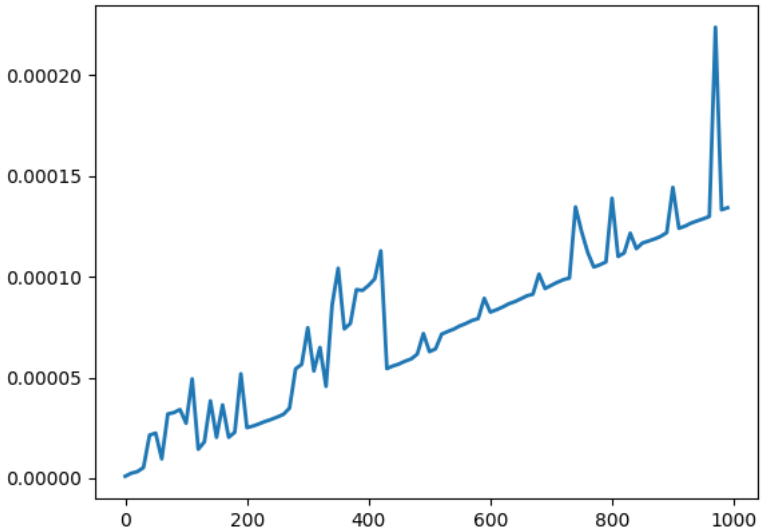
\sum_{i = 1}^{n}log_n(i + \dfrac{n}{2}) - Wolfram|Alpha

Uh oh! Wolfram|Alpha doesn't run without JavaScript. Please enable JavaScript. If you don't know how, you can find instructions . Once you've done that, refresh this page to start using Wolfram|Alpha.

https://www.wolframalpha.com/input/?i=%5Csum_%7Bi+%3D+1%7D%5E%7Bn%7Dlog_n%28i+%2B+%5Cdfrac%7Bn%7D%7B2%7D%29

WolframAlpha

$$T(n) = c_1 + c_2 \frac{log((\frac{n+2}{2})_n)}{log(n)} + c_3(\frac{log((\frac{n+2}{2})_n)}{log(n)} - n) = \Theta(log(n))$$




```
i = 1
while i < n:
    j = 1
    while j < n:
        j = j + 1
    i = 2 * i
```

```
i = 1
```

$$T_1 = c_1$$

```
while i < n:
```

$$T_2 = c_2 \times \sum_{i=1}^{n} t_i$$

```
    j = 1
```

$$T_3 = c_3 \times \sum_{i=1}^{n} (t_i - 1)$$

```
    while j < n:
```

$$T_4 = c_4 \times \sum_{i=1}^{n} (t_i - 1) \times \sum_{j=1}^{n} t_j$$

```
        j = j + 1
```

$$T_5 = c_5 \times \sum_{i=1}^{n}(t_i - 1) \times \sum_{j=1}^{n}(t_j - 1)$$

```
    i = 2 * i
```

$$T_6 = c_6 \times \sum_{i=1}^{n}(t_i - 1)$$

$$T(n) = c_1 + c_2 \sum_{i=1}^{n} t_i + c_3 \sum_{i=1}^{n}(t_i - 1) + c_4(\sum_{i=1}^{n}(t_i - 1)\sum_{j=1}^{n}(t_j)) + c_5(\sum_{i=1}^{n}(t_i - 1)\sum_{j=1}^{n}(t_j - 1)) + c_6 \sum_{i=1}^{n}(t_i - 1)$$

