## Contents

# Additional hints for matrix rotation

## Rotation of row & column position

Consider the $N$ X $N$ matrix $A$:

$$\begin{pmatrix} a_{0,0} & a_{0,1} & \cdots & a_{0,(n-2)} & a_{0,(n-1)} \\ a_{1,0} & a_{1,1} & \cdots & a_{1,(n-2)} & a_{1,(n-1)} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{(n-2),0} & a_{(n-2),1} & \cdots & a_{(n-2),(n-2)} & a_{(n-2),(n-1)} \\ a_{(n-1),0} & a_{(n-1),1} & \cdots & a_{(n-1),(n-2)} & a_{(n-1),(n-1)} \end{pmatrix}$$

As you see, each element of $A$ is identified by its row and column position, indicated by its subscripts. The element in row $i$, column $j$ is $a_{i,j}$. (In mathematics, it's a bit more common to start counting at 1 for positions within a matrix, but we'll start at 0 here, just as we do in Java.)

After rotation 90° clockwise, we'll have the matrix $B$, containing all of the elements from $A$, but in new positions:

$$\begin{pmatrix} a_{(n-1),0} & a_{(n-2),0} & \cdots & a_{1,0} & a_{0,0} \\ a_{(n-1),1} & a_{(n-2),1} & \cdots & a_{1,1} & a_{0,1} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{(n-1),(n-2)} & a_{(n-2),(n-2)} & \cdots & a_{1,(n-2)} & a_{0,(n-2)} \\ a_{(n-1),(n-1)} & a_{(n-2),(n-1)} & \cdots & a_{1,(n-1)} & a_{0,(n-1)} \end{pmatrix}$$

A brief inspection shows that everything in row 0 of $A$ (the original matrix) ended up in column $(n-1)$ of $B$ (the rotated matrix); everything in row 1 of $A$ ended up in column $[(n-1)-1]$, or $(n-2)$ of $B$; etc. Clearly, as we iterate across the original rows, starting at 0 and going up to $(n-1)$, the rotated column is starting at $(n-1)$, and going down to 0. Another way of saying this is that row $i$ in $A$ becomes column $(n-1-i)$ in $B$.

On the other hand, everything in column 0 of $A$ is now in row 0 of $B$; the contents of column 1 of $A$ are now in row 1 of $B$, and so on. So column $j$ in $A$ becomes row $j$ in $B$.

Putting together what we've observed, we have

$$a_{i,j} = b_{j,(n-1-i)}, \quad \forall i,j \in \{0, 1, \ldots (n-1)\}.$$

That is, the value in any row $i$, column $j$ of $A$ will be in row $j$, column $(n-1-i)$ of $B$.

## In-place rotation

While it's straightforward to copy all of the elements from $A$ to a new matrix $B$, it's a bit more difficult to see how we might rotate $A$ in place, without creating another array of the same size.

One way to do so is to identify 4 elements at a time, regularly spaced around the matrix, that will move into each other's positions through rotation. For example, in the matrix $A$, above, we see that each of the elements $a_{0,0}$, $a_{0,(n-1)}$, $a_{(n-1),(n-1)}$, and $a_{(n-1),0}$ (the elements at the corners) will rotate into the next element's position: the $a_{0,0}$ will stay in row 0, but will move to column $(n-1)$; $a_{0,(n-1)}$ will stay in column $(n-1)$, but will move to row $(n-1)$; and so on. So one way we could rotate just these 4 values would be to store $a_{0,0}$ in a temporary value; copy $a_{(n-1),0}$ (the element at the bottom-left of the matrix) to row 0, column 0 (top-left); copy $a_{(n-1),(n-1)}$ (bottom-right) to row $(n-1)$, column 0 (bottom-left); copy $a_{0,(n-1)}$ (top-right) to row $(n-1)$, column $(n-1)$ (bottom-right); finally copy the value originally located at row 0, column 0 (top-left)—which we initially copied to a temporary variable—to row 0, column $(n-1)$ (top-right).

By repeating the above process, starting the sequence with all of the elements in the top row (not including the right-most element, which has already been moved), we rotate all of the elements in the entire outer "ring" of the matrix. We can then do the same for the next ring of elements within that, continuing in that fashion until we have rotated the entire matrix.