Problem 1. Arrange the following functions in order of increasing growth, $2^{lg n}$, $2^{2^{lg n}}$, $n^{5/2}$, $2^n$, $n^2lg n$

Problem 2. Assume you have an array $A[1, \ldots, n]$, where every value is an integer between 1 and $n$, inclusive. You do not have direct access to the array $A$. You do have a function $equal(i, j)$ that will return TRUE if $A[i] = A[j]$, and FALSE otherwise.

(a) Give a quadratic ($\Theta(n^2)$) algorithm that counts the number of pairs $(A[i], A[j])$ ($i \neq j$) such that $A[i] = A[j]$. The algorithm can only use a constant amount of extra memory. Just give the “brute force” algorithm.

(b) Analyze exactly how many times the algorithm calls $equal(i, j)$ (as a function of $n$). Justify.

Problem 3. We are going to generalize Problem 1 to two dimensions. Assume you have a 2-dimensional array $A[1, \ldots, n; 1, \ldots, n]$, where every value is an integer between 1 and $n^2$, inclusive. You do not have direct access to the array $A$. You do have a function $square(i, j, k)$ (where $1 \leq i < i + k \leq n$ and $1 \leq j < j + k \leq n$) that will return TRUE if the four values $A[i, j]$, $A[i + k, j]$, $A[i, j + k]$, and $A[i + k, j + k]$ are all equal, and FALSE otherwise.

(a) Give a cubic ($\Theta(n^3)$) algorithm that counts the number of squares $A$ has. The algorithm can only use a constant amount of extra memory. Just give the “brute force” algorithm.

(b) Analyze exactly how many times the algorithm calls $square(i, j, k)$ (as a function of $n$). Justify.