Hidden Room of AGCA

By Zain Lateef

My solution uses the Knuth-Morris-Pratt Algorithm (KMP) to find the hint within the rows and columns. The algorithm is a modified version of the naive algorithm. Once a certain amount of the hint has been matched with the target text, but not entirely matched, the algorithm checks whether the suffix and the prefix of the matched string are the same. If they are, then the algorithm skips x comparisons, x being the length of the suffix/prefix. If not, then it continues in the naive fashion. The whole process has a worst case of O(n). When the user inputs a string (a row), I immediately run the KMP algorithm on it and map the results to an N-sized array, and I do this for N inputs (O(n^2)). During this time, I also take each character of the string and map it into a character array of size N^2, N characters apart, to represent the columns as an entire string (O($2n^2$)). Once the input process is complete, I run KMP on each “column” represented in the character array (O($3n^2$)). Finally, with the two rows of keys established, I run a Longest Common Subsequence (LCS) function using the two rows as input. The LCS function computes and compares every possible subsequence of both its inputs, but unlike the naive method, this function uses a 2D array to memorize previous computations to save time. It has a worst case of O(n^2) which leaves the entire program at a worst case of O($4n^2$).