**Time complexity**

|  |  |  |
| --- | --- | --- |
|  | diagonal\_difference\_1 | diagonal\_difference\_2 |
| Best case | O() | O() |
| Average case | O() | O() |
| Worst case | O() | O() |

**diagonal\_difference\_1**

The algorithm requires *two nested loops* to complete a single iteration on a matrix of size ; therefore, it must go through \* steps to complete the traverse.

Growth is quadratic.

**diagonal\_difference\_2**

The algorithm takes advantage of the previously known pattern of diagonal indices and uses *one loop* counter to extract the diagonal elements from the matrix; therefore, it must go through steps to compute the required summation.

Growth is linear.

**Space complexity**

Both algorithms depend on the input size of the matrix.

O( )

For both algorithms, space grows linearly with the input size of the matrix and constant space is used to initialize diagonal sums.