<https://leetcode.com/problems/shortest-common-supersequence/description/>

Given two strings str1 and str2, return *the shortest string that has both* str1 *and* str2 *as* ***subsequences***. If there are multiple valid strings, return **any** of them.

A string s is a **subsequence** of string t if deleting some number of characters from t (possibly 0) results in the string s.

**Example 1:**

Input: str1 = "abac", str2 = "cab"

Output: "cabac"

Explanation:

str1 = "abac" is a subsequence of "cabac" because we can delete the first "c".

str2 = "cab" is a subsequence of "cabac" because we can delete the last "ac".

The answer provided is the shortest such string that satisfies these properties.

**Example 2:**

Input: str1 = "aaaaaaaa", str2 = "aaaaaaaa"

Output: "aaaaaaaa"

**Constraints:**

* 1 <= str1.length, str2.length <= 1000
* str1 and str2 consist of lowercase English letters.

**Attempt 1: 2023-05-26**

**Wrong Solution:**

**Mismatch the expected result because of we change the order of characters of original Str1 and Str2**

Input:

str1 = "abac"

str2 = "cab"

Output: "acaab"

Expected: "cabac"

=========================================================================

class Solution {

public String shortestCommonSupersequence(String str1, String str2) {

String result = "";

int m = str1.length();

int n = str2.length();

String lcs = findLongestCommonSubsequence(str1, str2);

int len = lcs.length();

if(len == 0) {

return str1 + str2;

}

int i = 0;

int j = 0;

int k = 0;

while(i < m && k < len) {

if(str1.charAt(i) != lcs.charAt(k)) {

result += str1.charAt(i);

i++;

}

k++;

}

k = 0;

while(j < n && k < len) {

if(str2.charAt(j) != lcs.charAt(k)) {

result += str2.charAt(j);

j++;

}

k++;

}

result += lcs;

return result;

}

private String findLongestCommonSubsequence(String s1, String s2) {

int m = s1.length();

int n = s2.length();

// Instead of int[][] dp in L1143.Longest Common Subsequence,

// we have to create String[][] dp for actual common subsequence storage

String[][] dp = new String[m + 1][n + 1];

for(int i = 0; i < m; i++) {

Arrays.fill(dp[i], "");

}

for(int i = 1; i <= m; i++) {

for(int j = 1; j <= n; j++) {

if(s1.charAt(i - 1) == s2.charAt(j - 1)) {

dp[i][j] = dp[i - 1][j - 1] + s1.charAt(i - 1);

} else {

dp[i][j] = dp[i - 1][j].length() > dp[i][j - 1].length() ? dp[i - 1][j] : dp[i][j - 1];

}

}

}

return dp[m][n];

}

}

**Solution 1: Based on L1143. Longest Common Subsequence (10 min)**

class Solution {

public String shortestCommonSupersequence(String str1, String str2) {

int m = str1.length();

int n = str2.length();

String lcs = findLongestCommonSubsequence(str1, str2);

int len = lcs.length();

if(len == 0) {

return str1 + str2;

}

StringBuilder sb = new StringBuilder();

int i = 0;

int j = 0;

int k = 0;

while(k < len) {

while(i < m && str1.charAt(i) != lcs.charAt(k)) {

sb.append(str1.charAt(i));

i++;

}

while(j < n && str2.charAt(j) != lcs.charAt(k)) {

sb.append(str2.charAt(j));

j++;

}

sb.append(lcs.charAt(k));

i++;

j++;

k++;

}

sb.append(str1.substring(i)).append(str2.substring(j));

return sb.toString();

}

private String findLongestCommonSubsequence(String s1, String s2) {

int m = s1.length();

int n = s2.length();

// Instead of int[][] dp in L1143.Longest Common Subsequence,

// we have to create String[][] dp for actual common subsequence storage

String[][] dp = new String[m + 1][n + 1];

for(int i = 0; i <= m; i++) {

Arrays.fill(dp[i], "");

}

for(int i = 1; i <= m; i++) {

for(int j = 1; j <= n; j++) {

if(s1.charAt(i - 1) == s2.charAt(j - 1)) {

dp[i][j] = dp[i - 1][j - 1] + s1.charAt(i - 1);

} else {

dp[i][j] = dp[i - 1][j].length() > dp[i][j - 1].length() ? dp[i - 1][j] : dp[i][j - 1];

}

}

}

return dp[m][n];

