Construct Quad Tree

Problem shape:

- Grid is $n \times n$ with $n = 2^x$, $x \leq 6$ (so $n \leq 64$).
- Each node covers a square subgrid.
- A node is a leaf iff its subgrid is uniform (all 0s from all 1s).
- Otherwise it's an internal node with four children:
  - topLeft, topRight, bottomLeft, bottomRight (each size len/2).

How to test "uniform":

Two options:

A) Direct scan (perfectly fine here):
- Check the first cell, then scan the len x len block; if any cell differs, it's not uniform.
- Time is still good because $n \leq 64$.

B) 2D prefix sum (micro-optimization):
- Precompute pref[r][c] = sum of grid[0...r-1][0...c-1].
- Sum of any subgrid is O(1). A subgrid of size len x len is:

```
s = sum(r, c, len)
if s == 0   → all zeros
if s == len * len → all ones
else → mixed
```

- This avoids rescanning the same cells in higher recursion levels.

Invariants to keep straight:

- Leaf nodes: isLeaf = true, val = (subgrid value), children = null.
- Internal nodes: isLeaf = false, val can be either, but many set it arbitrarily to true or to (sum ≥ half); it doesn't matter.
- Base case shortcut: if len == 1, the node is always a leaf with the cell's value (this is equivalent to the uniform check).

Complexity:

- Without prefix sums: Each level touches each cell a constant number of

times overall $\Rightarrow O(n^2)$.

- With prefix sums: Build pref in $O(n^2)$, each node's uniform check is $O(1)$, total still $O(n^2)$.
- Recursion depth is $O(\log n)$.