### Algorithm: Binary Search

Binary search is an efficient algorithm for finding an item from a sorted list of items. It works by repeatedly dividing in half the portion of the list that could contain the item, until you've narrowed down the possible locations to just one.

#### Initialize Data Structures:

* Use three pointers: left, right, and mid to represent the search range and the middle element.

#### Search for the Target:

* Calculate the mid-point and compare it with the target.
* Adjust the search range based on the comparison.

#### Repeat:

* Repeat the process until the target is found or the range is empty.

#### Implementation:

| **def** binary\_search(nums: List[int], target: int) -> int:  left, right = 0, len(nums) - 1  **while** left <= right:  mid = (left + right) // 2  **if** nums[mid] == target:  **return** mid  **elif** nums[mid] < target:  left = mid + 1  **else**:  right = mid - 1  **return** -1  *# Example usage:*  nums = [1, 2, 3, 4, 5, 6, 7, 8, 9]  target = 4  print(binary\_search(nums, target)) *# Output: 3* |
| --- |

#### Explanation:

Initialize:

* left: The left boundary of the search range.
* right: The right boundary of the search range.

Iterate with Mid-point:

* Calculate the middle index mid.
* If the element at mid equals the target, return mid.
* If the element at mid is less than the target, adjust the left boundary.

### If the element at mid is greater than the target, adjust the right boundary.