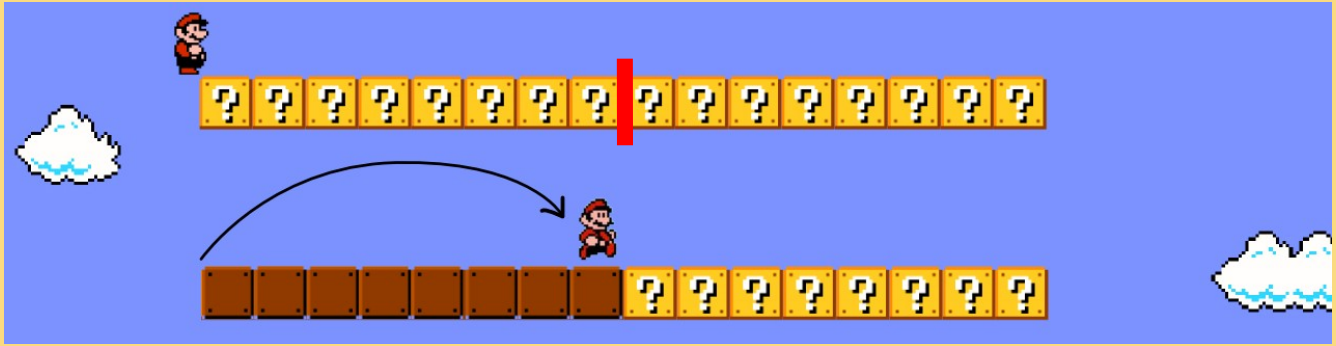
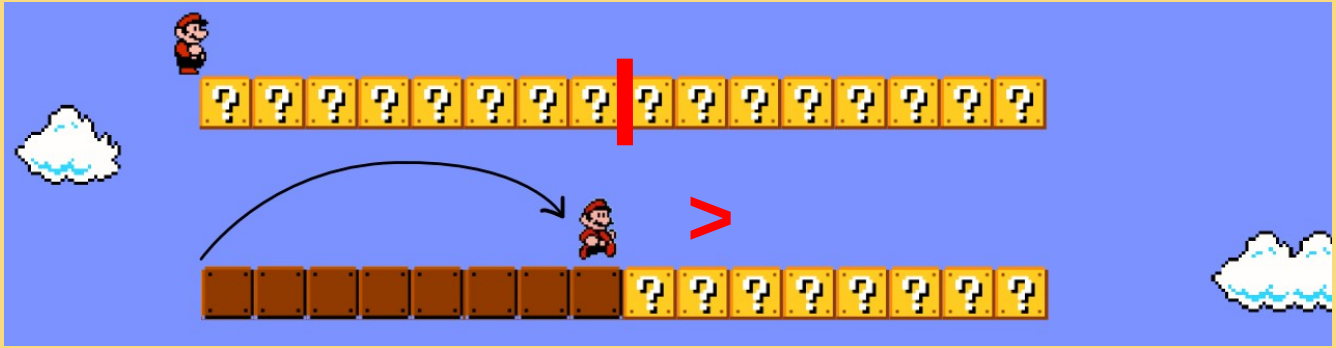
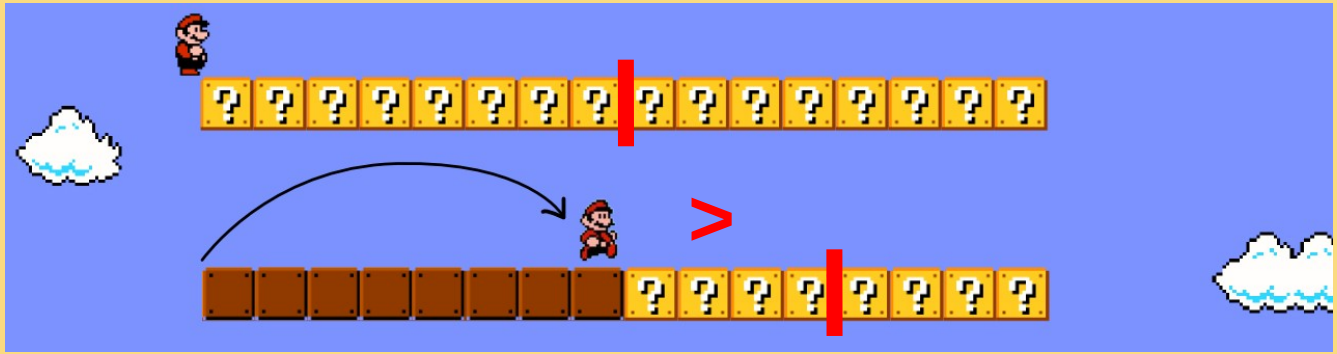
Algorithms: Binary Search

