1. (25%)

   (a) What is the smallest problem size n_0 such that algorithm B is (strictly) faster than algorithm A for all $n \geq n_0$?
   Setting $0.03n^2 = 0.15n \log n + 0.00001n^2$, we find the these two functions intersect at $n = 1.1772$ and $n = 22.451$, and $0.03n^2$ is smaller over that interval. Because $n = 1$ is a trivial case, this means that the smallest number n such that running time of $B = 0.15n \log n + 0.00001n^2$ seconds is faster than the running time of $A = 0.03n^2$ seconds is at $n = 23$ points.

   (b) What is the smallest problem size n_1 such that algorithm C is (strictly) faster than algorithm B for all $n \geq n_1$?
   Setting the running time of B equal to the running time of C, we must solve the following equation for n:

   $$0.15n \log n > n$$

   Simple algebra results in the relation:

   $$n > 2^{\frac{1}{.15}} = 2^{\frac{20}{3}} = 101.549$$

   So C is strictly faster that B for $n \geq 102$.

   (c) Describe how to construct a distance computing algorithm that always achieves the best running time of any algorithm A, B, and C on it's input.
   This would simply be to create an algorithm that runs either A, B, or C based on input size. So, we can create algorithm D that runs either A, B, or C based on n, the size of the input:

   $$D = \begin{cases} A & \text{if } n < 23 \\ B & \text{if } 102 > n \geq 23 \\ C & \text{if } n \geq 102 \end{cases}$$

   This would always result in the fastest possible output of all three algorithms.

   (d) Professor Nikrasch suggests processing the Lotto results using a different clustering algorithm altogether, one which avoids computing distances between results. This new algorithm runs in $n^{1.2}$ seconds of an input of size n. Is this algorithm ever faster than the fastest of A, B, or C? If so, for what value of n does it start to win?
   In order to check if this new algorithm (call it E for simplicity) is ever faster than the fastest of A, B, or C, we can solve for n in the following relation (where each character represents the runtime of the algorithm in terms of input size n):

   $$E < D.$$

   Solving for this, we find that $n = 1.65235 \times 10^6$, or $1,652,350$ points.