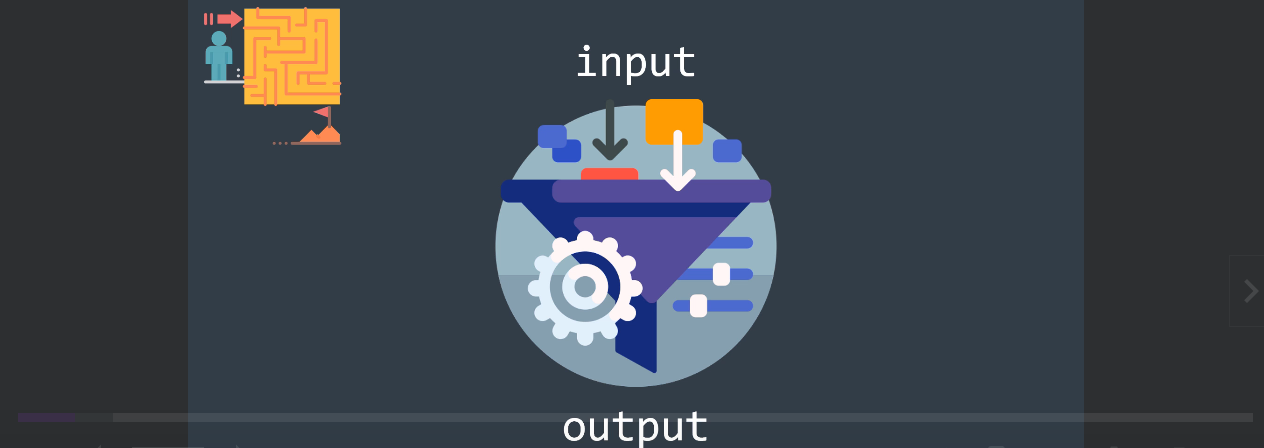
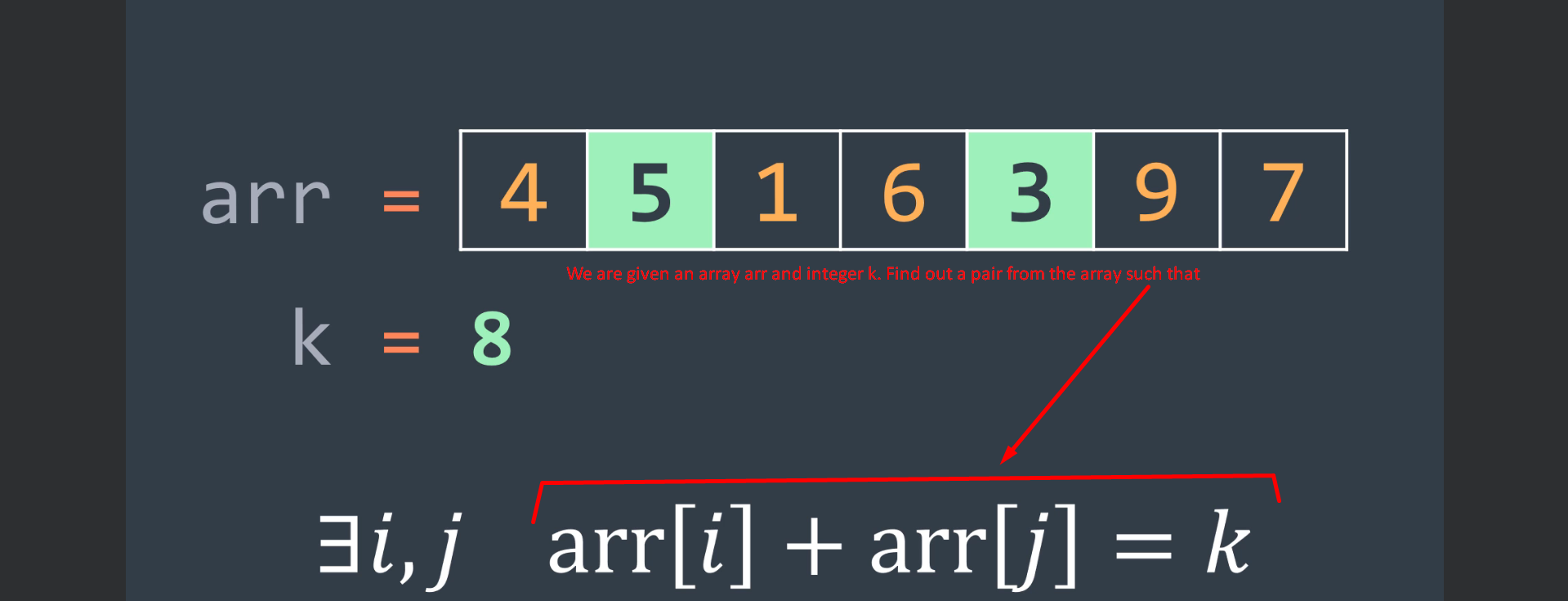
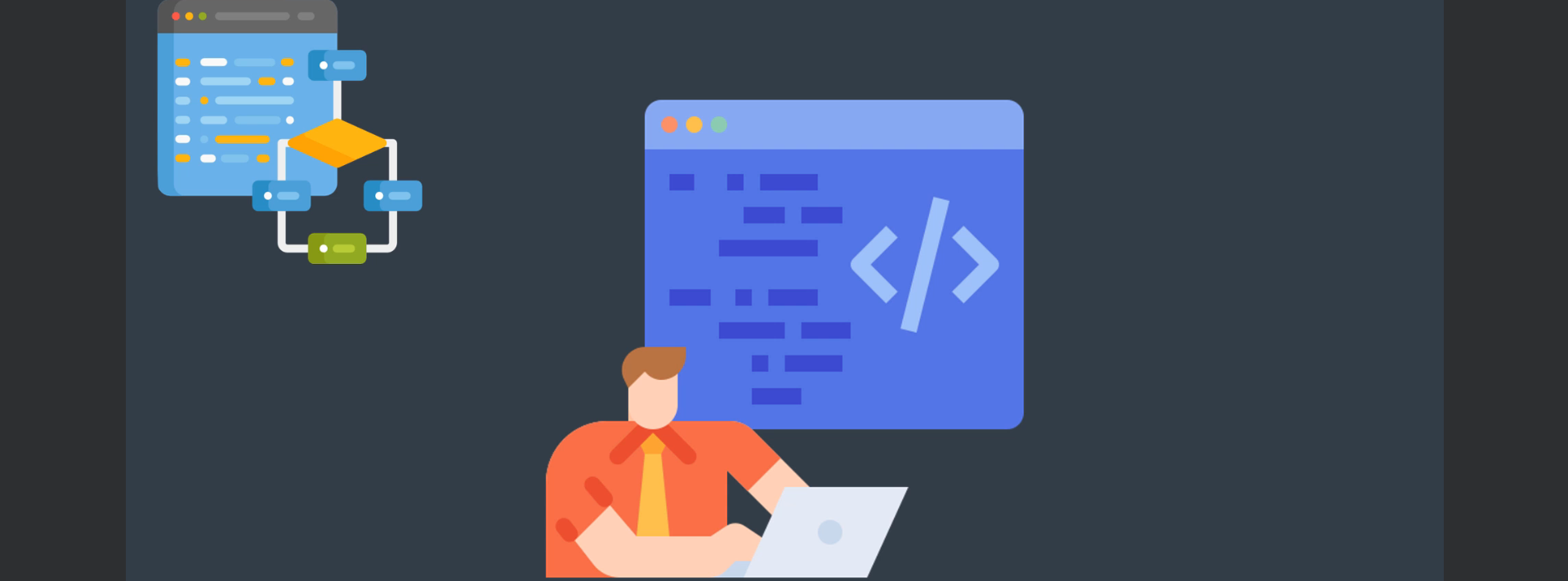
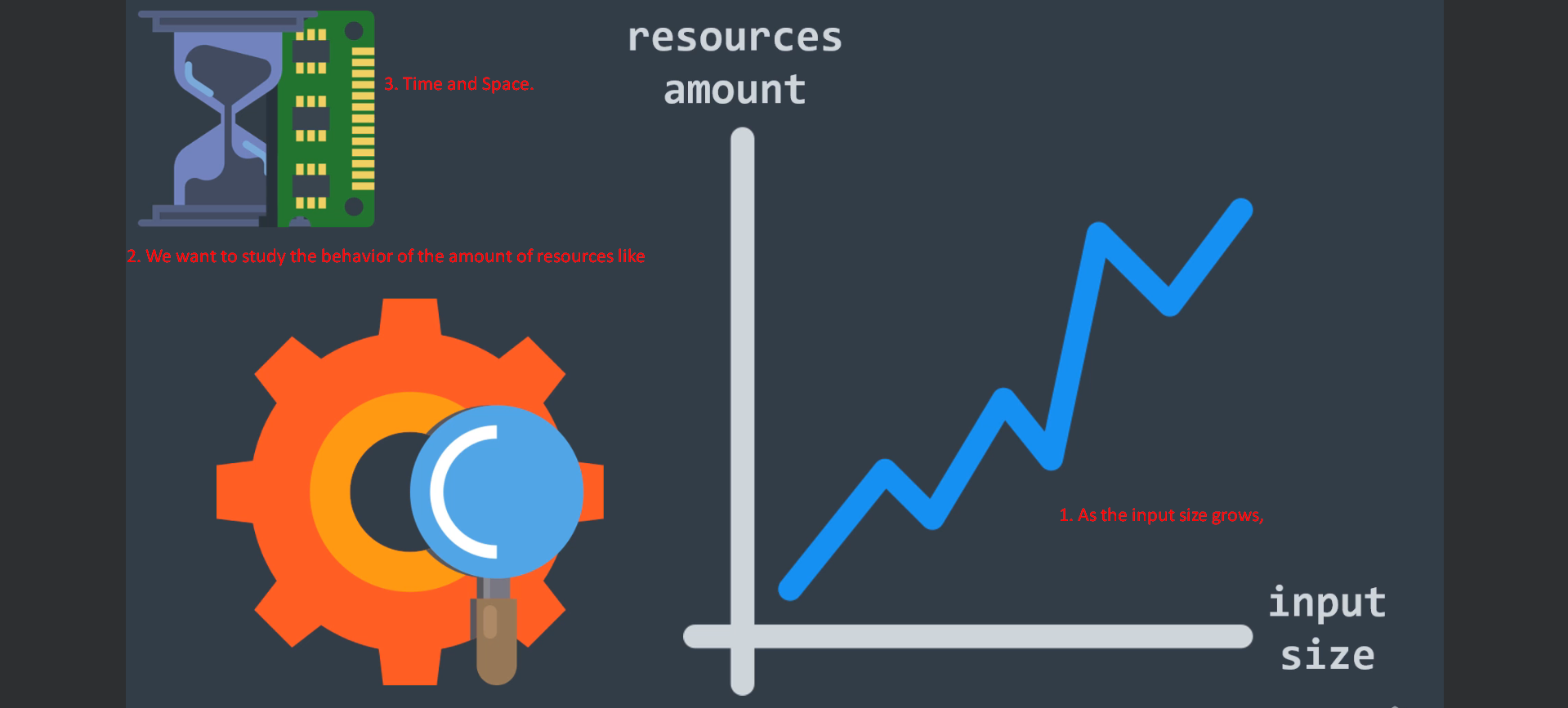