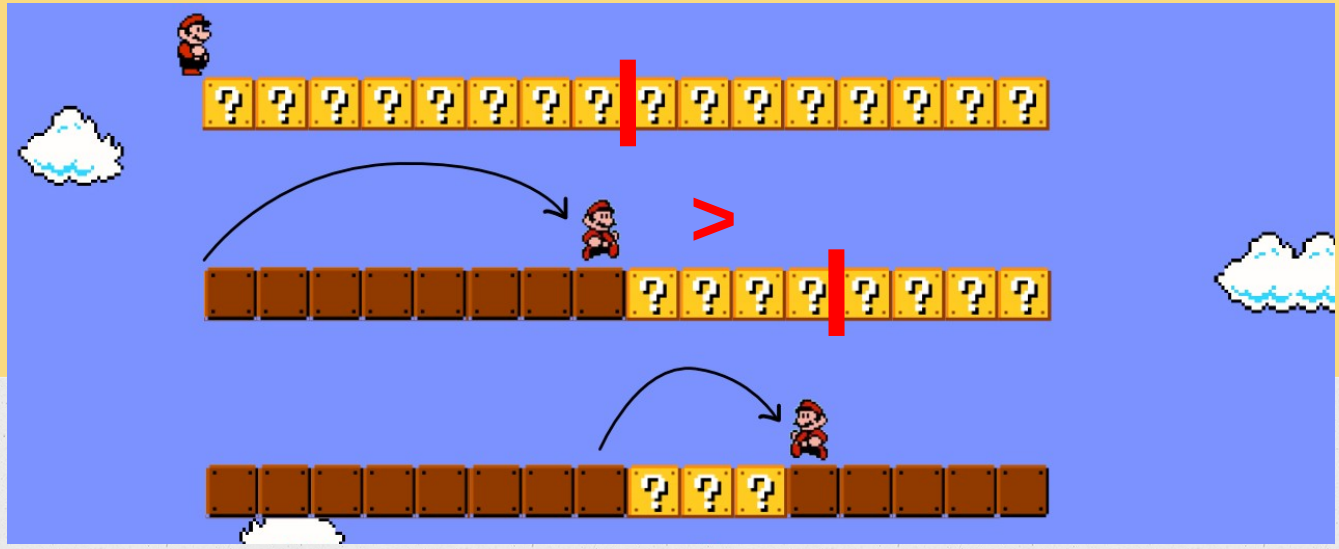
Marie Cho

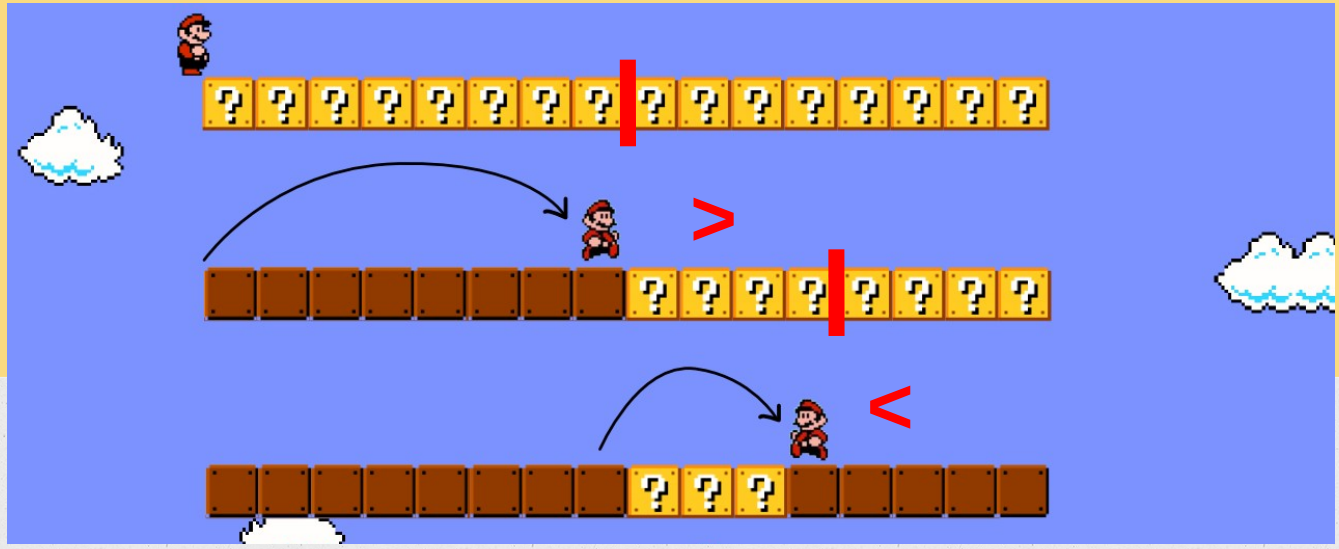
Binary Search: search algorithm that identifies index of target value in a sorted array by dividing the search scope in half.

