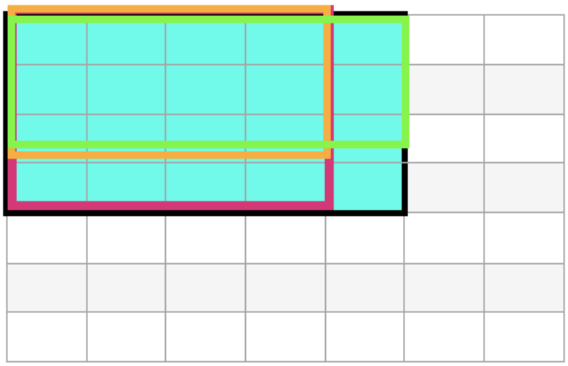
Mubarra Khan
Z5161249

Q4 solution

A clean approach to this problem is to use a prefix sum matrix which would process the orchard and calculate the total number of trees in a square in constant time. In the matrix each psm[i][j] value contains the sum of all values which are above it or on its left.

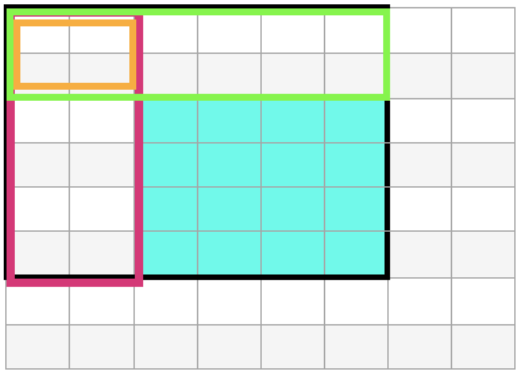
An efficient solution to creating this 2d array is to use previously computed values to compute psm[i][j] . The time complexity for this would be O(1) which is preferred over the simple solution of of $O(n^2)$ that would require traversing and adding values from psm[0][0] to psm[i][j]. The O(1) solution however is done by adding the two neighbouring sums together and subtracting the overlapping middle part to calculate the correct sum . So we add psm[i][j-1] and psm[i-1][j] and subtract psm[i-1][j-1]. A visual representation of this would be

Blue area = pink + green - orange



Once creating the prefix sum matrix we can find the total number of trees of any sized rectangle from any index in constant time O(1). However this question requires us to find a square of size n^2 with the largest number of apple trees. To find this we will go through each cell in the orchard with $O(n^2)$ time complexity. The total number of trees in each cell can be calculated efficiently using the neighbouring sums in the prefix sum matrix. The total value of sum[i][j] will be sum[i][j-n] minus sum[i-n][j] and plus sum[i-n][j-n]. A visual representation of this is

blue area = black - green - pink + orange.



With a second iteration we can determine the maximum trees in the cell of size n^2 with time complexity $O(n^2)$.