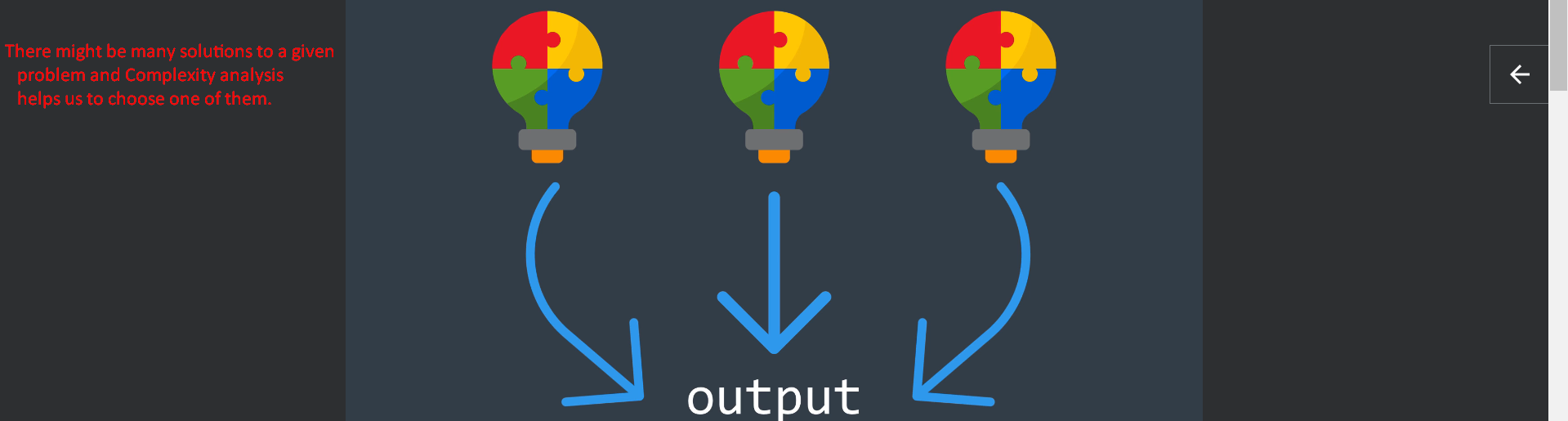
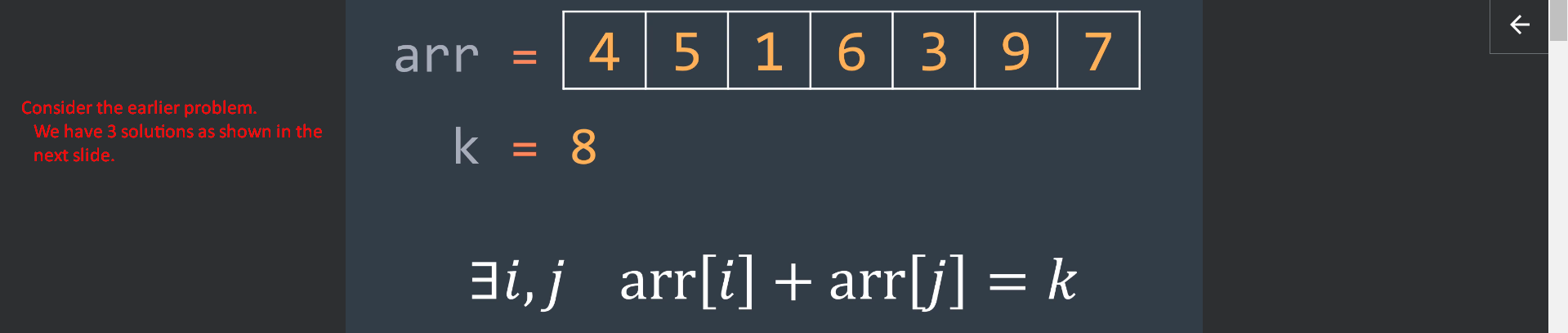
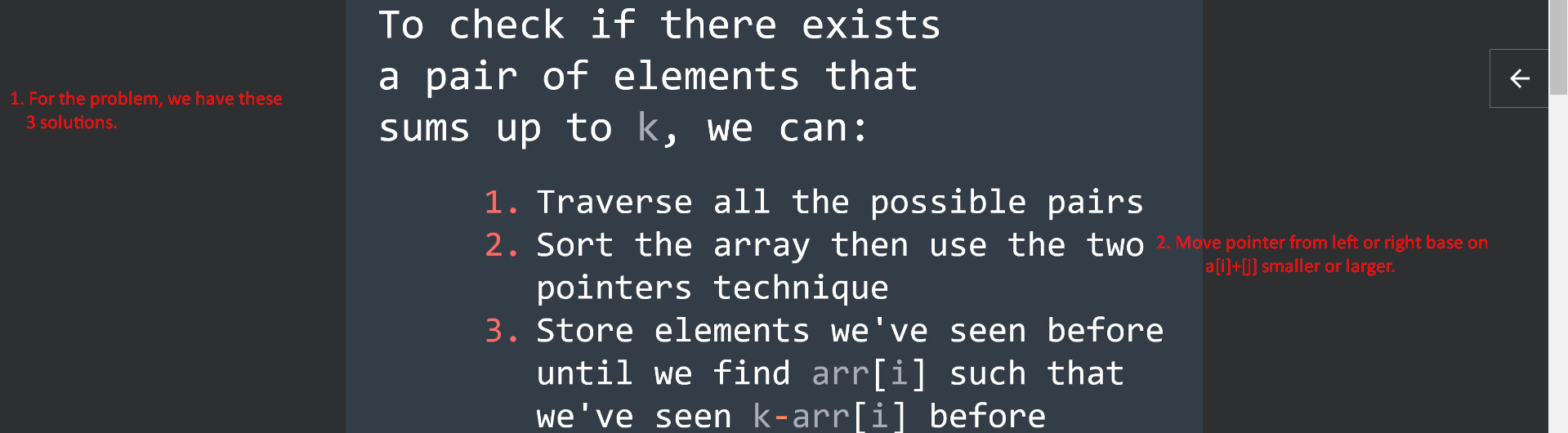
