DP#1: Given two strings $x = x_1 x_2 \ldots x_m$ and $y = y_1 y_2 \ldots y_n$ find their longest common substring, that is the largest k for which there are indices i and j with.

$$\underbrace{x_i x_{i+1} \ldots x_{i+k-1}}_{\text{substring of length } k} = \underbrace{y_j y_{j+1} \ldots y_{j+k-1}}_{\text{substring of length } k}$$

Show how to do it in $O(mn)$ time.

Ans: Define $L(i,j)$ to be the length of the longest common substring ending at position x_i in x and ending at position y_j in y.

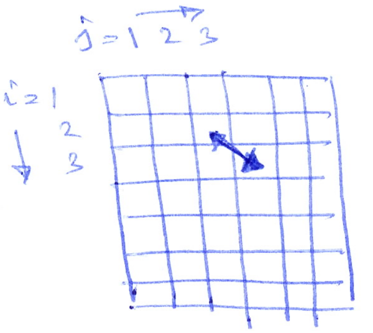
Note that $L(1,1) = 1$ if $x_1 = y_1$ and $L(1,1) = 0$ otherwise.

Infact, the recurrence relation is

$$L(i,j) = \begin{cases} 1 + L(i-1, j-1) & \text{if } x_i = y_j \\ 0 & \text{if } x_i \neq y_j \end{cases}$$

Note the difference between this problem and longest common subsequence problem. Here the optimal solution is not $L(m,n)$ but can be any where in the matrix L.

Since $L(i,j)$ depends only on $L(i-1,j-1)$, we can fill up this matrix rowwise or columnwise.



Here is the pseudocode

```
max = 0
for j = 0 to n
    L(0,j) = 0
for i = 1 to m
    L(i,0) = 0
    for j = 1 to n
        if $x_i = y_j$
            $L(i,j) = L(i-1,j-1)+1$
            if $L(i,j) > max$ then $max = L(i,j)$
        else
            $L(i,j) = 0$
return max
```

Two nested for loops
Running time $\Theta(mn)$

DP #2: A subsequence of a sequence is palindrome if it is the same whether read left to right or right to left. Give an algorithm that takes a sequence $x = x_1 x_2 \dots x_n$ and returns the length of the longest palindromic subsequence. Its running time should be $O(n^2)$.

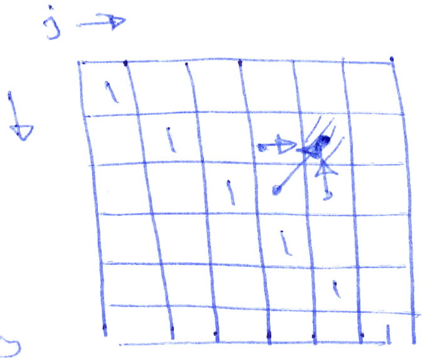
Ans: Let $c(i,j)$ be the length of the longest palindromic ~~subseq~~ subsequence within the sequence $x_i x_{i+1} \dots x_{j-1} x_j$ that is starting at position i and ending at position j

Clearly, the recurrence relation is

$$c(i,j) = \begin{cases} c(i+1, j-1) + 2 & \text{if } x_i = x_j \\ \max\{c(i+1,j), c(i,j-1)\} & \text{if } x_i \neq x_j \end{cases}$$

The base case is $c[i,i] = 1$ for all i.

Diagonals of this c matrix are known. From the recurrence relation of $c(i,j)$ it is clear that we have to fill up the matrix by filling the cells which are parallel to the diagonal.



So the pseudo code will be very similar to matrix multiplication. Here is one way to do it.

Pseudocode:

```
k = 1
for l = 2 to n
    for i = 1 to n - (l - 1)
        j = i + k
        if x_i = x_j
            if k = 1
                c(i,j) = 2
            else c(i,j) = c(i+1, j-1) + 2
        else
            c(i,j) = max{ c(i+1, j), c(i, j-1) }
    k = k + 1
Return c(1, n).
```

this check is to ensure that when $i+1 < j-1$ subsequence $x_{i+1} \dots x_{j-1}$ is meaningless.

Two nested for loops. Running time $\Theta(n^2)$