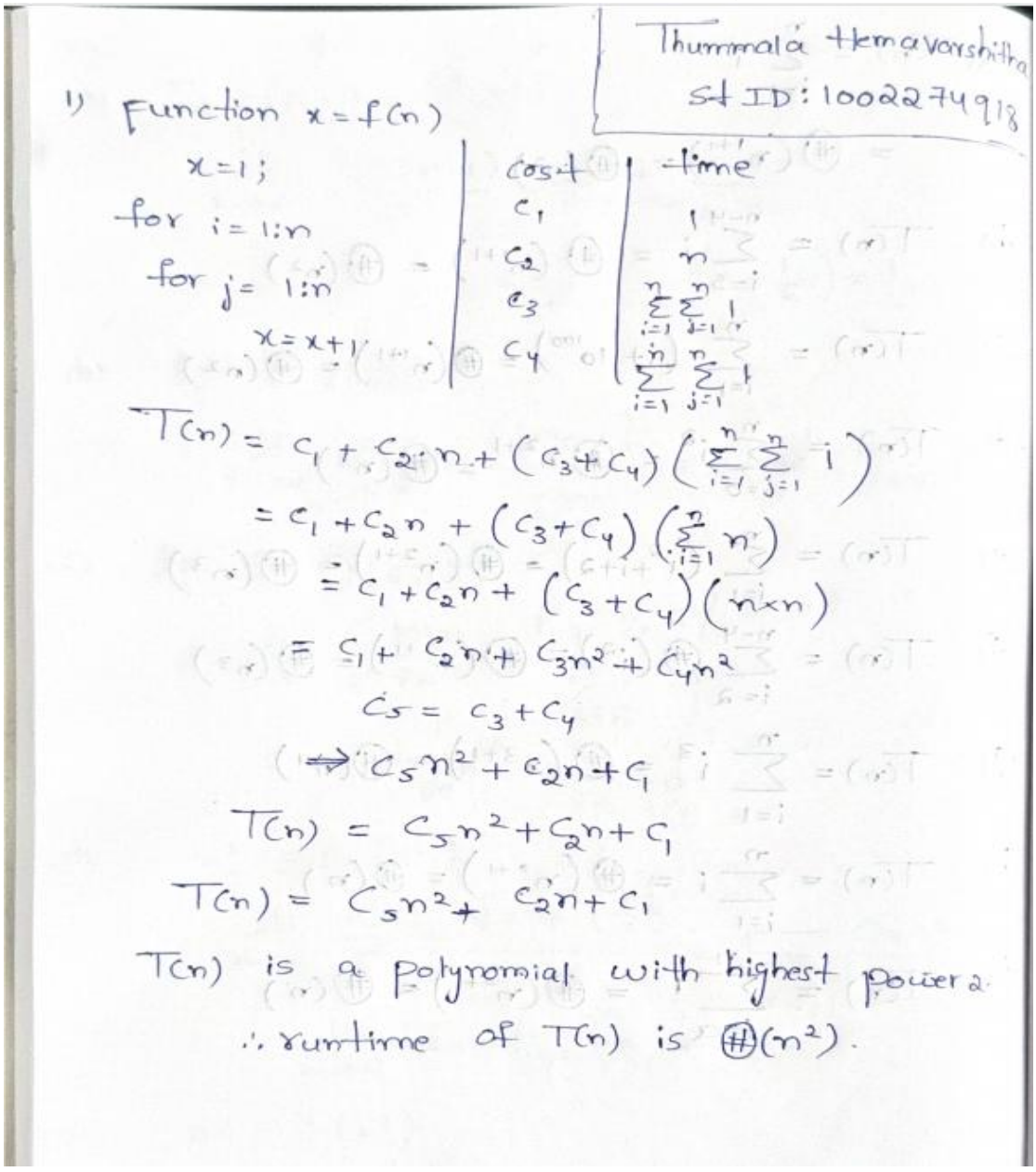
HANDS ON 3

St ID:1002274918
Hema varshitha Thummala

```
function x = f(n)
  x = 1;
  for i = 1:n
     for j = 1:n
        x = x + 1;
```

1.Find the runtime of the algorithm mathematically



2. Time this function for various n e.g. n = 1,2,3.... You should have small values of n all the way up to large values. Plot "time" vs "n" (time on y-axis and n on x-axis). Also, fit a curve to your data; hint it's a polynomial.

