Find Peak Element

Key Idea:

Look at the slope around a middle index:

- Pick mid:
- Compare nums[mid] and nums[mid+1] (safe if you keep the search range so mid+1 is valid).
  - If nums[mid] < nums[mid+1], you are on an uphill slope → a peak must exist to the right. Move lo = mid+1.
  - Else, you are on a downhill (or at a local ridge) → a peak must exist to the left or at mid.

Why is this safe? Because with the "$-\infty$" padding outside the array and the nums[i] != nums[i+1] constraint, any uphill must crest somewhere; any downhill came from a crest to the left.

Algorithm (binary search):

- Initialize lo = 0, hi = n-1.
- While lo < hi:
  - mid = (lo + hi) / 2.
  - If nums[mid] < nums[mid+1]: lo = mid+1
  - Else: hi = mid
- Return lo (or hi; they're equal). That index is a peak.

Complexity:

- Time: O(log n)
- Space: O(1)