# Data Structures and Algorithms By Umar Saad

# Table of contents

## Strategy for solving a problem

1. State the problem clearly
2. Come up with some example in/outputs, trying to cover all edge cases
3. Come up with a solution
4. Implement the solution with testing
5. Understand the complexity and identify inefficiencies
6. Fix the inefficiency, repeat steps 3 - 6

### **The problem at hand:**

**Alice has some cards with numbers written on them. She arranges the cards in decreasing order, and lays them out face down in a sequence on a table. She challenges Bob to pick out the card containing a given number by turning over as few cards as possible.Write a function to help Bob locate the card.**

### 1a. Stating the problem clearly

Essentially the problem is we need to find the specific card’s location (index) which holds the target (target) from a list of numbers (nums) that are in decreasing order with the least amount of card checks possible.

### 2. Analysing the inputs, Outputs and edge cases

In this step we need to ensure we understand the question, the inputs and outputs, and all possible edge cases, ask as many questions here.

2a. Inputs

* List of the card: Nums
* The number we want to find: Target

2b. Outputs

* The index of the card which holds target: Position

3c. Edge cases

You dont need to cover every single edge cases but covering around 1-3 is good practise.

* The target is not in the list
* The target is the last element of the list
* The target is the first element of this list
* The list is only one element
* The list has no elements
* The list has duplicate numbers
* Target is duplicated

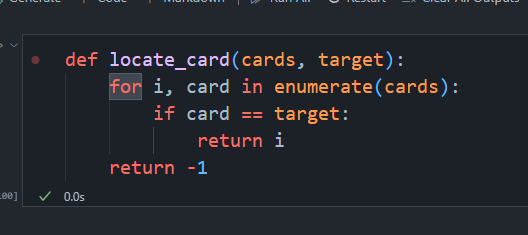
### 3. Come up with a solution

This first solution doesn't have to be the BEST solution of the most Efficient. For now we just need to solve the problem.

In plain English:

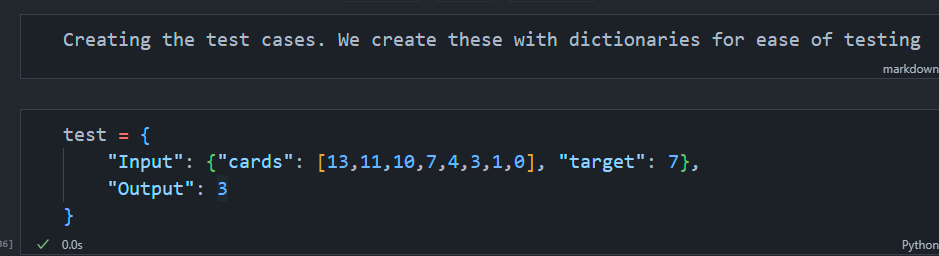
Thinking of the problem at hand, the easy way to solve this is by looping through each card until I find the target, brute force. Loop through with an index and value and find the card with the value and return the index of that number. If the number isn't found we can return -1.

### 4a. Implementing the solution

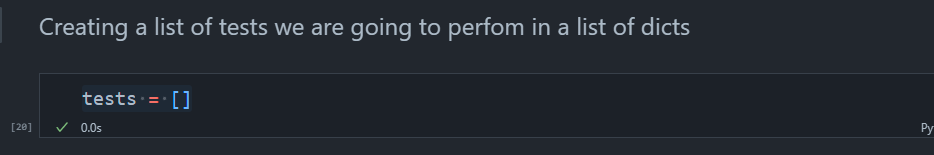


### .4b. Testing our solution - Test Cases

### Now we can start by creating test cases for our solution. To create a test case we ..

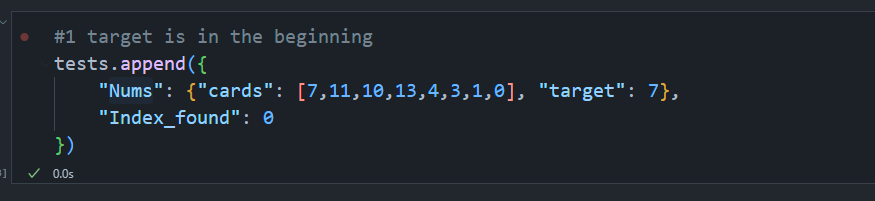


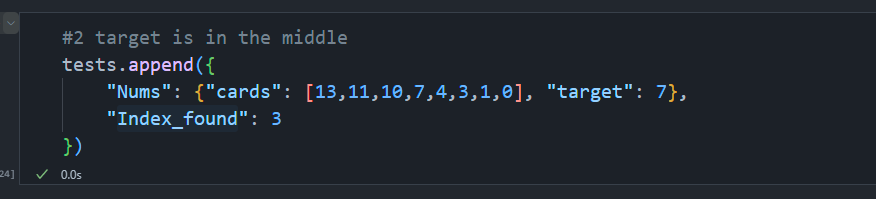
To solve these test cases we create a list called tests which holds each test for the edge cases

