CITS2200 Project Report

Arun Muthu

May 22, 2020

Contents

1 FloodFill Count

Explanation

floodFillCount uses a non-recursive depth-first search to search for every contiguous pixel matching the brightness of the original pixel. The inductive step is as follows:

Remove the top-most pixel from the stack and check the pixels above, below, left and right. For any pixel that matches the brightness of the original pixel, convert it to black and push it onto the stack. Increase the count by 1 after this is done.

Since the first original pixel is converted to black and pushed onto the stack, the algorithm will convert its matching neighbours to black, the neighbours' matching neighbours to black and so on. Count is incremented by one for every pixel removed from the stack, and hence once the stack is empty (all suitable pixels processed) the algorithm will return a correct count.

The algorithm does take into account the scenario where the starting pixel is black, in which case 0 is immediately returned.

Time Complexity Analysis

The algorithm first creates an empty stack and pushes the original pixel onto it. The algorithm then uses a while loop to perform the depth-first search, stopping when the stack becomes empty. For every iteration of the while loop, the current pixel's 4 neighbours are examined with array accesses and count is incremented by 1. Let these constant time operations be denoted by the variable c. In the worst-case scenario, every pixel in the image matches the brightness of the original pixel and thus the stack will remain non-empty until every pixel in the image is examined. In other words, the while loop will have to execute p times to examine each pixel.

$$\begin{aligned} time &= p \times c \\ &= \mathbf{O(p)} \end{aligned} \tag{1}$$

Hence floodFillCount's worst-case complexity is $\mathbf{O(p)}$.

2 Brightest Square

Explanation