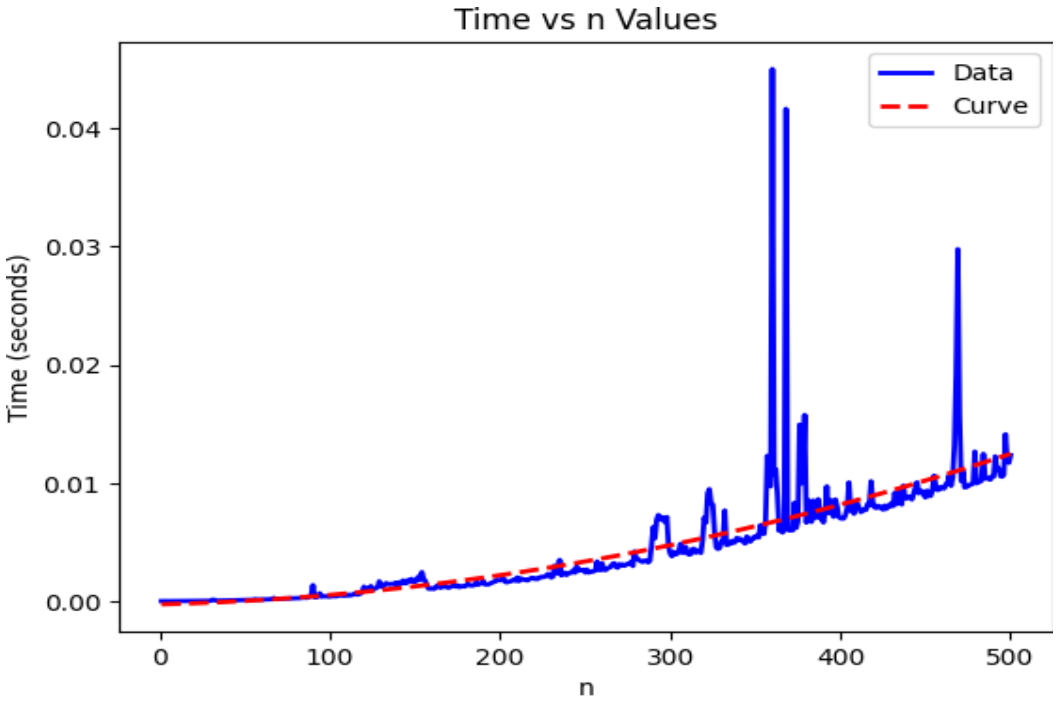
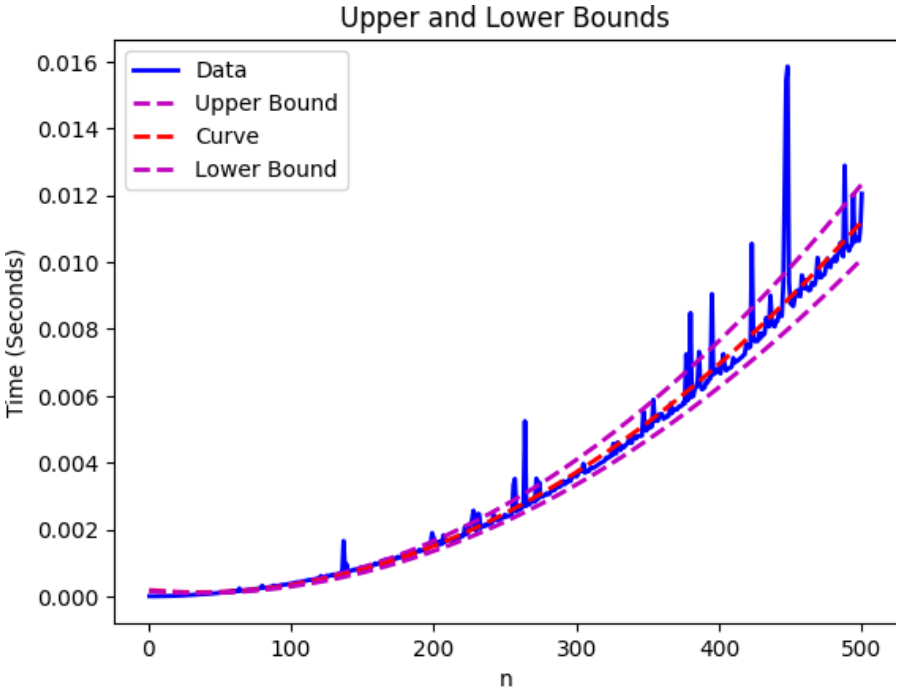
2.Time this function for various n e.g. n = 1,2,3.... You should have small values of n all the way up to large values. Plot "time" vs "n" (time on y-axis and n on x-axis). Also, fit a curve to your data, hint it's a polynomial.