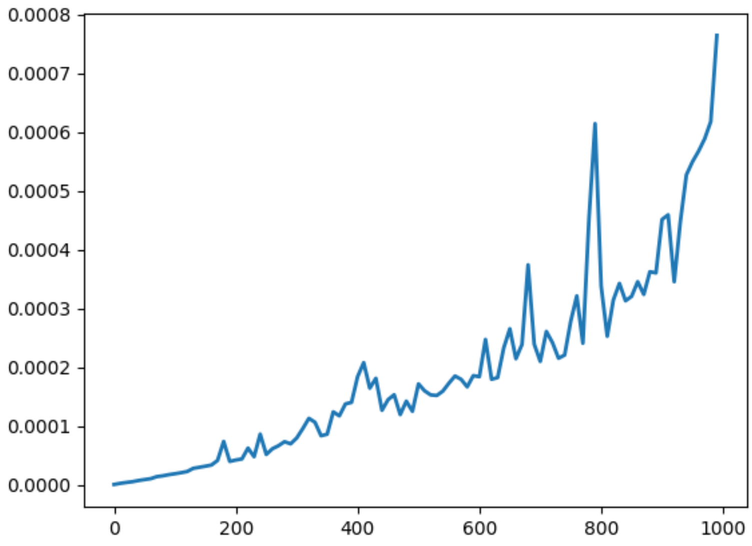
$$T_2 \text{ stops when } 2^{i-1} = n \to log_2(n) = i - 1$$

$$\sum_{i=1}^{n} t_i = log_2(n)$$

$$T_4 \text{ stops when } j = n$$

$$\sum_{j=1}^{n} t_j = n$$

$$T(n) = c_1 + c_2 log_2(n) + c_3 log_2((n-1)) + c_4 log_2((n-1))n + c_5 log_2((n-1))(n-1) + c_6 log_2((n-1)) = \Theta n(lo$$



```
for i in range(n):
    j = 1
    while j < 100:
        j = j + 1
```

```
for i in range(n):
```

$$T_1 = c_1 \times n$$

```
    j = 1
```

$$T_2 = c_2 \times (n - 1)$$

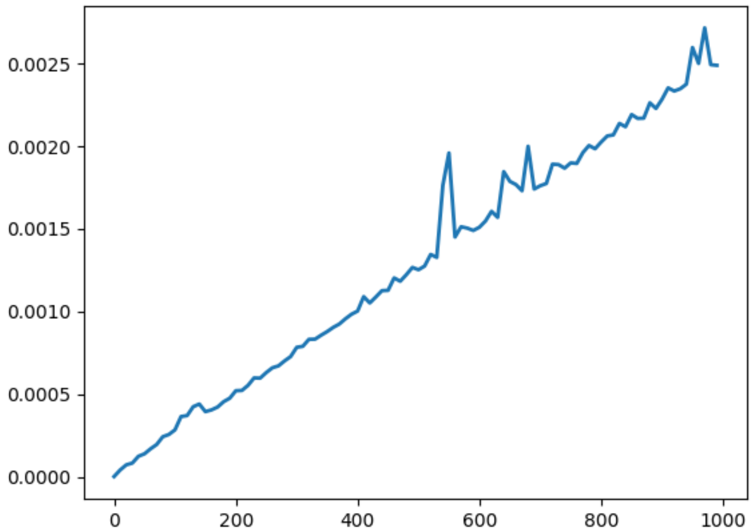
```
while j < 100:
```

$$T_3 = c_3 \times (n-1) \times 100$$

```
        j = j + 1
```

$$T_4 = c_4 \times (n-1) \times 99$$

$$T(n) = c_1 n + c_2(n-1) + c_3(100n - 100) + c_4(99n - 99) = \Theta(n)$$



Question 2

Analyze the time complexity of the following recursive function:

- Find the recurrence relation (T(n) = ?)
  - Including all the base cases
- Derive the big-Oh complexity (O(?))

```
def foo(n):
    if n < 5:
        total = n
    else:
        total = 0
        for i in [1,2,3]:
            total += foo(n//4 + i)
    while n > 0:
        total += n
        n = n // 2
    return total
```

```
def foo(n):
```