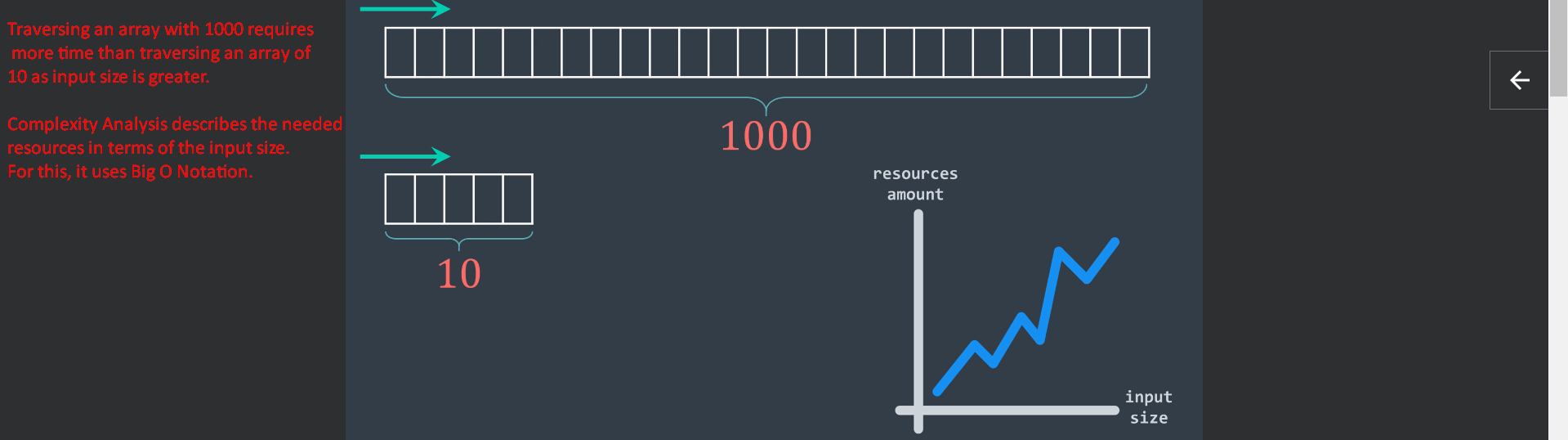
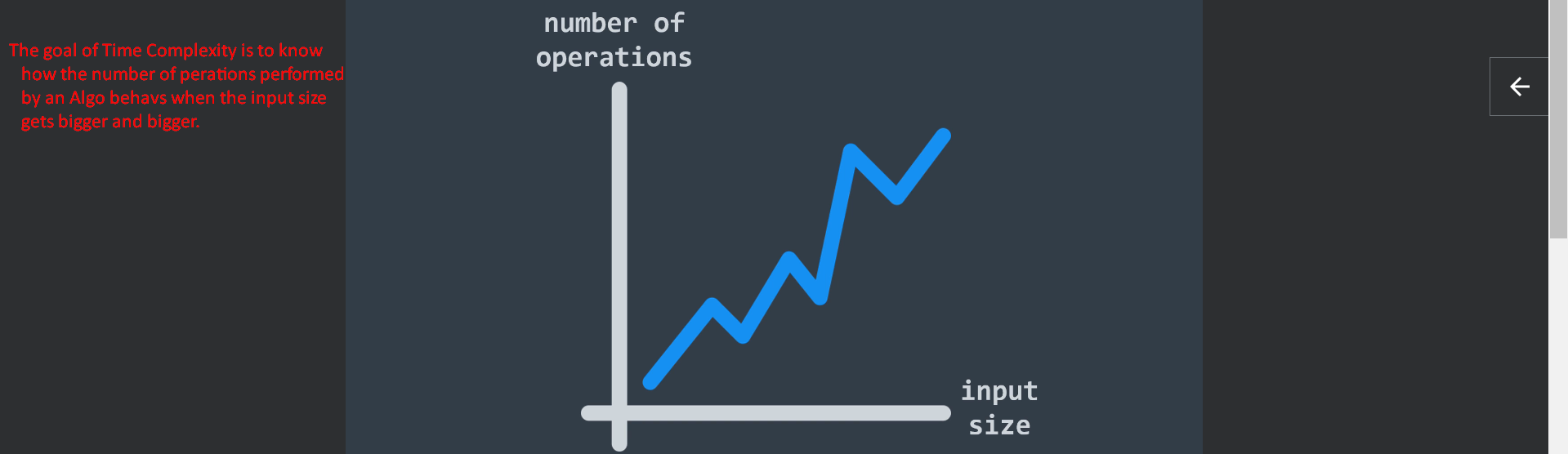
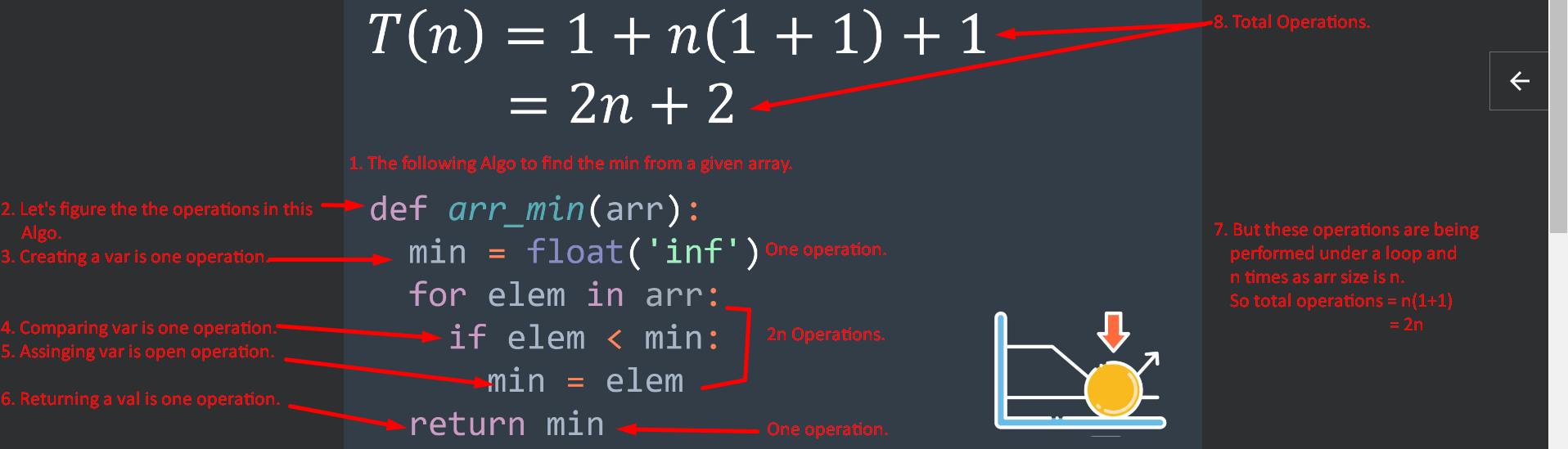
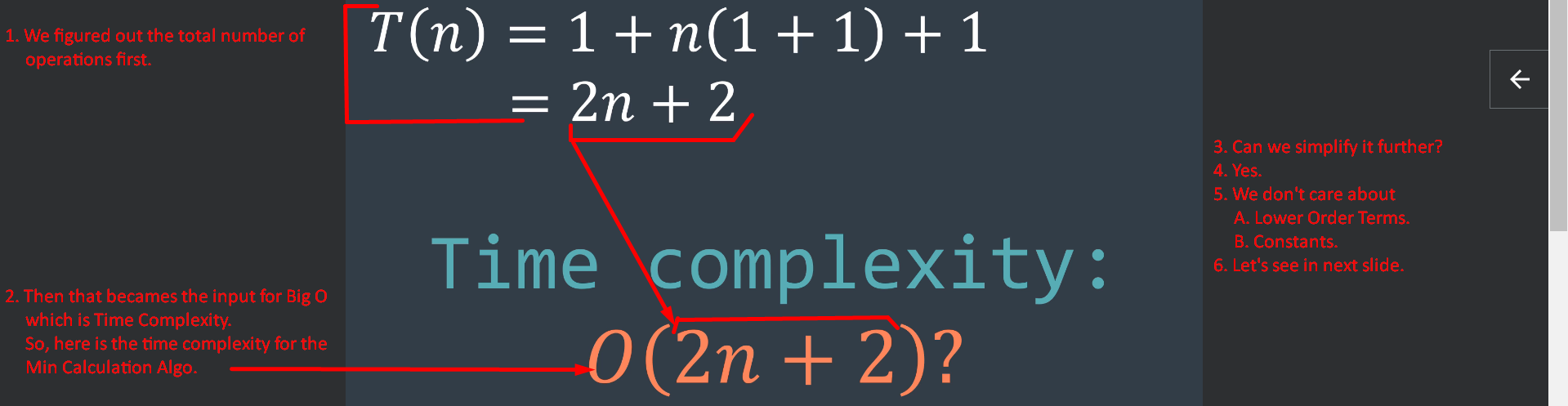