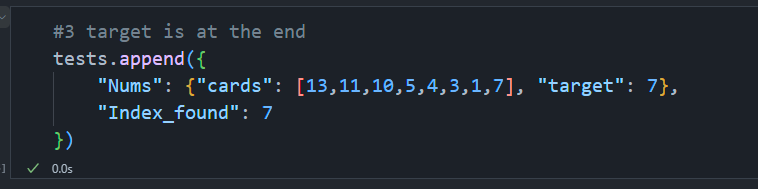


And then we append our edge cases we mentioned above

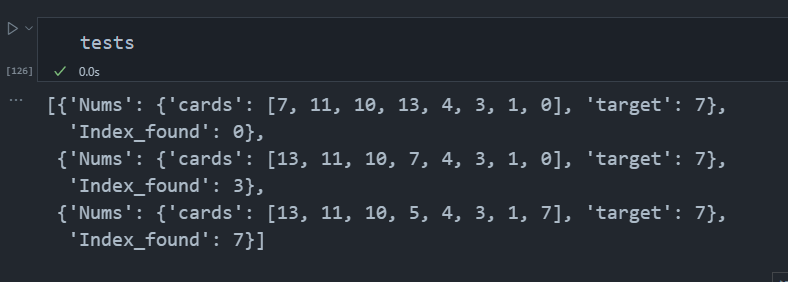
Target is in the beginning, middle or end



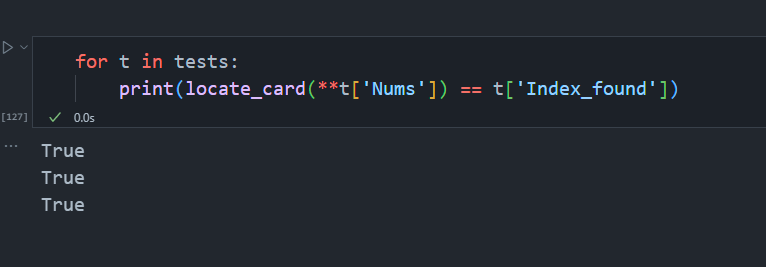




So now we can see our tests list (used 3 examples of edge cases)

