Convert Sorted Array to Binary Search Tree

This one is a textbook "build a BST from a sorted array":

Key idea:

Pick the middle element as the root so that left and right halves have (almost) the same size.
Recursively do the same for the left half (becomes left subtree) and the right half (right subtree). That guarantees the tree is height-balanced.

Steps:

1. Define a helper that takes a half-open range $[L, R)$ (or closed $L \dots R$) into nums.
2. Base case: if the range is empty, return null.
3. Midpoint: $mid = (L+R)/2$ (or $L + (R-L)/2$ to avoid overflow).
4. Create node with nums[mid].
5. Recurse left on $[L, mid-1]$, right on $[mid+1, R]$.
6. Return the node.

Why balanced?

At each step you split the array into two halves whose sizes differ by at most one, so subtree heights differ by at most one all the way down.

Complexity:

- Time: $O(n)$ - each element becomes exactly one node.
- Space: $O(\log n)$ average recursion depth (balanced), $O(n)$ for the output tree.