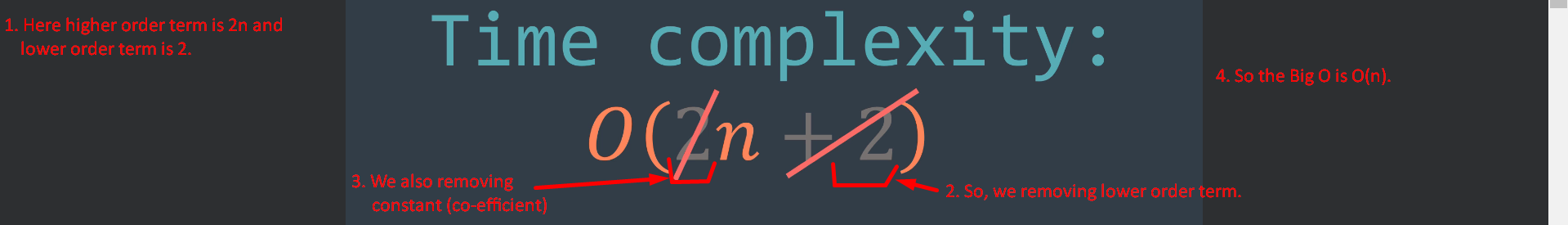
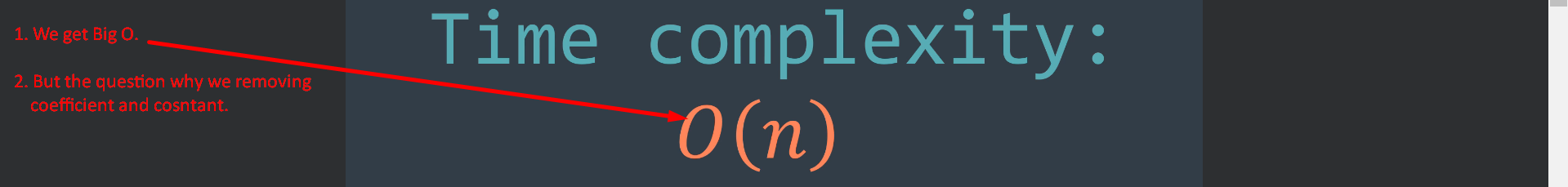
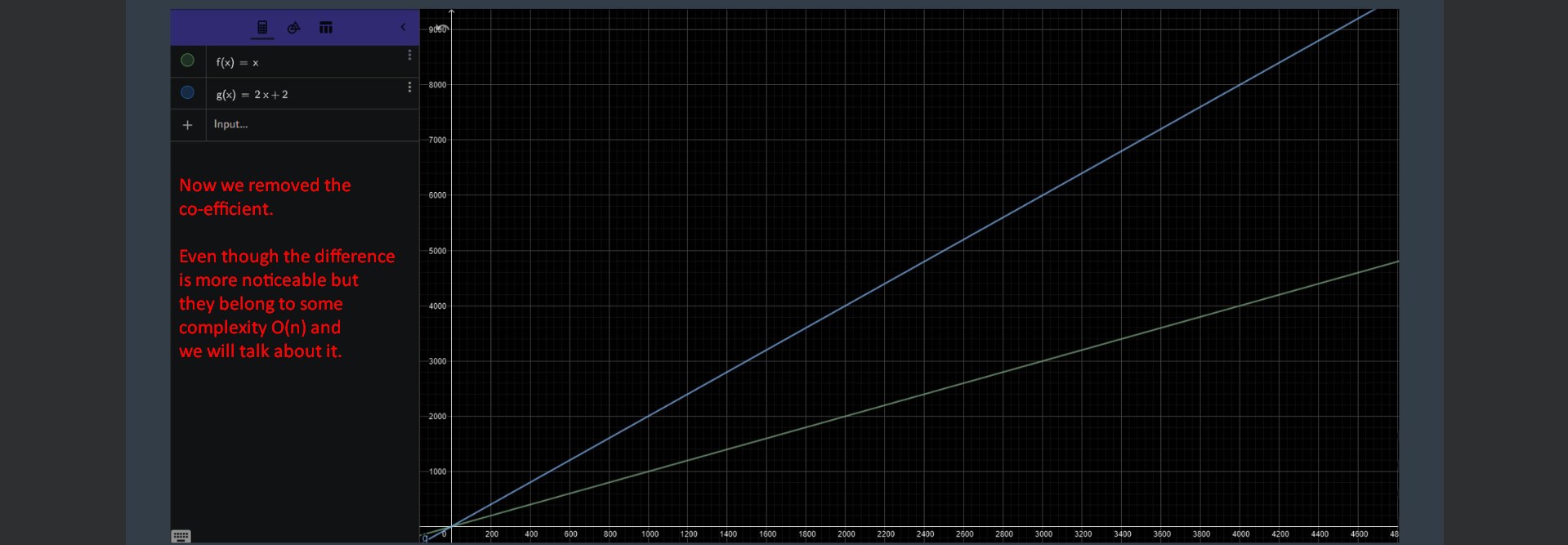
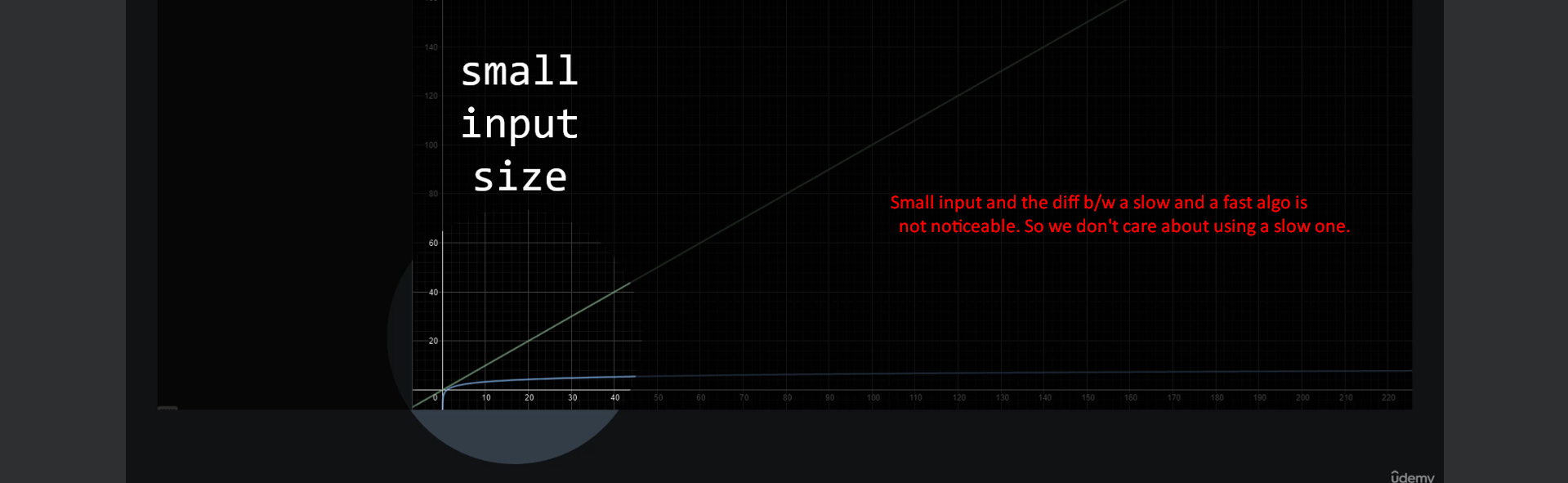
