Jonathan Koning
koningj@oregonstate.edu

HW 7

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."

```
def longsub(str1, str2):

    longstr = ""

    i = 0

    k = 0

    while i <len(str1):

        idxs = [idx for idx, value in enumerate(str2) if value==str1[i]]

        for idx in idxs:

            k=0

            tempstr = ""

            while i+k < len(str1) and k+idx < len(str2):

                if(str1[i+k] == str2[k+idx]):

                    tempstr+= str1[i+k]

                    k+=1

                else:

                    break

            if(len(tempstr) > len(longstr)):

                longstr = tempstr

        i+=k
```

This is O(nm) because str1 is only being iterated through once which accounts for the n. We then iterate through str2 for every instance of the checked string. This makes the overall complexity

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.