$$T_1 = c_1$$

```
    if n < 5:
```

$$T_2 = c_2$$

```
        total = n
```

$$T_3 = c_3$$

```
else:
```

$$T_4 = c_4$$

```
    total = 0
```

$$T_5 = c_5$$

```
    for i in [1,2,3]:
```

$$T_6 = c_6 \times 3$$

```
        total += foo(n//4 + i)
```

$$T_7 = c_7 \times 2 \times T(n)$$

```
while n > 0:
```

$$T_8 = c_8 \times \sum_{i=1}^{n} t_i$$

```
    total += n
```

$$T_9 = c_9 \times \sum_{i=1}^{n} (t_i - 1)$$

```
    n = n // 2
```

$$T_{10} = c_{10} \times \sum_{i=1}^{n} (t_i - 1)$$

```
return total
```

$$T_{11} = c_{11}$$

$$T_8 \text{ stops when } n = 2^{i-1} \rightarrow i - 1 = log_2(n)$$

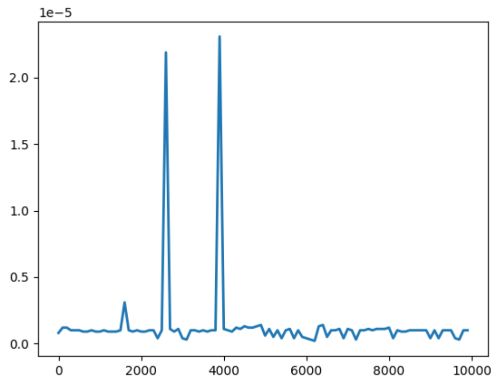
$$\sum_{i=1}^{n} t_i = log_2(n)$$

$$T(n) = c_1 + c_2 + c_3 + c_4 + c_5 + 3c_6 + 2c_7 T(n) + c_8 log_2(n) + c_9 log_2(n-1) + c_{10} log_2(n-1) + c_{11}$$

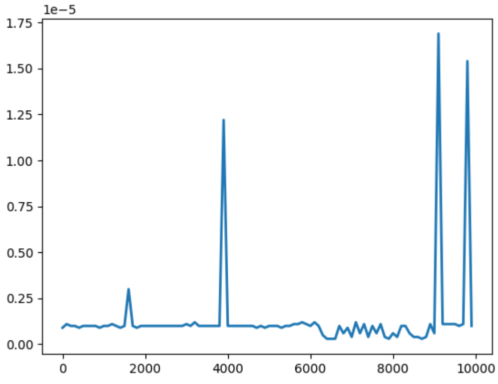
$$\text{Deducing constants: } T(n) = 2T(n) + log_2(n)$$

$$T(n) = \Theta(log(n))$$

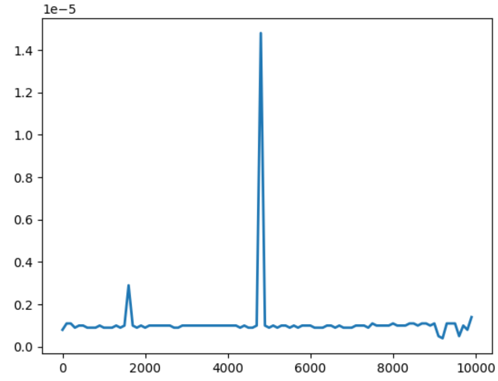
```
def foo(n):
    if n < 5:
      pass
```



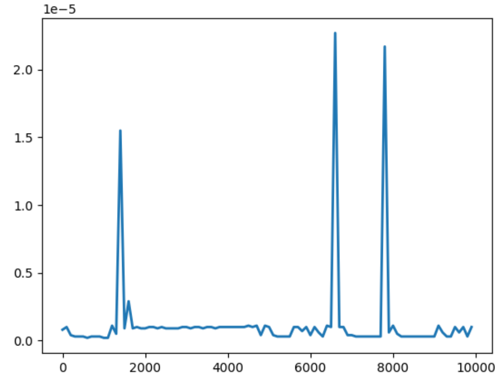
```
def foo(n):
    if n < 5:
      total = n
```