From 1,

We get $T(n) = C_5 n^2 + C_2 n + C_1$
assume $C_5 = 1$, $C_2 = 3$, $C_1 = 1$

$$\Rightarrow T(n) = n^2+3n+1$$

```
import time
import numpy as np
import matplotlib.pyplot as plt

def original_function(n):
    x = 1
    for i in range(1, n + 1):
        for j in range(1, n + 1):
            x += 1
```

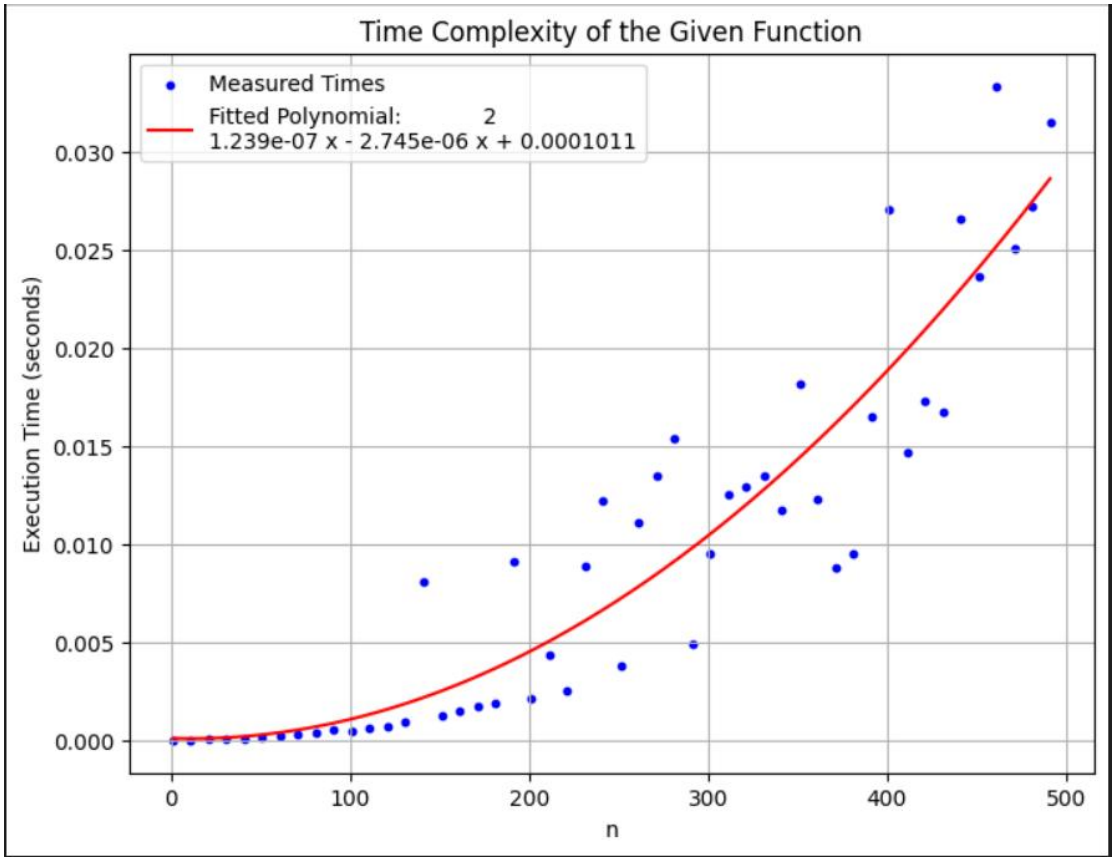
```
n_values = np.arange(1, 501, 10)
times = []
```

```
for n in n_values:
    start_time = time.time()
    original_function(n)
    end_time = time.time()
    times.append(end_time - start_time)
```

```
coefficients = np.polyfit(n_values, times, 2)
polynomial = np.poly1d(coefficients)
```

```
smooth_n = np.linspace(min(n_values), max(n_values), 500)
smooth_times = polynomial(smooth_n)
```

```
plt.figure(figsize=(8, 6))
plt.scatter(n_values, times, label="Measured Times", color="blue", marker="o", s=10)
plt.plot(smooth_n, smooth_times, label=f"Fitted Polynomial: {polynomial}", color="red")
plt.xlabel("n")
plt.ylabel("Execution Time (seconds)")
plt.title("Time Complexity of the Given Function")
plt.legend()
plt.grid(True)
plt.show()
```



3. Find polynomials that are upper and lower bounds on your curve from #2. From this specify a big-O, a big-Omega,and what big-theta is.

