⇨ 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.

⇨ Answer:

Step 1: Understanding n_0 in Asymptotic Notation

n0 is the threshold where our fitted polynomial $T(n) \approx an^2 + bn + c$ starts accurately representing the algorithm's runtime. Before n0, lower-order effects and system noise (e.g., CPU caching, OS scheduling) might cause deviations from the expected n^2 growth.

Step 2: Identifying n0 Using the Plot

- **Zooming into the lower n values**: At small n, the measured execution time might **not** follow the expected n^2n2curve because:
    - The function executes too quickly to measure accurately.
    - Overhead (like loop setup) dominates.
- **Looking for deviations**: n0 is where the measured data **starts aligning** with the fitted quadratic trend.

Step 3: Finding n0 on the Plot

- In the **MATLAB or Python plot**, zoom into small values of nn (e.g., n=1 to n=20).
- Identify the **first n where the measured time consistently follows $\boldsymbol{\Theta}(\boldsymbol{n^2})$**
- **Mark n0n0 visually**: Use a **vertical line** to indicate where the behavior shifts.

Step 4: Updating the Code to Visualize `n0`

```
plt.axvline(x=10, color='green', linestyle='--', label='n_0')  # Adjust based on observations

plt.legend()
```

Step 5: Conclusion

- n0 is found by inspecting where the measured data **begins to align** with the quadratic trend.

- Before n0, runtime fluctuations occur due to **hardware and system noise**.
- After n0, the algorithm follows $\Theta(n^2)$) growth.

Final Answer: n0 is typically **around 10-20**, but you should **eyeball it** by zooming into the plot and marking the transition point.