c) The storage and access times are likely very different with the larger input size n. We might have been forced to look up the input data from slower memory, or the caching performance might have been better for the first trial with 1000 elements. Another source of this disparity could be some optimizations the library we are using makes use of for small n's. Another less likely source is that the computer was slowed down by another process hogging the CPU's execution time.

2) a) The inner loop assumes that all elements in $A[0, \ldots, j]$ are less than the variable min, which is initially equal to $A[j]$. The inner loop also assumes that, as a corollary, $A[j+1, \ldots, n] \geq A[0, \ldots, j]$.

Proof: By induction.

Base case: $j = 0, i = j+1 = 1$

The inner loop searches $A[1, \ldots, n]$ for a minimum value and keeps that in the variable min.

After the inner loop has iterated, the outer loop swaps that minimum value to $A[0]$. The minimum value is guaranteed to be less than or equal to $A[0]$, so then $A[0]$ is the smallest element in the array and the loop invariant holds.

Inductive Step: Assume $i = j+1$. As $A[0,\ldots,j] \leq A[i,\ldots,n]$, and the inner loop finds the smallest value in $A[i,\ldots,n]$, the outer loop maintains this relationship by ensuring that minimum value in $A[i,\ldots,n]$ is inserted at $A[j]$ every time j increments by 1, and thus the loop invariant holds for any j s.t. $0 \leq j \leq n$. ■

b) The inner loop runs $n-(j+1)$ times, and $0 \leq j \leq n-1$. Therefore, assuming n is finite, the loop must terminate as the steps inside the inner loop always execute to completion, and the loop runs a maximum of $n-1$ times. ■

c) The outer loop assumes that $A[0,\ldots,j]$ is already sorted as it swaps the minimum value found in $A[j,\ldots,n]$ determined by the inner loop to $A[j]$.

Proof:

Base case: Assume $A[0,\ldots,n]$ is initially unsorted. The minimum element defaults to $A[0]$, as $j=0$. The inner loop determines the minimum value, and the outer loop swaps it to $A[0]$, ensuring the loop invariant is true.

Inductive Step: Assume $A[0,\ldots,j-1]$ is already sorted. As the inner loop finds the minimum value in $A[j,\ldots,n]$ and inserts it into $A[j]$, the loop invariant holds for and $0 \leq j \leq n-1$. ■

d) Assume n is finite. The outer loop contains a function that terminates as well as a for loop that was shown to terminate in b). As the for loop contains elements that all terminate, we can say that the outer loop terminates if n is finite. ■

e) Assume the two loop invariants are true. Thus for any b, $A[b+1] \geq A[b]$.

3) Assume that each person has 0-13 friends, ie. they cannot be friends with themselves. Also assume that if one person is friends with the other, that person is also friends with the first person.

Case one: somebody is friends with everybody, so no one is friends with nobody. As everyone has between 1 and 13 friends and there are 14 people, two people must have the same number of friends via the pigeonhole principle.

Case two: This is the opposite of the first, where nobody is friends with everybody, and as a result somebody is friends with nobody. Therefore, again by the pigeonhole principle there must be somebody who has the same number of friends as somebody else, since everyone has 0-13 friends and there are 14 people.

Both cases are true, so the result is true. ■