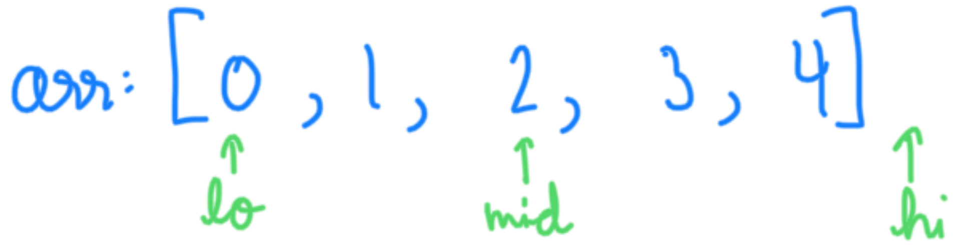
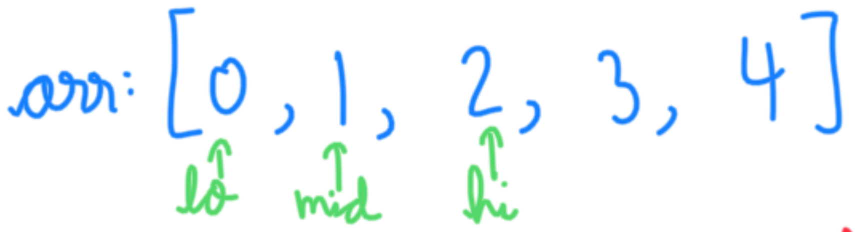
0 Binary Search

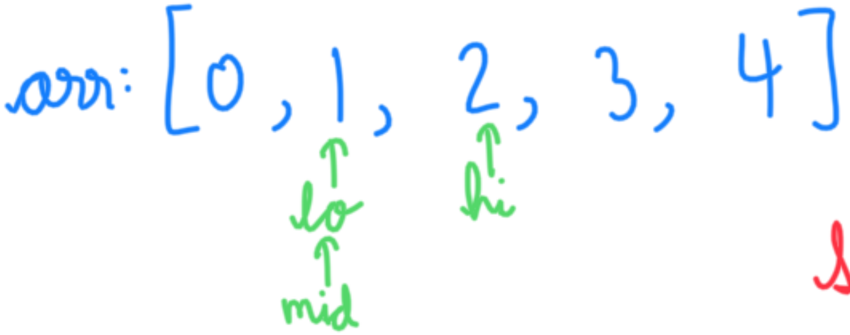


key: 1

Since, arr[mid] < key, hi = mid



Since, arr[mid] ≥ key, lo = mid



Since, lo < hi-1, loop ends.

Since, arr[lo] = key, return lo
else, return null.

Output: 1

Invariant: Binary Search

If *key* ∈ *array*, then at each iteration:

1. *array*[*lo*] ≤ *key*,

2. if $hi \neq n+1$, then $array[hi] > key$.

Combined with the sortedness of array, this implies that *key* (if it exists) lies within the range *[lo..hi)*.

```
$lo = 1$
$hi = n + 1$
                    LI: $arr[lo]$ is $<=$ key, where key is in $arr[lo \ldots hi-1]$
while $lo < hi-1$:
                          LI: $arr[lo] \leq key$, where $key \in arr[lo \ldots hi-1]$
    $mid = \left\lfloor \frac{(lo+hi)}{2} \right\rfloor$

    if $key \geq arr[mid]$:
        $lo = mid$    LI: $arr[lo] \geq key$, where $key \in arr[lo \ldots hi-1]$
    else:
        $hi = mid$    LI: $arr[lo] > key$, where $key \in arr[lo \ldots hi-1]$
                                      Note: If $lo < hi-1$, $lo < mid < hi$
if $arr[lo] == key$:
    return lo         LI: $lo = hi-1$ OR $hi = lo+1$ indicates search space is
else:                     narrowed down to 1 element, which either
    return null           could be the key OR not.
                          $arr[lo] \leq key$, where $key \in arr[lo \ldots hi-1]$
                                                          OR
```

$key \in arr\,[lo \ldots lo-1]$

Code:

```
def binarySearch(key, arr, n):
    lo = 0
    hi = n

    while lo < hi-1:
        mid = (lo+hi)//2
        if arr[mid] <= key:
            lo = mid
        else:
            hi = mid

    if arr[lo] == key: return lo
    else: return None
```

Time:

$$T(n) = \begin{cases} T(n/2) + a, & \text{if } n > 1 \\ b, & \text{if } n = 1 \end{cases} \longrightarrow T(n) = a \log_2(n) + b$$

Worst-Case: $O(\log n)$ Best-Case: $O(1)$

Space: $O(1)$ In-place: Yes, as aux space is $O(1)$