Tabulation

```
bool isInterleave ( string s1, string s2, string s3) {
    int n1 = s1.length(), n2 = s2.length(), n3 = s3.length();
    if (n1 + n2 != n3) return false;
    bool dp[101][101] = {};
    dp[0][0] = true;
    for (int i=0; i<n1; i++) {
        for (int j=0; j<n2; j++) {
            if (i>0 && s1[i-1] == s3[i+j-1]) {
                dp[i][j] = dp[i-1][j];
            }
            if (j>0 && s2[j-1] == s3[i+j-1]) {
                dp[i][j] = dp[i][j] || dp[i][j-1];
            }
        }
    }
    return dp[n1][n2];
```

(1) (2) (1)

(2)

ans = ans + (3-1)

(2) 1

→ (2)

Interleaving

s and t
↓ ↓
n m substring.

$s_1 + s_2 + s_3 - \dots + s_n$ $(t_1 + t_2 + t_3 + \dots + t_m)$

$n - m \mid \leq 1$

$s_1 + t_1 + s_2 + t_2 + \dots$ or $t_1 + s_1 + t_2 + s_2 + t_3 + s_3 + \dots$

$(a + b)$ → concatenation

aabcc dbbca.

a	ab
aa	abb
aab	abcc
aabc	
aabcc	

aa bcc

dbbca

Naive

aadbbcbcac.

```
Bool recur-(string &s1, string &s2, string &s3, int i, int j, int k,
            string &s3){
    if(k == s3.length()){
        return true;
    }
Bool a = false;  Bool b = false;
if( i < s1.length() && s1[i] == s3[k]){
    a = recur-(s1, s2, s3, i+1, j, k);
}
if(j < s2.length() && s2[j] == s3[k]){
        b = recur-(s1, s2, s3, i, j+1, k+1);
}
return a||b;
}
```

Replace
k = i + j
and apply 2D Memorization