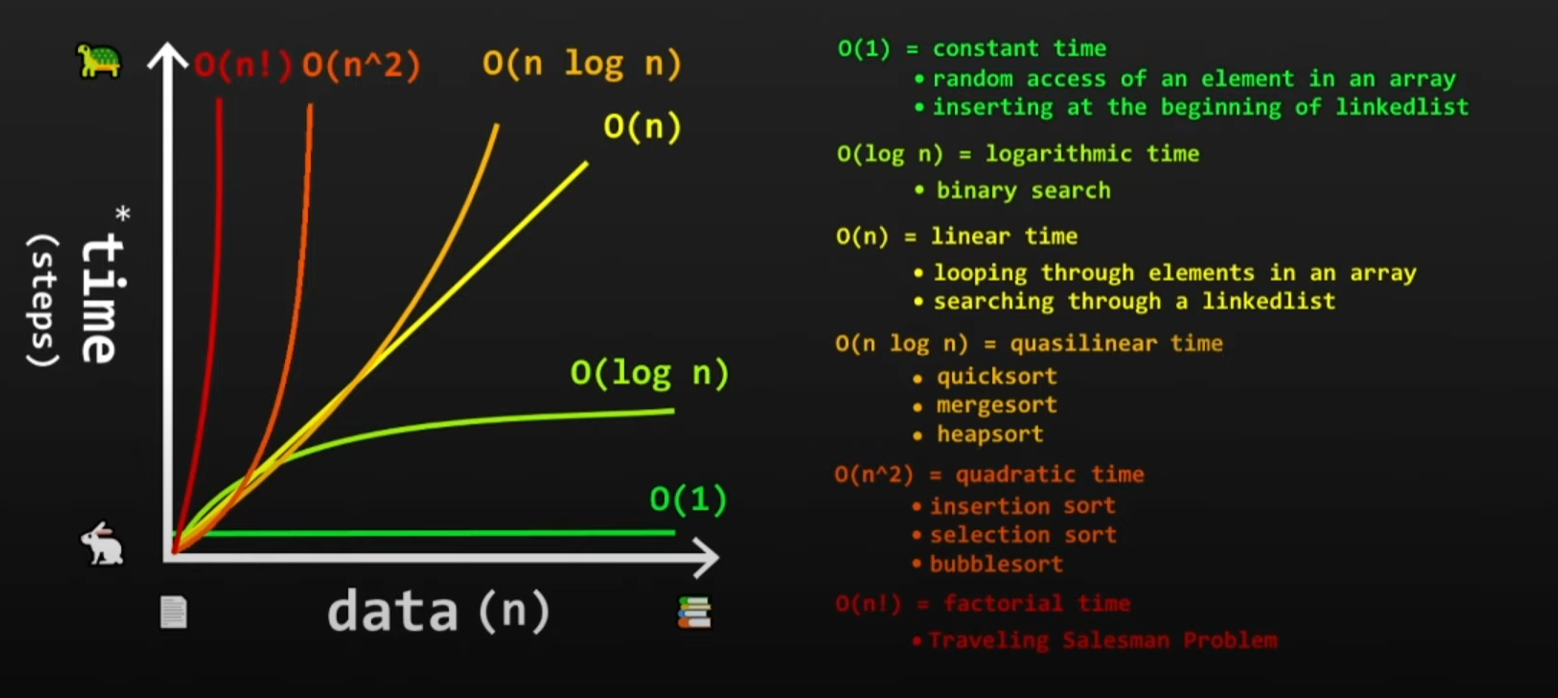


And as we can see our 3 test cases have passed as they have found the target as we check if the input.



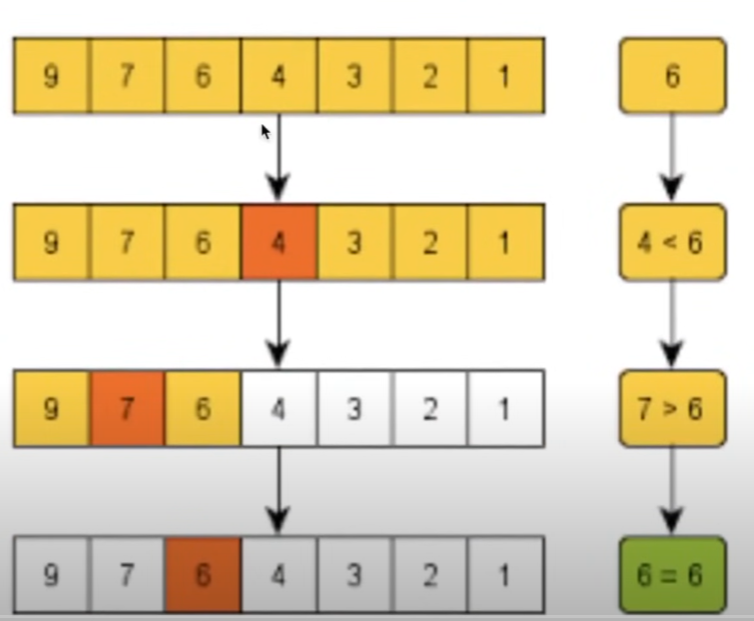
5. Understanding the Complexity & Big O notation of our solution: -

After correctness, aim for efficiency. Now analyse the complexity and efficiency of our current solution. Right now we are....

[](https://www.youtube.com/watch?v=XMUe3zFhM5c)

6. Overcoming the inefficiency with our linear search (brute force) solution using big O

Currently we know that our list is sorted which we can use to our advantage. This means that we can obviously intergrate a binary search.

