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 array that is the reverse of the previously found index differences is formed in O(M) time totaling O(N+W+M) time so far. Lastly, the method diffid is called to compare the arrays diff and ffid holding the differences in forward and reverse. A Password object containing containing the arrays, an empty string called code, the key, the hint, and a few other variables is passed into diffid. If the last integers of the array are equal, then the integer is added to the front of the recursively returned code and the object is returned to its previous call. If they are not equal, then the object is copied into two new temporary objects and the index of diff is decremented in the first one and the index of ffid is decremented and the longer of the codes between the two recursive output codes is returned to the previous call. In the worst case, every integer in each array is compared to every integer in the reverse array in O(M^2) time. This in total is O(N+W+M+M^2) which is still equivalent to O(N+W+M^2) time.