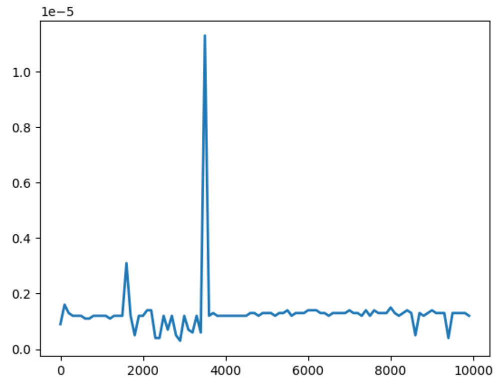

```
def foo(n):
    if n < 5:
        total = n
    else:
        pass
```



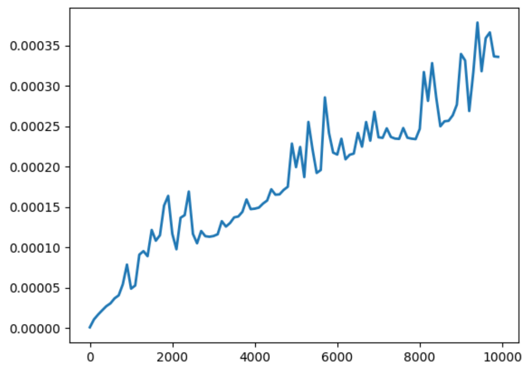

```
def foo(n):
    if n < 5:
        total = n
    else:
        total = 0
```



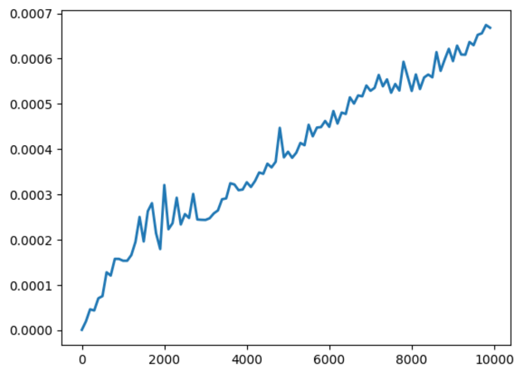

```
def foo(n):
    if n < 5:
        total = n
    else:
        total = 0
        for i in [1,2,3]:
          pass
```




```
def foo(n):
    if n < 5:
        total = n
    else:
        total = 0
        for i in [1,2,3]:
            total += foo(n//4 + i)
    return total
```




```
def foo(n):
    if n < 5:
        total = n
    else:
        total = 0
        for i in [1,2,3]:
            total += foo(n//4 + i)
    while n > 0:
        total += n
        n = n // 2
    return total
```



💡 Below is the code used for all visualizations

```
# Ozan Yetkin | 1908227
import matplotlib.pyplot as plt
from time import perf_counter

# Initialize the algorithms to calculate elapsed time
def question_1a(n):
    k = 0
    for i in range(n):
        k += 1
```

```
def question_1b(n):
    i = 1
    while i < n*n:
        i += 2

def question_1c(n):
    k = 1
    for i in range(n):
        for j in range(i+1,n):
            k += 1

def question_1d(n):
    i = 1
    while i < n*n*n:
        i = i + (2*n)

def question_1e(n):
    i = 1
    while i < n*n:
        i = i + (n // 2)

def question_1f(n):
    i = 1
    while i < n:
        j = 1
        while j < n:
            j = j + 1
        i = 2 * i

def question_1g(n):
    for i in range(n):
        j = 1
        while j < 100:
            j = j + 1

def foo(n):
    if n < 5:
        total = n
    else:
        total = 0
        for i in [1,2,3]:
            total += foo(n//4 + i)
    while n > 0:
        total += n
        n = n // 2
    return total

n_list = []
t_list = []
for n in range(0, 1000, 10):
    # Start the stopwatch / counter
    t1_start = perf_counter()

    # Call the function with n
    foo(n)

    # Stop the stopwatch / counter
    t1_stop = perf_counter()

    print("Elapsed time:", t1_stop - t1_start)
    n_list.append(n)
    t_list.append(t1_stop - t1_start)

# Plot
fig, ax = plt.subplots()
ax.plot(n_list, t_list, linewidth=2.0)

plt.show()
```