

Q.

~~Q~~ Flood fill and boundary fill algorithms are usually implemented using recursive function calls.

For large or complex regions, the recursion depth becomes very high, which may exceed the system stack limit and cause a stack overflow ~~pattern~~ problem.

- two alternative approaches to overcome stack overflow :-
- 1. Scan-line fill Algorithm
  - Instead of filling pixel by pixel recursively, the scan line algorithm fills entire horizontal lines (spans) at once.
  - It processes intersections of scan lines with polygon edges.
  - This approach reduces recursion, improves efficiency, and avoids stack overflow.
- 2. Iterative fill using explicit stack or queue.
  - Replace recursion with an explicit stack (DFS) or queue (BFS).
  - Pixels are stored and processed iteratively rather than through recursive calls.
  - Since memory is managed manually, it prevents stack overflow caused by deep recursion.
- Transforming Area fill Algorithm for Pattern filling.  
To convert area fill algorithm into a pattern fill algorithm.
  - Instead of ~~fill~~ filling pixels with a single solid color, use pattern or bitmap.
  - The pattern is repeated across the filled region using modulus operation on pixel coordinates.
  - Each pixel color is selected from the pattern based on its position:

$$\text{Color} = \text{Pattern} [x \bmod \text{width}] [y \bmod \text{height}]$$

This allows textures, designs or tiled pattern to be filled inside closed areas.

Conclusion :-

Flood fill & boundary fill algorithm can suffer from stack overflow due to deep recursion. Scan-line filling & Iterative Stack/Queue based methods effectively overcome this issue.

Area fill algorithms can be ~~extended~~ extended to pattern filling by mapping pixel position to repeating patterns.