MATH 282 Analysis of Algorithm’s Complexity

**Algorithm:** Matrix addition

**Factor to be analyzed:** Time (number of steps, speed)

**Situation to be analyzed:** Square matrix (assumed) of size *n* by *n*

**Explanation of situation:** Number of rows is same as number of columns (*n* by *n*)

Makes analysis easier

**Key step (reflects work done):** Accessing the matrix (getting or setting an element)

**Parameter for analysis:** *n* is the number of rows or columns of the square matrix

*(what determines how the algorithm’s efficiency changes as the size of data increases?)*

**Questions/Process:**

* Are there any steps which are not simple steps (comparable to the key step)?
  + If so, what is the complexity of those steps (relative to the key step)? Factor into the analysis.
* If desired, count the number of times each step is carried out (or just the key step).
* Identify each loop and determine how many times the loop is carried out (in relation to *n*).
* How are the loops related?
  + If nested, multiply the steps.
  + If separate, add the steps.
* Eliminate any constants and any lower-level terms.

Shortcut analysis: There are two loops that are nested; each loop is carried out *n* times. There are *n* \* *n* = *n*2 steps being done.

Technically, the creation of a new matrix is also a complex step, so we should analyze it too. Creating a new matrix involves two loops that are nested, so that is also *n*2. That is completely done before the addition is carried out, so we have *n*2 + *n*2 steps, which is still O(*n*2).  
  
Detailed analysis: We can ignore the 3 assignments and 2 comparisons at the start since they are not matrix accesses and are not in the loop.

Creating a matrix is not a simple steps, so we must analyze that step:

There are two loops; each loop is carried out *n* times

(once through every row, once through every column)

The loops are nested, so multiply the steps: *n* \* *n* = *n*2 steps to create a new matrix

Then we must analyze addition:

There are two loops; each loop is carried out *n* times

(once through every row, once through every column)

The loops are nested, so multiply the steps: *n* \* *n* = *n*2 steps to add the two matrices

(actually 3*n*2 steps to get two values and set the value in the sum matrix)

Creating the new matrix is done separately (completed before) the addition, so we add the steps:

*n*2 + 3*n*2 = 4*n*2 steps; we can ignore constants.

**Result:** Algorithm is O( *n*2 )