Binary Search

The Big O notation for Binary Search is **O(log N)**. In contrast to O(N) which takes an additional step for each data element.

O(log N) means that the algorithm takes an additional step each time the data doubles.

Array in Ascending/Descending Order.

Recursive :

(Returns index of x in arr if present, else -1)

```
def binary_search(arr,low,high,x):
# Check base case
  if high >= low:

  mid = low + (high-low) // 2  # or (low+high)//2 depends on conditions

  # If element is present at the middle itself
    if arr[mid] == x:
      return mid

  # If element is smaller than mid, then it can only
  # be present in left subarray
    elif arr[mid] > x:
      return binary_search(arr, low, mid - 1, x)

  # Else the element can only be present in right subarray
    else:
      return binary_search(arr, mid + 1, high, x)

    else:
  # Element is not present in the array
      return -1
```

Iterative:

(Returns index of x in arr if present, else -1)

```
def binary_search(arr, x):
low = 0
high = len(arr) - 1
mid = 0
  while low <= high:
    mid = (low+high) // 2
  # If x is greater, ignore left half
    if arr[mid] < x:
      low = mid + 1
  # If x is smaller, ignore right half
    elif arr[mid] > x:
      high = mid - 1
  # means x is present at mid
    else:
      return mid
# If we reach here, then the element was not present
  return -1
```

💕 Thinking

- Correctly initialize the boundary variables `left` and `right`. Only one rule: set up the boundary to **include all possible elements**;
- Decide return value. Is it `return left` or `return left - 1`? Remember this: **after exiting the while loop, `left` is the minimal k satisfying the `condition` function**;
- Practice on setting up Conditions.

Practice Sets:

- ▼ 704 Binary Search
- ▼ 278 First Bad Version
- ▼ 35 Search Insert Position