[](https://www.youtube.com/watch?v=XMUe3zFhM5c)

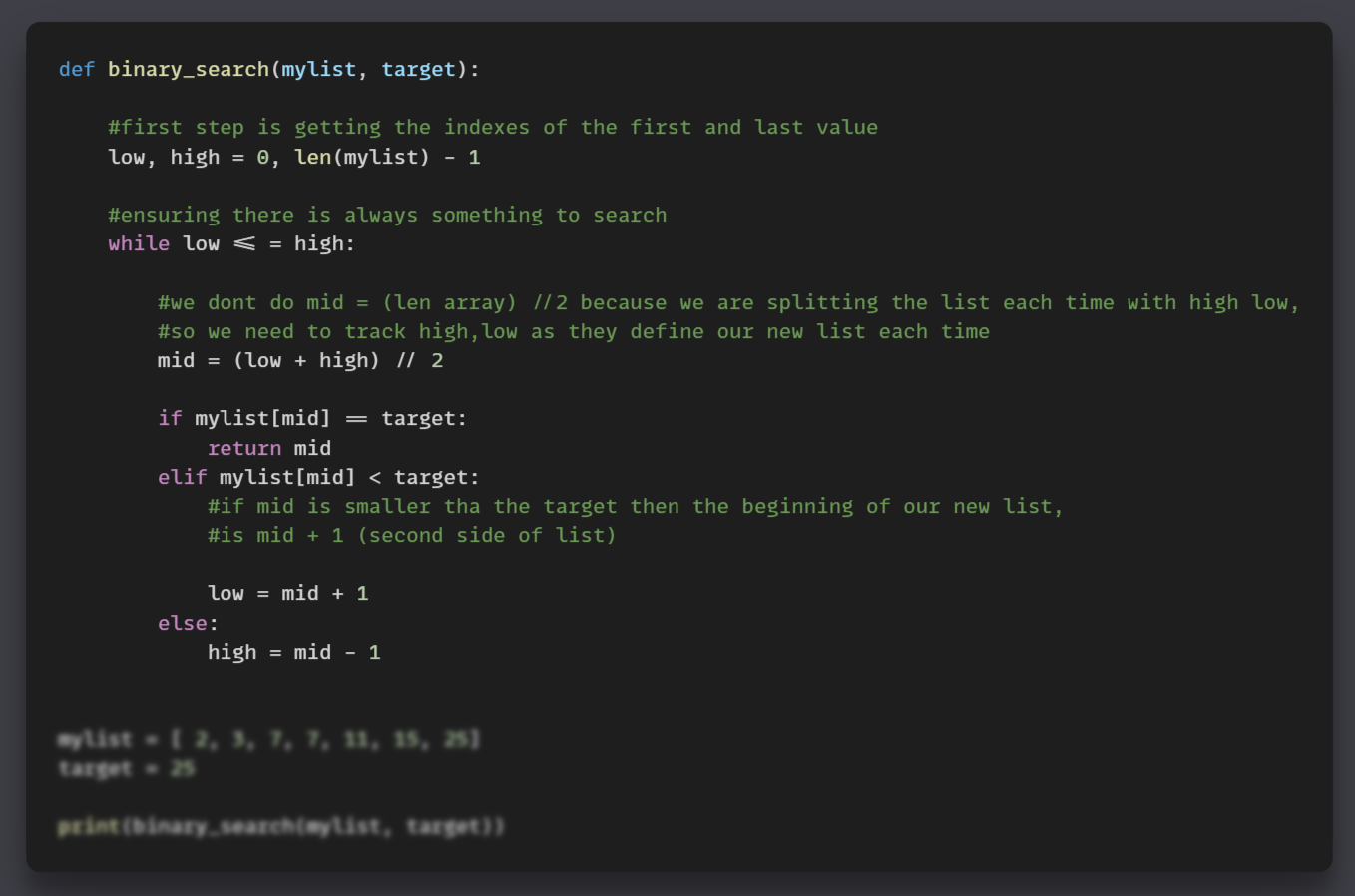
### 3. (revisited). Come up with a solution in plain English

Now that we have finished step 6 and found a solution to our inefficiency we now need to to re-do steps 3-6.

Above we agreed to use a binary search. In our problem they way we will implement is

1. Find the middle element of the list
2. If the middle element is the target, return the element
3. If it is less than the target, then search the first half
4. If it is greater, then search the second half
5. If it is not found, return -1

### 4. Now we have to implement our new solution



This solution works but doesn't cover our test cases. The test cases we need to cover are:

1. The target is not in the list
2. The target is the last element of the list
3. The target is the first element of this list
4. The list is only one element
5. The list has no elements
6. The list has duplicate numbers
7. Target is duplicated

So we start by creating our test case function for each test case.

Latest Notes

Video stopped at: 1hr 5 mins

To learn:

Understanding appending dictionaries to lists and making this a staple in how we perform test cases.