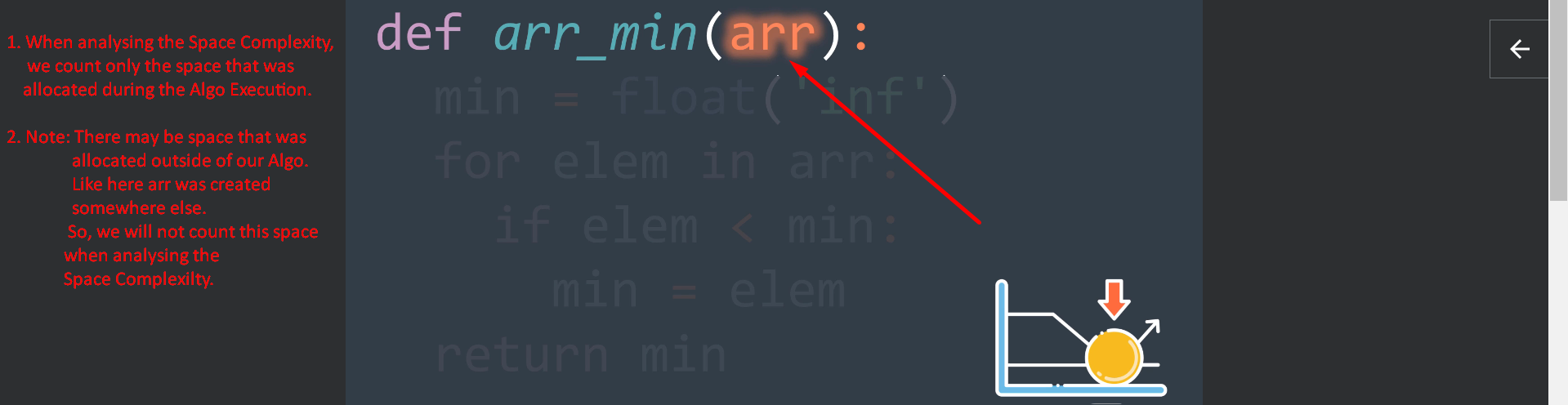
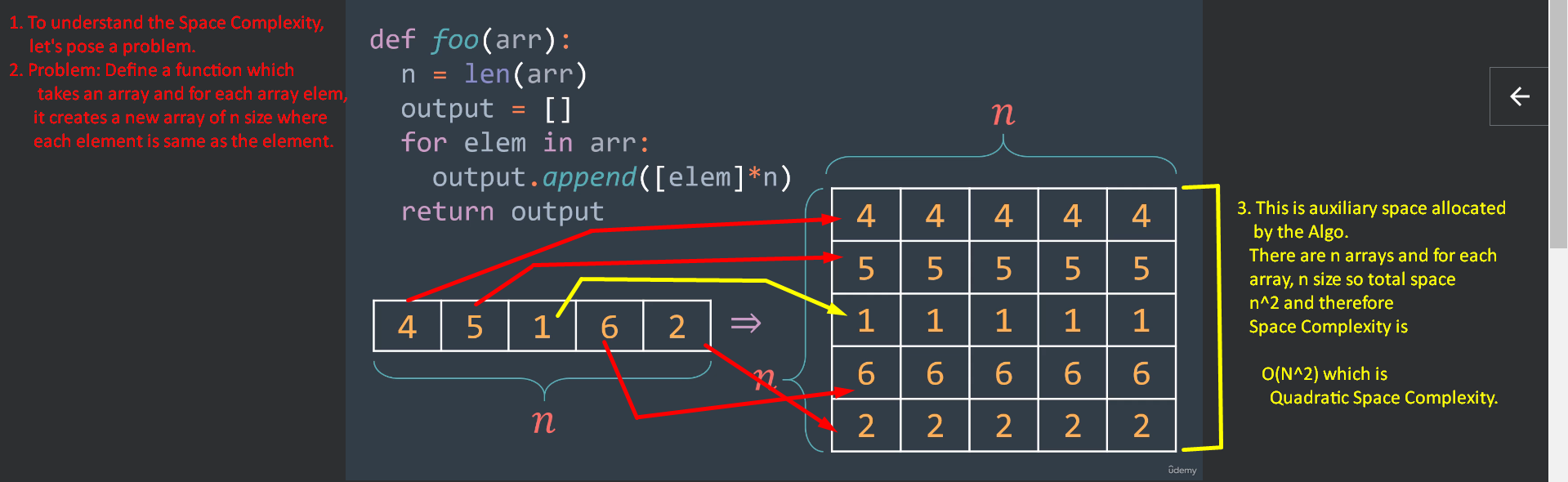
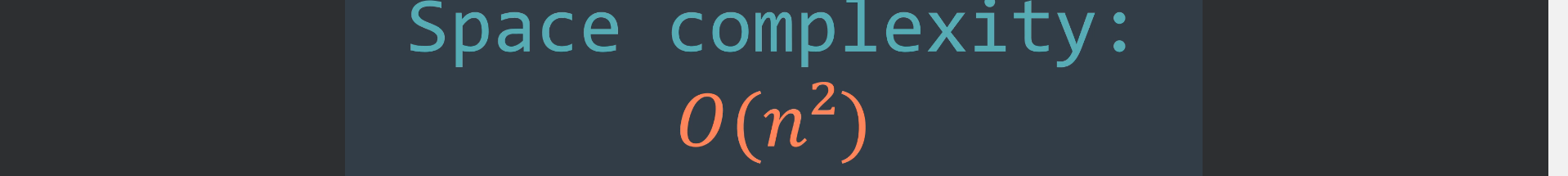