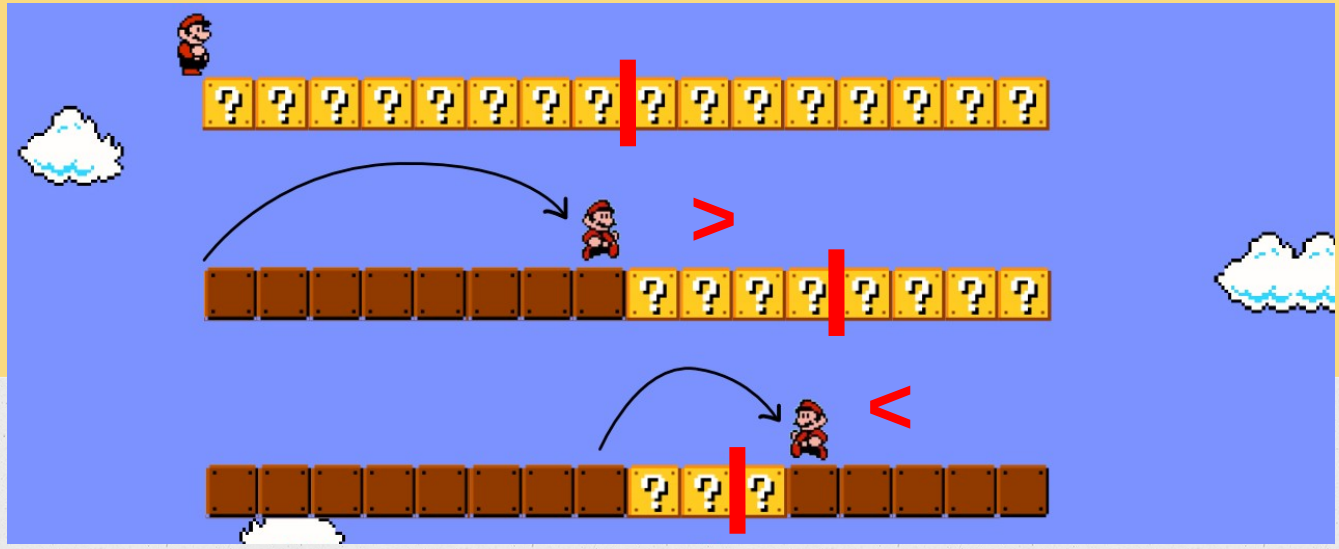


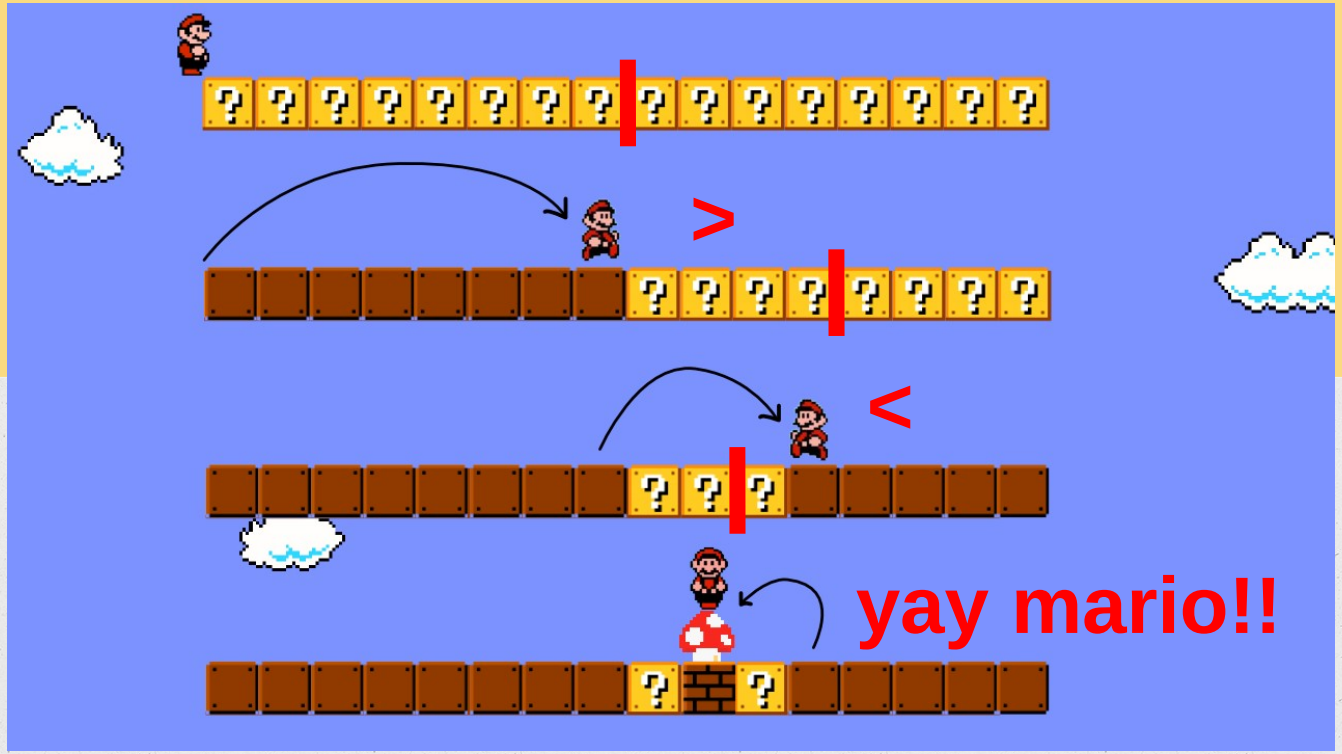











>
<
yay mario!!

[1]

OR

[1,2]

```python
def binary_search(nums, begin, end, value): # list, beginning index, ending index, searched value

    if len(nums) == (1 or 2): # if only 1 or 2 elements
        for i in range(0,len(nums)): # search for value individually
            if nums[i] == value:
                return i
            else:
                return "not in list"
```

[1,2,3,4,5,6,7,8,9]

[1,2,3,4,5,6,7,8,9]

```
mid = (begin+end)//2 # calculate middle index

if nums[mid] == value: # if middle element is value
    return mid
```

[1,2,3,4,5,6,7,8,9]

[1,2,3,4,5,6,7,8,9]

[1,2,3,4,5,6,7,8,9]

[1,2,3,4,5,6,7,8,9]

```
elif value < nums[mid]: # if value is less than middle element
    return binary_search(nums, begin, mid, value) # repeat process from beginning to middle index

else: # if value is greater than middle elements
    return binary_search(nums, mid+1, end, value) # repeat process from middle index to end
```

