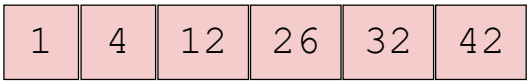
Logarithmic Complexity - O(log(n))

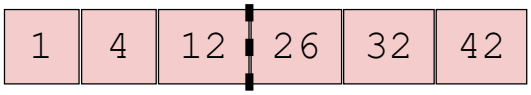
The size of the input gets split into half with each iteration of the function. (Reading time: under 2 minutes)

If an algorithm has logarithmic time complexity, it means that the size of the input we are considering gets split into half with each iteration. Let's say that we have a function that takes 1 second to execute if the input size is 100. With a logarithmic runtime, it would then take 2 seconds if the input size is 1000, and 3 seconds if the input size is 10,000. The bigger the input size gets, the smaller the difference in runtime!

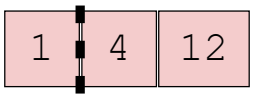
Consider the following array:



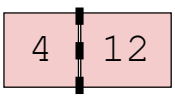
Let's say that we're looking for the value of 7 in this sorted array. With a specific searching algorithm, we split the input array on every round, and only check the side of the half (displayed as the dotted line) where the value could potentially be.



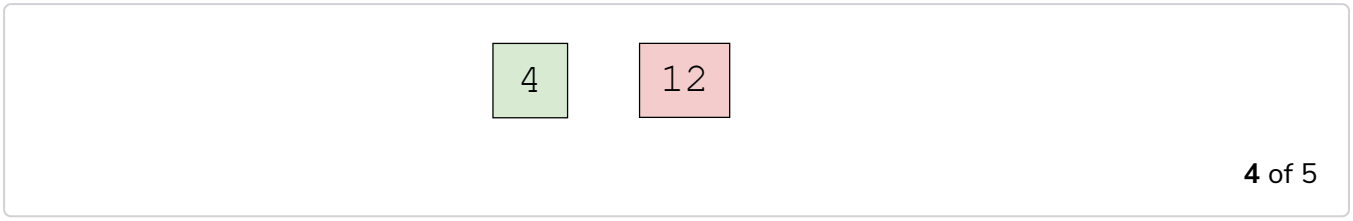
1 of 5

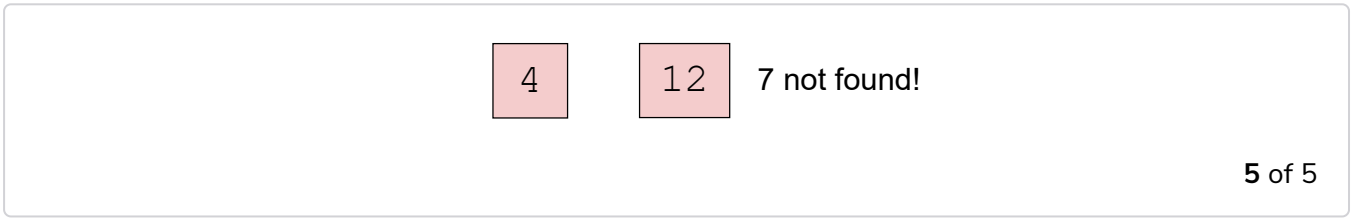


2 of 5



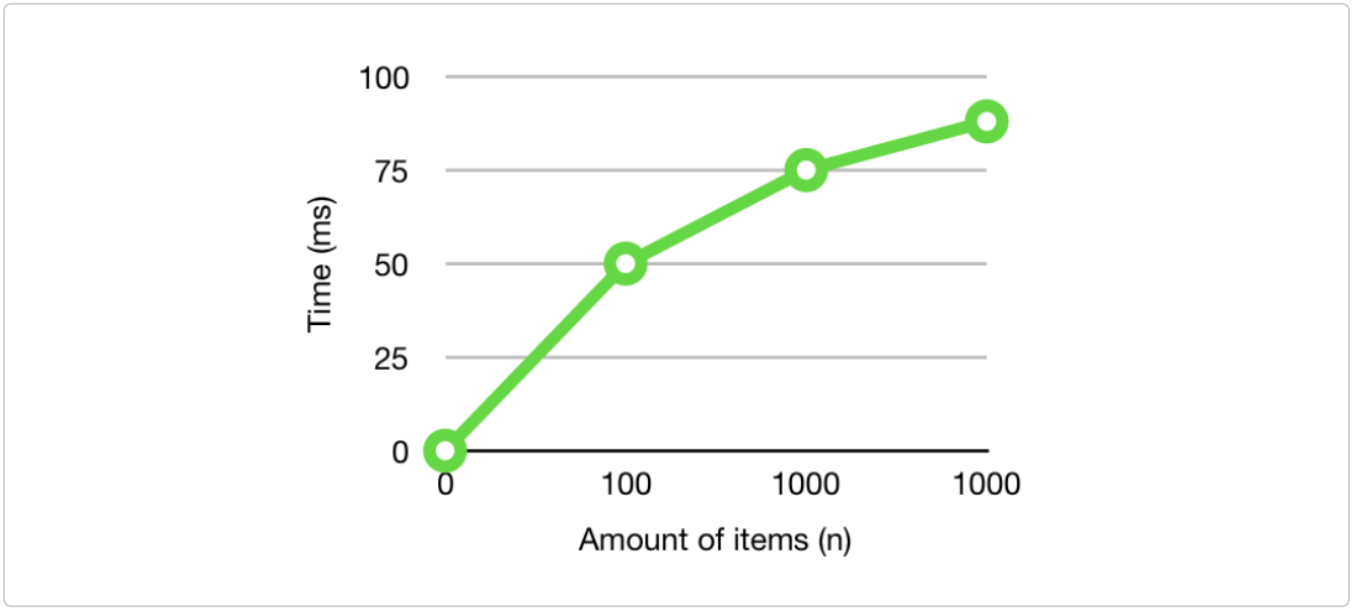
3 of 5





After every execution, the size of the array gets split (approximately) in half!

In a graph, it would look like this:



In the next lesson, I will discuss algorithms with exponential time complexity.