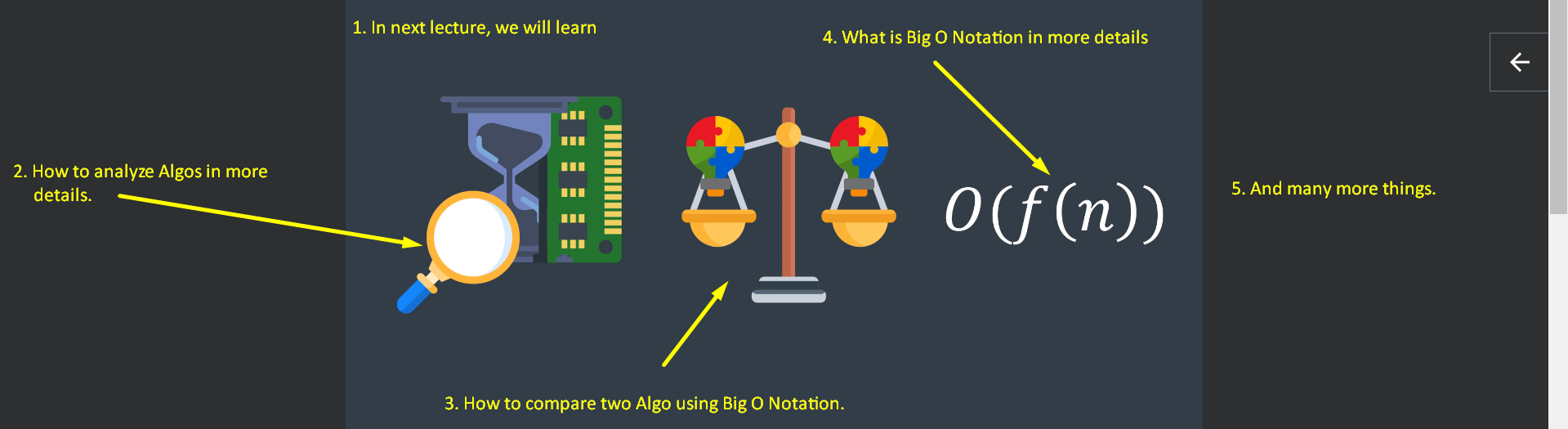
1. 
2. First, we need to understand what does a problem mean??
3. A problem in programming means that we are given some data, called the input, and our goal is to find a way to use that input to produce the output ( for example a specific information we are searching for).  
   