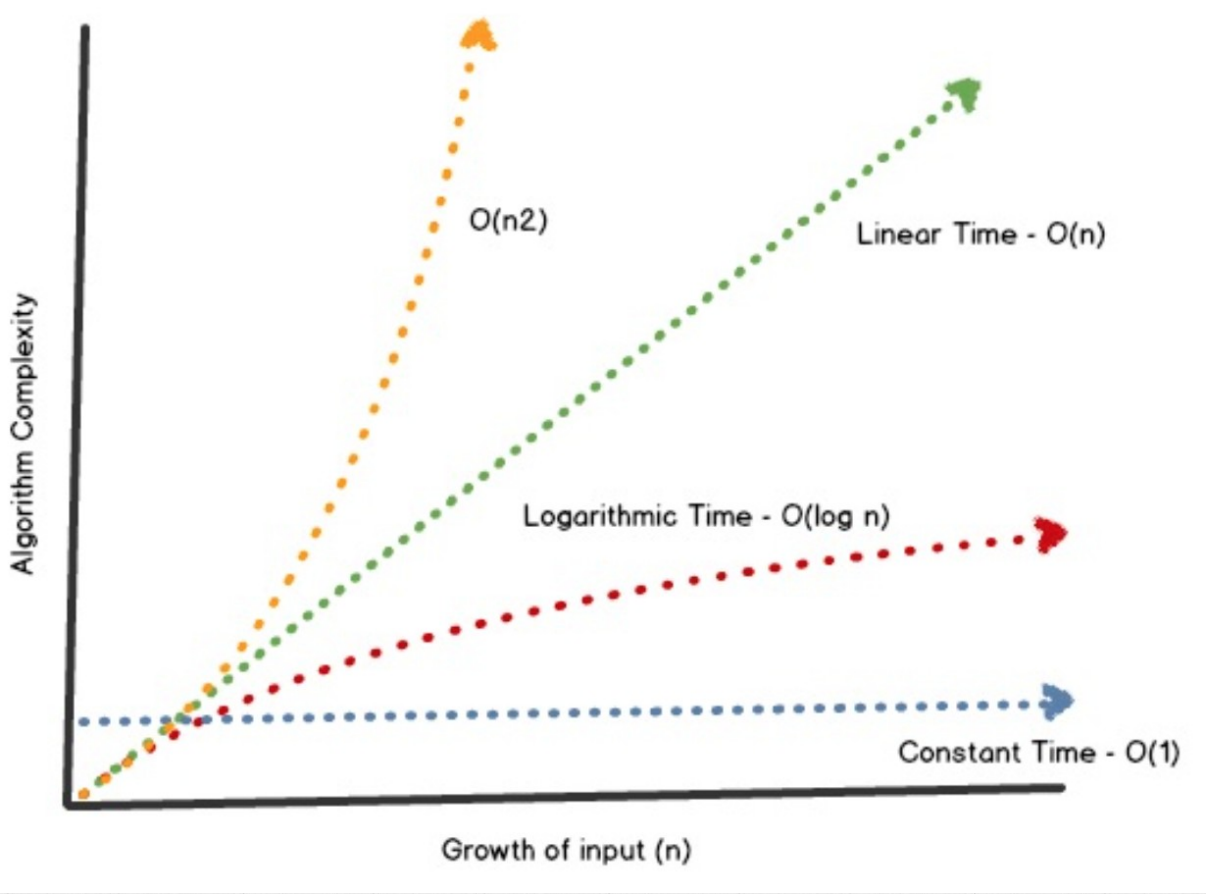
```
28  list = [1,2,3,4,5,6,7,8,9]
29  index = binary_search(list, 0, len(list), 5)
30  print("Index:", index)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER

```
~ % /usr/bin/python3 "/Users/mariecho/Downloads/binary_search (1).py"
Index: 4
~ %
```

O(n): the time complexity and efficiency for a specific algorithm

For binary search: average time complexity is O(log(n))


O(n2)
Linear Time - O(n)
Logarithmic Time - O(log n)
Constant Time - O(1)
Algorithm Complexity
Growth of input (n)

Why O(log(n))?

Because we are cutting the search scope by half every iteration we are able to search faster and faster each time and don’t need to check every number.

THANK YOU! :)