CPSC 221 Assignment 2

Scott Pidzarko
33730128

1) a) Asymptotic analysis assumes a "best case scenario" as far as machine specifics. This can cause the theoretical performance of an algorithm to ~~greatly~~ vary greatly from the performance in-practice due to caching RAM performance, RAM size, on-chip memory size, etc.

Additionally, we make no assumptions about the impact of constant factors in asymptotic analysis. This can greatly affect the performance in practice, especially ~~the~~ in the case of a small n, where an inefficient algorithm might be a better choice due to the constant factors.

Besides the two reasons listed above, we can also sometimes get misleading results with asymptotic analysis because of simplifying assumptions made; or because certain steps of the algorithm vary with n, instead of having every step have similar execution times.

b) If O(log(1000)) = 5 seconds, then we would expect O(log(10000)) = 6 seconds.

As log(1000) = 3 sec, the constant factor must be 2 seconds.
So as log(10000) = 4 sec, the execution time with 10,000 elements should take 6 seconds.

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.