# Homework 7

Due Oct. 21st, 2021

**Note** This homework assignment covers dynamic programming from Kleinberg and Tardos.

## 1 Graded Problems

1. Given a non-empty string $s$ and a dictionary containing a list of unique words, design a dynamic programming algorithm to determine if $s$ can be segmented into a space-separated sequence of one or more dictionary words. If $s$="algorithmdesign" and your dictionary contains "algorithm" and "design". Your algorithm should answer Yes as $s$ can be segmented as "algorithmdesign".

2. Solve Kleinberg and Tardos, Chapter 6, Exercise 10.

3. Solve Kleinberg and Tardos, Chapter 6, Exercise 24.

4. You are given an $m \times n$ binary matrix. Each cell either contains a "0" or a "1". Find the largest square containing only 1's and return its area. Can you improve your solution to use $O(n + m)$ space?

## 2 Practice Problems

5. Given a string $s$ and an integer $k$, find out if it can be transformed into a palindrome by removing at most $k$ characters from it.

(a) Let's first try to come up with a recursive solution *validPalindrome()*. Consider the bite sized question, "if I can remove $\leq K$ characters, can I make the string between indices $i$ and $j$ a palindrome?" Create a recursive solution that answers this bite sized question.

(b) Your *validPalindrome()* function likely takes 3 numbers as parameters: the starting index, ending index, and the remaining amount of characters you are allowed to remove. Think about how this solution would convert to a dynamic programming solution. You may be tempted to use a 3D array to save each result:

```
int answers[/*start index*/][/*end index*/][/*# of removable chars*/]
```

If you iterate over all three indices in order to fill up the array, the runtime and space complexity of your solution becomes $O(n^2k)$. This is very not great. Let's use a different bite sized question: "HOW MANY characters do I have to remove in order to make the string between indices $i$ and $j$ a palindrome?" Create a recursive solution that answers this bite sized question.

(c) If you were to save the solutions in an array, the array would look like:

```
int answers[/*start index*/][/*end index*/]
```

This drastically cuts down on the space and time used by your solution. Now consider the order that you would need to fill up this array based on your recursive solution. Once you figure that out, write out your final solution. What is the new runtime and space complexity?