Algorithm Proof

The algorithm implimented runs in $O(N+W+M^2)$ time complexity. First, the inputs are taken through getline() and stored in variables key and hint. Key and hint are then passed as parameters into the constructor of a Password object. The object's variables are set within the constructor and the method showPattern is called to identify the LPS array that aligns with the given hint. The method showPattern runs in O(W) as it loops through all W characters of hint once updating the LPS pattern each iteration. Next, the method makediff is run in O(N) time as it runs through each character of key and compares it to a character of hint. If the characters match, then the next character in each is compared; otherwise, a previous character in hint is compared to the character in key until either the first character of hint fails or a match is found. When a match occurs, the difference between the found index and the last index are stored in an integer array with the exception of the first index which is stored in a temp variable that is updated each time a match is found. Because of the comparisons with W hint characters and the fact that the hint index can only be decremented as far as it has been incremented between 0 and W-1, then the worst case time complexity for finding the differences is O(2*N) which is still O(N). Next, an integer array forming the basis of the Memoization method is created in $O(M^2)$ time by assigning all values in the 0 column or 0 row to zero and sequentially assigning integers to the rest of array in the following way. The array is actually (M+1)x(M+1) so the row indexes {1...M-1} correspond to the diff[] array values in order and the column indexes {1…M-1} corresponds to the diff[] array values in reverse order. If diff[ii] equals diff[jj], then LCSmatrix[ii][jj] equals LCSmatrix[ii-1][jj-1] + 1. If it does not, then LCSmatrix[ii][jj] will equal the maximum value between LCSmatrix[ii][jj-1] and LCSmatrix[ii-1][jj]. Once the matrix is complete, a while loop is used to trace back through the incrementations and the resulting integers that belong in the answer are diff[kk] where kk is the column index where a diagonal trace occurs; otherwise, the trace occurs either horizontally or vertically in the matrix until either another diagonal trace occurs or ii = 0 or jj = 0. In the end, the tracing and matrix creation is $O((M+1)^2 + M)$ which equals $O(M^2+3M+1)$ which is still $O(M^2)$ simplified and the final time complexity becomes $O(N+W+M^2)$.