Problem 1. Suppose we have two sorting algorithms. One of them performs at most $\frac{1}{2}n^2$ (runtime) operations and the other algorithm performs at most $6nlg\,n + 6n$ (runtime) operations. Show their runtime in a single plot as n grows and answer the following questions:

(a) Which algorithm would you prefer for smaller values of n? Why?

(b) Which algorithm would you prefer for larger values of n? Why?

(c) If your choice for the previous two parts is not the same, what is the approximate cross over point for n when your preference for one or the other algorithm changes?

Problem 2. Assume you have an array, A, of length, 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). Show your work.

Problem 3. We are going to generalize *Problem 1* to two dimensions. Assume you have a 2-dimensional array, A, of size, $n \times n$, 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) Given 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). Show your work.