4. Let’s try to understand by an example:  
     
   The output is Boolean.

Now we know what is input and what is the output (Boolean).   
Now we have to find out a way to find an output starting from the input and for that we use **Algorithm**.

1. **Algorithm:** A finite set of instructions that in our Context, represents a solution to a problem.  
   It will tell computer who to do step by step to produce that output.  
   As Problem Solvers, it is our job to write that Algorithm which requires resources and we care about two resources **How Much Time** and **Space**.  
     
   With Time and Space Complexity analysis, we won’t exactly find how many seconds or bytes an algorithm needs because it is impossible but we will study the behavior of the amount of resources (Time and Space) as the input size grows.  
   
2.   
     
     
     
   
3. We can choose a solution among many based on some criteria like
   1. Code Shortness.
   2. Understandability.
   3. Readability.
4. But the most important one is usually the amount of resource needed in terms of time and space, which we can analyze by using complexity analysis.
5. So, complexity analysis will not only help us to understand the behavior of our algorithm but it will also give us the ability of comparing it with other algorithms in order to find the most efficient one.
6.   
   Till now 3:50 Done.
7. We know that Algo requires resources like Time & Space which is directly related to input size.
8. Traversing an array of 1000 elements requires more time than traversing an array of 10 elements, because the size is greater.  
   And the **Complexity Analysis** is here to describe the needed resources in terms of the input size and for that it uses the **Big O Notation**.  
     
   Bit O Notation 🡺 O(f(n)) where **f(n) represents the upper bound of the amount of resources**, we will talk.
9. Let’s talk about Time Complexity.
10. Algo performs different operations such as
    1. Comparing two number.
    2. Creating a variable etc.
11. The goal of Time Complexity is to know how does the number of operations done by an algo behaves as the input size gets bigger and bigger.  
    
12. Let’s try to understand this by an example.  
    
13. 
14.   
    
15. Why do we not care about Co-efficient and Constant?
16. **Answer**: Because in Complexity Analysis, we are interested in knowing the behavior of the Algo when input size is very big.  
      
      
    
17. **So, the Time Complexity of this Algo is O(n) which is called Linear Time Complexity which means the number of operations follows the input size linearly.**
18. Why do we not count the exact time that an Algo takes? Like this also will take x seconds.
19. It is impossible as execution time depends on so many factors like
    1. CPU Speeds. We have CPUs faster than other ones.
    2. Programming Language.
20. 2ndly, execution time is impacted by input size. So, it is more interesting, how will input size affect the number of operations rather than trying to search for the exact execution time. Jatin: So here we are interested only in the way the algo will behave when input starts growing rather than how much exact time it will take.
21. **2nd Question**: Why does the Complexity become important when the input size is really big?
22. **Answer**: Because the difference b/w a slow Algo and a fast Algo is very and very small and we can say non-existent sometimes. So, we do not care about using a slow Algo but when the input size starts getting bigger, the difference is more and more important. So, using a fast Algo becomes necessary.  
      
      
    

Let’s talk about Space Complexity

1. Space Complexity follows the same logic but instead of caring about the number of operations, it cares about the auxiliary Space Size that is needed by an Algo.  
   Auxiliary Space means the space that is allocated during the execution of an Algo. Because we can have space that comes from outside,   
   For example, in our min function, **the array we received was allocated elsewhere**. So, we do not count this space when analyzing space complexity. 
2.   
   
3. **Same Question here**: Why do we not just calculate the number of types required by the Algo?
4. Answer: It is impossible and we are more interested in how needed amount of space grows as input size grows.
5. 

Recap

1. What does a problem mean?  
   We said we have some input and searching for particular output.
2. What does a solution mean?  
   Solution is an algo that processes the input and produces the output we are searching for.
3. Then we talked about resources (Time & Space) required by an Algo.  
   We said that we will learn how to measure them with Complexity Analysis.
4. Then we said that a problem may have multiple solutions and Complexity Analysis helps to compare them and choose the most efficient one.
5. Then we discussed what Complexity Analysis mean and we said that we use it to describe the amount of needed resources in terms of input size.
6. Then we talked about time complexity which we can get by calculating the number of operations then simplifying.
7. Then we talked about Space Complexity which we can get by calculating the amount of extra space an Algo uses then simplifying.