ASSIGNMENT - 04
COMP 544 - ALGORITHM
ALGORITHM ASSIGNMENT GROUP - 03

Shuffle Algorithm Assignment:

1. Sub-Problems

   Given x, y, and z, we can ask whether z is a shuffle of x and y. For this to be true, the last character of z must be equal either to the last character of x or to the last character of z, and the remaining characters of z must be a shuffle of the remaining characters in x and y.

   Base Cases:
   If x is empty, then z must be equal to y.
   If y is empty, then z must be equal to x

2. Recurrence Relation

   Recurrence relation can be formed as follows:

$$S(i,j) = \begin{cases} \text{True} & \text{if } i = j = 0 \\ \text{True} & \text{if } i > 0 \text{ and } u_i = w_{i+j} \text{ and } S(i-1,j) = \text{True} \\ \text{True} & \text{if } j > 0 \text{ and } v_j = w_{i+j} \text{ and } S(i,j-1) = \text{True} \\ \text{False} & \text{otherwise} \end{cases}$$

3. Pseudocode

```
function isShuffle(x, y, z)
n <- length(x)
m <- length(y)
r <- length(z)
```

```
if r != n + m
    return "r must be equal to lengths of x and y. otherwise z cannot
contain x and y"

if m == 0:
    return x == r

if n == 0:
    return y == r

string x1 = the first n-1 characters in x
string y1 = the first m-1 characters in y
string z1 = the first r-1 characters in z

return ( (z[r-1] == x[n-1] && isShuffle(x1, y, z1))
|| (z[r-1] == y[m-1] && isShuffle(x, y1, z1)) );
```

4. Example

Given x, y, and z, we can ask whether z is a shuffle of x and y. For this to be true, the last character of z must be equal either to the last character of x or to the last character of z, and the remaining characters of z must be a shuffle of the remaining characters in x and y. There are also base cases where x or y is empty. If x is the empty string, then z must be equal to y; if y is empty, then z must be equal to x

Example-1:

For example,

x = “aab”

y = “axy”

w = “aaxaby”

For the above test case, w[0] can be matched with x[0] and w[1] can be matched with y[0], and w[2] must be matched with y[1], and w[3] must be matched with x[1], and w[4] and w[5] must be matched with x[2] and y[2] respectively. So, there is at least one way of forming w from x and y, and hence this is True.

Example-2:

For example,

x = “aab”
y = “axy”
w = “abaaxy”

For the above test case, w[0] can be matched with either x[0] or y[0] and w[1] cannot be matched with either x[0] or x[1] or y[0] or y[1]. So, we “w” cannot be formed with x and y; hence, this is False.

5. Python Code
   https://drive.google.com/file/d/1qnCHx5KBjfqXUUmRxe6tKPrMLvc2FFV-/view?usp=sharing