Hole-Filling Algorithm - Complexity Analysis & Optimization

1. Complexity of the Basic Hole-Filling Algorithm

For each hole pixel $h \in H$, the algorithm iterates over the boundary pixels B to compute its value. Since each hole pixel will interact with m boundary pixels, the total number of operations is $O(n \cdot m)$.

However, since each hole pixel h has at most |connectivity| boundary pixels, we can refine this:

$n \cdot m \leq n \cdot |connectivity| \cdot n$

Given that |connectivity| is at most 8, we approximate:

$n \cdot m \leq 8 \cdot n^2 \leq 9 \cdot n^2 = O(n^2)$.
Final answer: $O(n^2)$.

2. Approximate Algorithm in O(n)

To achieve high approximated result in more efficient way, we modify the approach by grouping boundary pixels into a small, fixed set of representative pixels.

Approach:

1. 1. Dividing the Boundary Pixels:

   - The boundary pixels B are split into |connectivity| groups.

   - Each group is represented by a single "mean" boundary pixel.

2. 2. Computing Representative Pixels:

   - This step requires iterating through all boundary pixels once, which takes O(n) time (in one iteration, we calculate, compute and store the mean current pixel).

3. 3. Filling the Hole:

   - Instead of summing over all boundary pixels for each hole pixel, we only sum over the O(1) representative pixels, which are the pixel we created at the phase above.

   - Since each calculation is O(1) for O(1) “fixed” boundary pixels, and there are O(n) hole pixels, this step takes O(n) time.

Final Complexity:

• Approximating the boundary pixels: O(n)

• Filling the hole pixels: O(n)

Final answer : O(n).