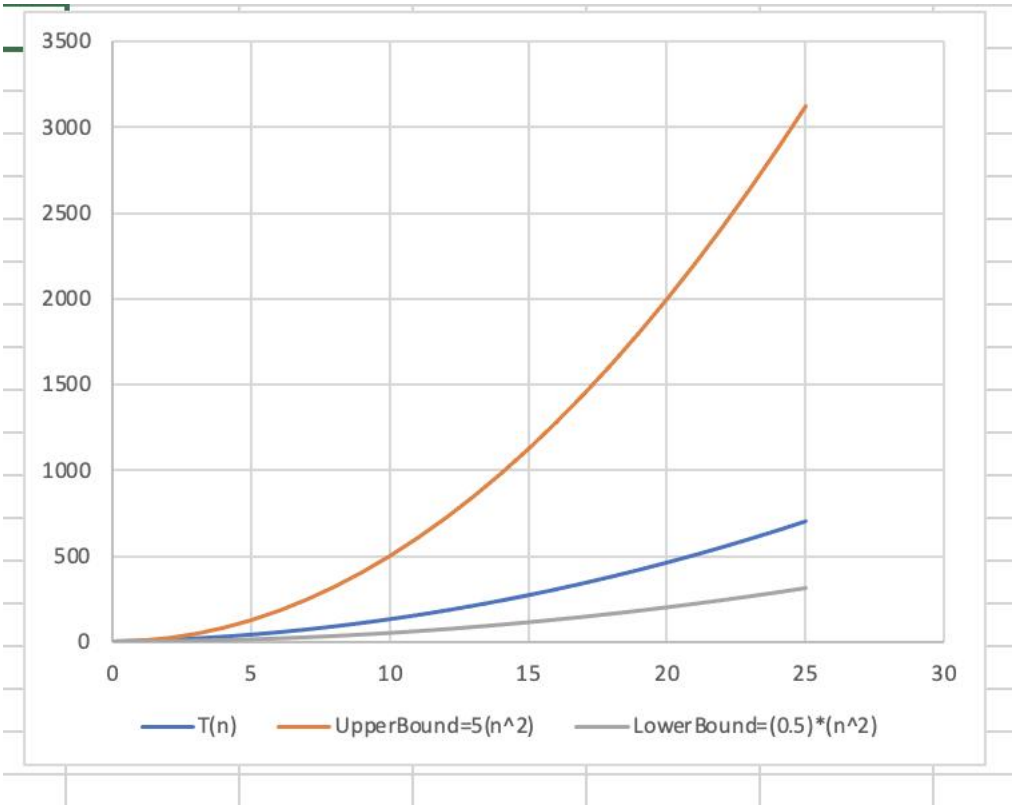
Let us consider that,

$$\Rightarrow T(n) = C_5\, n^2 + C_2\, n + C_1$$

Upper bound: $Cg(n) = 5\, n^2$ big - O = $O(n^2)$

Consider, the lower bound as $Cg(n) = 0.5n^2$

Big Omega is Ω(n2) Below is the graph for T(n), upper bound and lower bound



Since the upper bound is $5n^2$ and the lower bound is $0.5n^2$, and based on the definition of Big-Theta $\Theta(g(n))$, which states that there exist positive constants c_1, c_2, and n_0 such that $0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)$ for all $n \geq n_0$, we can conclude that the asymptotic complexity is $\Theta(n^2)$.