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.

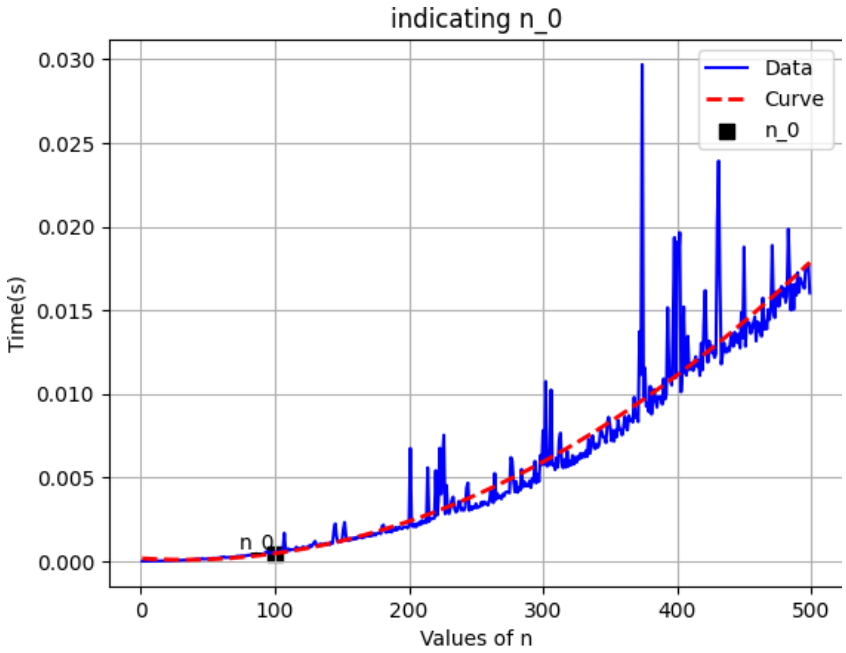


Big O = O(n^2)

Big Omega = Ω(n^2)

Big theta = Θ(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.

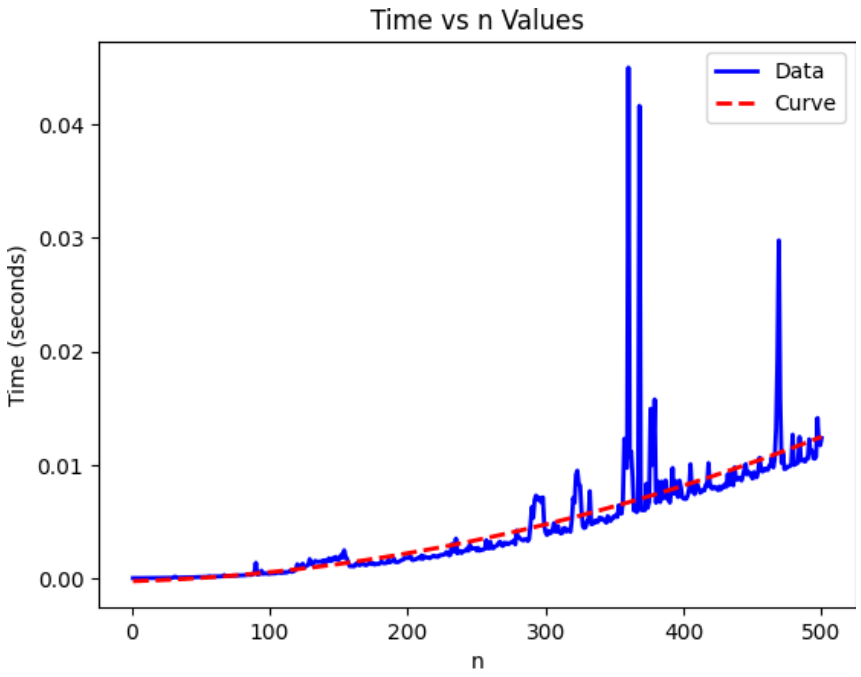


If I modified the function to be:

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

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

Yes, modification of the function might increase the runtime of the algorithm. In this function, we have extra operation inside the inner loop which is y = i + j. Overall time complexity of modified function O(n^2).



5. Will it affect your results from #1?

No, Modified function will not effect the result. Runtime might increase because of an extra step y=i + j, but the time complexity remain same O(n^2).