COMP3121 Homework Q4

Arth Sanskar Patel
z5228942

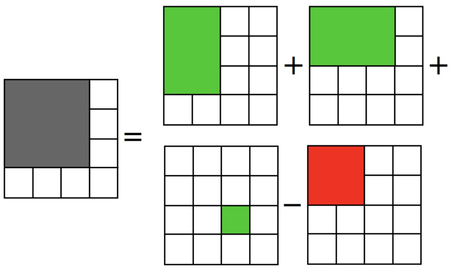
June 15, 2020

0.1 Answer

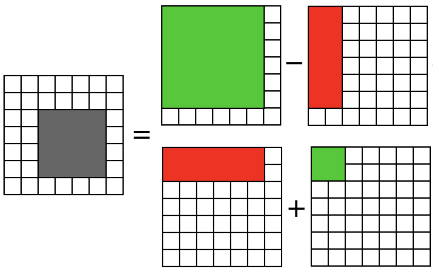
For this part, we are going to use pre-process techniques. Lets say we have a 4n x 4n matrix of apple orchards with each tree having a number which represents the number of apples on it. We create a auxiliary $4nx4n$ matrix with all zeroes (Lets call it B). This can be done in $O(n^2)$ time.

After this we initialise the first element of B equal to the first element of our orchard matrix. We go through the first row of B changing each element to be sum of the elements before it. For eg, the third element in B will be the sum of first,second and third element in our orchard matrix. This can be achieved in $O(n)$ time.

Similarly we go through the first column of B and change it to represent the sum of values before it. This can be again done in $O(n)$ time. Now we traverse the remaining zeros changing them to be the sum of elements before it. For example an element at position (x, y) will be sum of all the number from $(0, 0)$ to (x, y). We don't need to compute the sum by going through all the elements every time. We can simply add element (x, y) to the sum of $(x, y - 1)$ and $(x - 1, y)$ and then subtract sum of $(x - 1, y - 1)$ to prevent double counting. We do this because while traversing through the matrix B, when we reach (x, y), we will already know the sum of $(x, y - 1)$, $(x - 1, y)$ and $(x - 1, y - 1)$. The visual representation of the formula is given below. This will be done in $O(n^2)$ time

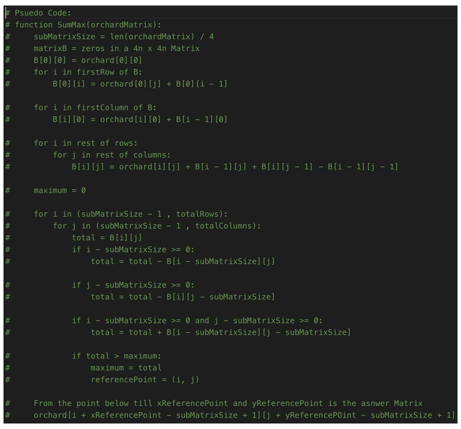


Now we start from the (n, n) element in our matrix B and find the maximum number. We add some conditional if statements in between to make sure we get the sum which is only of nxn matrix. The visual representation for it is given below



Lets say we want to find the sum of the grey area which is a $4x4$ matrix. So we take the number at (x, y) (The bottom right corner of grey area) which is the sum of all numbers till (x, y) (The green area) and subtract the red parts and add the smaller green part again for overlapping. We know all the sums because be calculated all the sums till any point (x, y) from $(0, 0)$ in the previous step. And we update our maximum variable if this is larger than our current maximum. This whole process is done in $O(n^2)$ time. Then we send the reference point (x, y) which had the maximum sum and then take n trees to the left, and top to make a nxn square with maximum sum.

The pseudo code for this is given below



```
# Psuedo Code:
# function SumMax(orchardMatrix):
#     subMatrixSize = len(orchardMatrix) / 4
#     matrixB = zeros in a 4n x 4n Matrix
#     B[0][0] = orchard[0][0]
#     for i in firstRow of B:
#         B[0][i] = orchard[0][j] + B[0][i - 1]

#     for i in firstColumn of B:
#         B[i][0] = orchard[i][0] + B[i - 1][0]

#     for i in rest of rows:
#         for j in rest of columns:
#             B[i][j] = orchard[i][j] + B[i - 1][j] + B[i][j - 1] - B[i - 1][j - 1]

#     maximum = 0

#     for i in (subMatrixSize - 1 , totalRows):
#         for j in (subMatrixSize - 1 , totalColumns):
#             total = B[i][j]
#             if i - subMatrixSize >= 0:
#                 total = total - B[i - subMatrixSize][j]

#             if j - subMatrixSize >= 0:
#                 total = total - B[i][j - subMatrixSize]

#             if i - subMatrixSize >= 0 and j - subMatrixSize >= 0:
#                 total = total + B[i - subMatrixSize][j - subMatrixSize]

#             if total > maximum:
#                 maximum = total
#                 referencePoint = (i, j)

#     From the point below till xReferencePoint and yReferencePoint is the asnwer Matrix
#     orchard[i + xReferencePoint - subMatrixSize + 1][j + yReferencePOint - subMatrixSize + 1]
```

Reference for this answer and the explanation image were taken from here or goto
https://www.techiedelight.com/calculate-sum-elements-sub-matrix-constant-time/.