Time complexity of MirrorMazeBruteForce Algorithm

## Best cases:

First, we analyze the time complexity of each of the methods that getSolution() invokes, after that we add those time complexities in order to get the time complexity of the getSolution() method.

getStartingPosition():

Ω(1)

getMirrorPositions(maze) - Ω(width\*height), if width = m and height=n, then we can write:

Ω (m\*n)

getCombinations(mirrors.length) - Ω (2mirrorNumber\*mirrorNumber), where mirrorNumber=width\*height-2\*(width+height) in the worst scenario, if width = m and height = n, then we can write the time complexity function for the best case of this method as:

Ω ( 2m\*n-2\*(m+n) \* (m\*n-2\*(m+n)))

getSolution()(lines 52 up to 64)- Ω (2mirrorNumber \*(mirrorNumber)), where mirrorNumber = m\*n-2\*(m+n) in the worst scenario, as discussed previously, so:

Ω ( 2m\*n-2\*(m+n) \* (m\*n-2\*(m+n)))

Finally, we add these time complexities to get the time complexity of getSolution() method.

C + m\*n + 2m\*n-2\*(m+n) \* (m\*n-2\*(m+n)) + 2m\*n-2\*(m+n) \* (m\*n-2\*(m+n)) = C + m\*n + 2m\*n-2\*(m+n)+1 \* (m\*n-2\*(m+n))

C – constant which represents the constant time complexity Ω(1) of the getStartingPosition() method

## Worst cases:

getStartingPosition()- O(2\*(height+width)-8), if height=width, and we represent them as “n”, then we can generalize time complexity- O(4n-8)

getMirrorPositions(maze)- O(width\*height), if width=height, and we represent them as “n”, then we can generalize time complexity - O(n2)

getCombinations(mirrors.length)- O(2mirrorNumber\*mirrorNumber), where mirrorNumber=width\*height-2\*(width+height) in the worst scenario

getSolution()(lines 52 up to 64)- O(2mirrorNumber \*(mirrorNumber+(height-2)\*(width-2)+2)), where mirrorNumber in the worst scenario is the same as above