1. (7 points) In this assignment you will implement the basic dynamic program for edit distance and edit string between two input strings. Edit distance is a widely used algorithm with a variety of applications from spell checkers to version control to DNA matching. The basic edit distance algorithm is based on dynamic programming and has the complexity of *O(mn)*. There are 3 possible operations, insert (I), replace (R), and delete (D), each of which costs 1 unit (unless the characters exactly match).

For example, if x= [babble] and y='apple', the best alignment and the corresponding edit distance are.

B A B B L E

\- A P P L E

edit distance: 3

Write a python function called `editDistance' that takes two strings as inputs and returns the edit distance between the two strings.

(1) Make sure that your program works on this test data.

editDistance("ATCAT", "ATTATC")[0] == 2
editDistance("taacttctagtacatacccgggttgagcccccatttcttggttggatgcgaggaacattacgctagaggaacaacaaggtcagaggcctgttactcctat",
"taacttctagtacatacccgggttgagcccccatttccgaggaacattacgctagaggaacaacaaggtcagaggcctgttactcctat")[0]==11
editDistance("CGCAATTCTGAAGCGCTGGGGAAGACGGGT",
"TATCCCATCGAACGCCTATTCTAGGAT")[0]==18
editDistance("tatttacccaccacttctcccgttctcgaatcaggaatagactactgcaatcgacgtagggataggaaactccccgagttttccacagaccgcgcgcgatattgctcgccggcatacagcccttgcgggaaatcggcaaccagttgagtagttcattggcttaagacgcttaagtacttaggatggtcgcgtcgtgccaa",
"atggtctccccgcaagataccctaattccttcactctctcacctagagcaccttaacgtgaaagatggctttaggatggcatagctatgccggtggtgctatgagatcaaacaccgctttcttttagaacgggtcctaatacgacgtgccgtgcacagcattgtaataacactggacgacgcgggctcggttagtaagtt")[0]==112

(2) Generate 10 random pairs of edit strings of length 100, 200, ...,1000, and find their edit distances. Plot the average time taken to compute their edit distances as a function of the length of strings. Are the empirical running times consistent with what you expect from theory? Comment on the performance of your algorithm.

2. (1 point) Give an *O(mn)* algorithm for finding the longest common substring of two input strings of length *m* and *n*. For example if the two inputs are 'Philanthropic" and "Misanthropist," the output should be "anthropi."

How do you use this algorithm to find the longest palindrome substring (a substring that reads the same from right to left as from left to right) of an input string?

3. (1 point) BigBucks wants to open a set of coffee shops in the I-5 corridor. The possible locations are at miles d1,...,dn in a straight line to the south of their Headquarters. The potential profits are given by p1...pn. The only constraint is that the distance between any two shops must be at least k (a positive integer).

- Construct a counterexample to show that a greedy algorithm that chooses in the order of profits could miss the optimal (most profitable) solution.
- Give an efficient dynamic programming based algorithm to maximize the profit.

4. (1 point) In a rope cutting problem, cutting a rope of length *n* into two pieces costs *n* time units, regardless of the location of the cut. You are given m desired locations of the cuts, X1, ..., Xm. Give a dynamic programming-based algorithm to find the optimal sequence of cuts to cut the rope into m+1 pieces to minimize the total cost.

Hint: Let X0 and X[m+1] be the two ends of the rope. Write a Bellman equation for Cost[Xi,Xj] which represents the minimum cost of cutting the part of the rope from location Xi to location Xj into j-i pieces in between.

How to submit?

Label the source file 'hw7.py' and the report report7.pdf. Submit the report on Canvas and the source at the teach interface.