CSE525 Monsoon 2020 — Ashima Garg (PhD19003)
Homework 9 Problem 1 — Pramil Panjawani (PhD19008)

You CANNOT consult any other person or online resource for solving the homework problems. You can definitely ask the instructor or TAs for hints and you are encourage to do so (in fact, you will get useful hints if you ask for help at least 1-2 days before the due date). If we find you guilty of academic dishonesty, penalty will be imposed as per institute guidelines.

Solution:

- **Problem Definition**: Given sequences $X[1\dots i]$, and $X[1\dots j]$, Disjoint(i,j,l) = whether both occurs as disjoint subsequences of $Y[1...l]$.
- **Formula for DP**: Case 1: $Disjoint(0,0,\text{l}) = True\ \forall l = 1\dots n$, i.e. when both the subsequences to be matched are empty and therefore, empty strings can be found in the main string. Case 2: $Disjoint(i,j,l) = False\ \forall i+j > l$, i.e. the total length of the subsequences to be searched is more than the main sequence, and since we want disjoint subsequences, the length of sequence Y should be atleast sum of the two sequences. Case 3: If the last character of both the sequences $X[1...i]$ and $X[1\dots j]$ matches the last character of sequence $Y[l...l]$, this character can be considered in either of the two sequences to be matched. So, we take both the cases and therefore, $Disjoint(i,j,l) = (Disjoint(i-1,j,l-1)$ or $Disjoint[i,j-1,l-1])$ Case 4: If the last character matches with only first sequence, then $Disjoint(i,j,l) = (Disjoint[i-1,j,l-1]$. Case 5: If the last character matches with only second sequence, then $Disjoint(i,j,l) = (Disjoint(i,j-1,l-1)$. Case 6: If the last character does not occur in both the sequences, then we simply skip that character and $Disjoint(i,j,l) = Disjoint(i,j,l-1)$.
- **Memoization Structure**: Use 3D array $DP[0\dots k][0\dots k][0\dots n]$. $DP[i][j][l]$ stores the Disjoint(i, j, l) i.e. whether the sequences $X[1\dots i]$ and $X[1\dots j]$ occur as disjoint sequences of the sequence $Y[1\dots l]$.
- **How to fill the memo**: Initialize $DP[0][0][l] = True,\ \forall l = 1\dots n$. For every character in main sequence $Y[1\dots l]$, consider all the characters of both $X[1\dots i]$ and $X[1\dots j]$. Outer loop (loop 1): for $l = 0\dots n$, (loop 2) for $i = 0\dots k$, (loop 3) for $j = 0\dots k$, compute $DP[i][j][l] = Disjoint(i,j,l)$ using the recursive formula on the sequences.
- **How to solve original problem from memo**: Twice disjoint occurences of sequence X as subsequences of $Y = DP[k][k][n]$. Since $DP[i][j][l]$ stores the occurrences of $X[1\dots i]$ and $X[1\dots j]$ as subsequences in $Y[1\dots l]$. Therefore, $DP[k][k][n]$ stores the occurences of $X[1\dots k]$ and $X[1\dots k]$ (i.e. twice occurence of X) as subsequences in $Y[1\dots n]$. Therefore, it represents twice occurrences of sequence X of length k as subsequences in sequence Y.
- **Space and time complexity**: Time complexity for outer loop (loop 1)$(l = 0\dots n)$ $O(n)$, then for inner loop (loop 2)$(i = 0\dots k)$ 1 its O(n)*O(k) and again for inner most loop (loop 3) $(j = 0\dots k)$ it is O(n)*O(k)*O(k).
  Total $T(n) = O(n) * O(k) * O(k) + C$(for all the comparisons)
  $T(n) = O(nk^2)$
  We create a 3D memoization array $DP[i][j][k]$ so space complexity is $O(n^3)$.

```
def Disjoint(X[1...k], Y[1...n]):
    create empty DP[k][k][n].
    for l in range(0,n):
        for i in range(0,k):
            for j in range(0,k):
                if (i == 0 and j == 0): DP[i][j][l]=True ⟨⟨Case 1: Empty sequence is to be searched.⟩⟩
                else if(i + j > l): DP[i][j][l]=False ⟨⟨Case 2: When sum of the two length is less than length of Y⟩⟩
                else if(X[i] == Y[l] and X[j] == Y[l]): ⟨⟨Case 3⟩⟩
                    DP[i][j][l]=(DP[i-1][j][l-1] or DP[i][j-1][l-1])
                else if(X[i] == Y[l]): ⟨⟨Case 4⟩⟩
                    DP[i][j][l] =DP[i-1][j][l-1]
                else if(X[j] == Y[l]): ⟨⟨Case 5⟩⟩
                    DP[i][j][l] =DP[i][j-1][l-1]
                else: DP[i][j][l] =DP[i][j][l-1] ⟨⟨Case 6: If the last character doesn't match⟩⟩
    return DP[k][k][n]
```

■