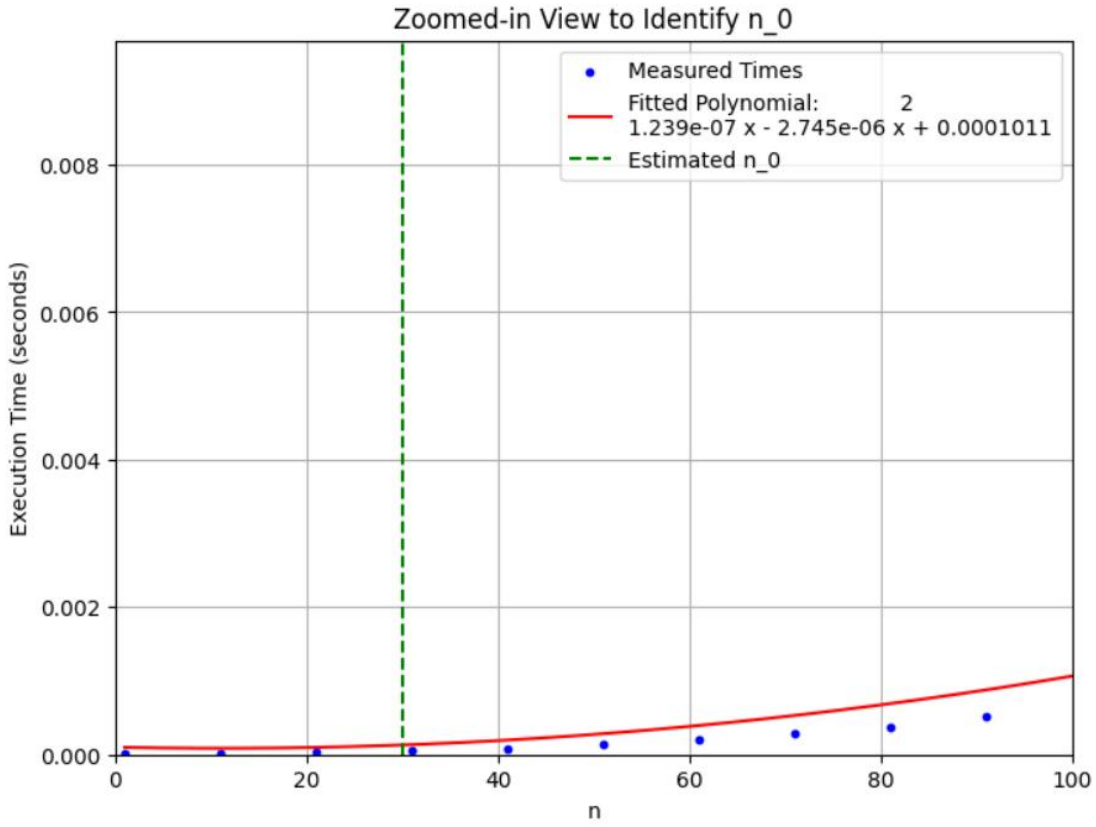
4. Find the approximate (eye ball it) location of "n_0" . Do this by zooming in on your plot and indicating on the plot where n_0 is and why you picked this value. Hint: I should see data that does not follow the trend of the polynomial you determined in #2.

From this we can say that "n_0" is 1.

Because for all values of n which are greater than or equal to 1. For all n>1 we get $5n^2 <= T(n) <= 0.5n^2$

```
plt.figure(figsize=(8, 6))
plt.scatter(n_values, times, label="Measured Times", color="blue", marker="o", s=10)
plt.plot(smooth_n, smooth_times, label=f"Fitted Polynomial: {polynomial}", color="red")
plt.xlabel("n")
plt.ylabel("Execution Time (seconds)")
plt.title("Zoomed-in View to Identify n_0")
plt.xlim(0, 100)
plt.ylim(0, max(times[:15]) * 1.2)
```

```
plt.axvline(x=30, color='green', linestyle='--', label="Estimated n_0")
plt.legend()
plt.grid(True)
plt.show()
```



