Shortest Superstring Problem:
Here this algorithm takes length of kmers say k and the sequence as input.The program initially generates all the kmers using the function kmers() with k as input. Then all the kmers are compared to each other to find which ones have maximum overlap i.e., we start comparing each single kmer with all others and see if it's overlap is greater than the previous one checked. If yes, we update it else leave it as it is.Hence we find the maximum overlap of 1st kmer. Now we join the prefix and suffix of the strings with overlap in between and generate the shortest string possible for the 1st kmer. This goes on for all the kmers and we thus get the shortest superstring.

Complexity analysis:
Function kmers have a time complexity of O(n)
Function find_shortest_superstring have 3 loops (1 while and two for loops) running k no. of times where k is the length of the kmers. And it calls overlap function in between thus increasing the time complexity.
Function overlap has a complexity of O(k) as the strings passed will have length k always.
The total time complexity thus will be O(k*k*k*k) i.e. O(k^4)

We will not always get input as the output. Suppose we find 3-mers for the string acgtacgt we see there will be a few similar kmers which in turn leads to max overlap within themselves thus will not result in same output as input.
Kmers generated stepwise are as follows: [acg,cgt,gta,tac,acg,cgt] -> [acg,cgt,gta,tac,cgt] ->
[acg,cgt,gta,tac] -> [acgt,gta,tac] ->[acgta,tac]->acgtac
The final output is acgtac which is not same as the input