641

It shows how an algorithm scales.

O(n^2): known as **Quadratic complexity**

- 1 item: 1 second
- 10 items: 100 seconds
- 100 items: 10000 seconds

Notice that the number of items increases by a factor of 10, but the time increases by a factor of 10^2. Basically, n=10 and so O(n^2) gives us the scaling factor n^2 which is 10^2.

O(n): known as **Linear complexity**

- 1 item: 1 second
- 10 items: 10 seconds
- 100 items: 100 seconds

This time the number of items increases by a factor of 10, and so does the time. n=10 and so O(n)'s scaling factor is 10.

O(1): known as **Constant complexity**

- 1 item: 1 second
- 10 items: 1 second
- 100 items: 1 second

The number of items is still increasing by a factor of 10, but the scaling factor of O(1) is always 1.

O(log n): known as **Logarithmic complexity**

- 1 item: 1 second
- 10 items: 2 seconds
- 100 items: 3 seconds
- 1000 items: 4 seconds
- 10000 items: 5 seconds

The number of computations is only increased by a log of the input value. So in this case, assuming each computation takes 1 second, the log of the input `n` is the time required, hence `log n`.

That's the gist of it. They reduce the maths down so it might not be exactly n^2 or whatever they say it is, but that'll be the dominating factor in the scaling.