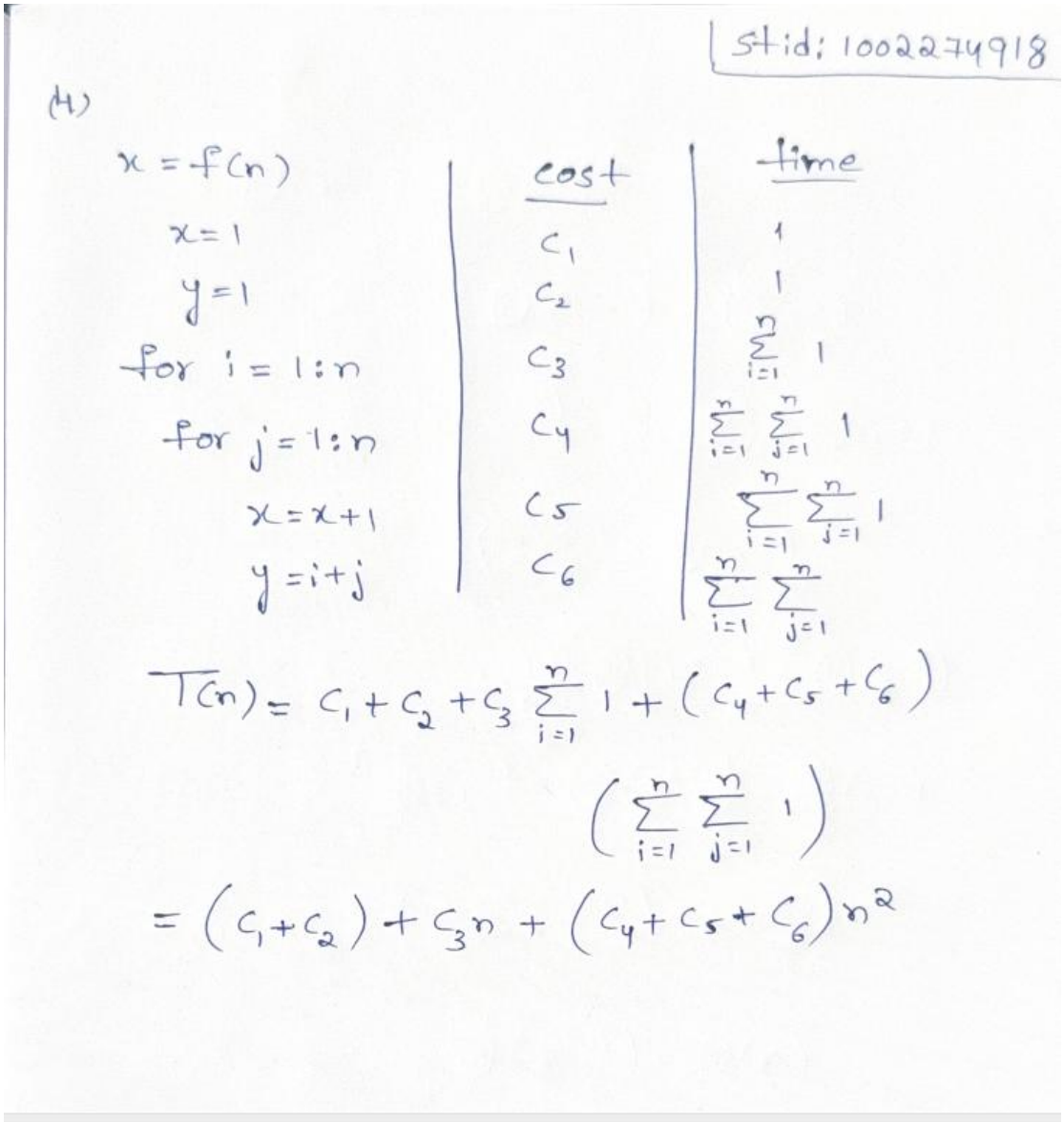
If I modified the function to be:

```
x = f(n)
  x = 1;
  y = 1;
  for i = 1:n
      for j = 1:n
          x = x + 1;
          y = i + j;
```

4.Will this in-create how long it takes the algorithm to run (e.x. you are timing the function like in #2)?

Yes, the modification adds an extra operation within the nested loop, leading to an increase in the overall execution time.It will take longer to

run due to the extra computation, but the Big-O complexity remains the same at $O(n^2)$.



5. Will it effect your results from #1?

No, the modification does not alter the overall asymptotic complexity; it remains $O(n^2)$. The number of steps in the function has increased, but the only difference is in the constant factor. The new function takes slightly longer to execute due to this increase in the constant value, while the fundamental structure of T(n) remains unchanged.

6.Implement merge sort, upload your code to github and show/test it on the array [5,2,4,7,1,3,2,6].

Github link: https://github.com/VARSHI26/UTA-DAA-CSE-5311-006-/blob/main/Merge%20sort