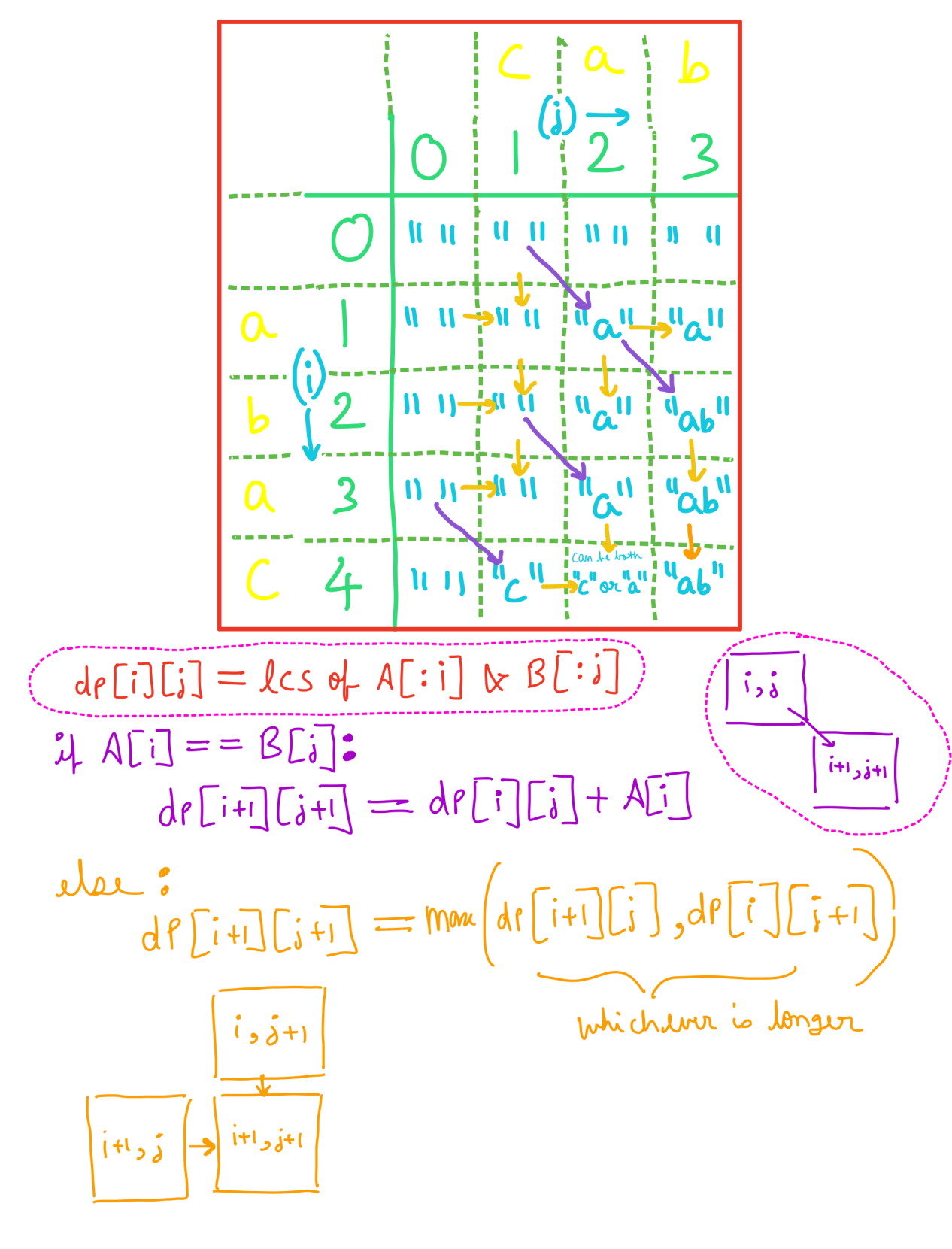
}

}

**Refer to**

<https://leetcode.com/problems/shortest-common-supersequence/solutions/312702/java-dp-solution-similiar-to-lcs/comments/701773>

Here is a diagram of the given test case for finding the lcs.



<https://leetcode.com/problems/shortest-common-supersequence/solutions/312710/c-python-find-the-lcs/comments/290904>

public String shortestCommonSupersequence(String str1, String str2) {

String lcs = longestCommonSubSeq(str1, str2);

int p1 = 0, p2 = 0;

StringBuilder sb = new StringBuilder();

for (int i = 0; i < lcs.length(); i++) {

while(p1 < str1.length() && str1.charAt(p1) != lcs.charAt(i)) {

sb.append(str1.charAt(p1++));

}

while(p2 < str2.length() && str2.charAt(p2) != lcs.charAt(i)) {

sb.append(str2.charAt(p2++));

}

sb.append(lcs.charAt(i));

p1++;

p2++;

}

sb.append(str1.substring(p1)).append(str2.substring(p2));

return sb.toString();

}

private String longestCommonSubSeq(String str1, String str2) {

String[][] dp = new String[str1.length() + 1][str2.length() + 1];

for (int i = 0; i < dp.length; i++) {

Arrays.fill(dp[i], "");

}

for (int i = 1; i <= str1.length(); i++) {

for (int j = 1; j <= str2.length(); j++) {

if (str1.charAt(i - 1) == str2.charAt(j - 1)) {

dp[i][j] = dp[i - 1][j - 1] + str1.charAt(i - 1);

} else {

dp[i][j] = dp[i - 1][j].length() > dp[i][j - 1].length() ? dp[i - 1][j] : dp[i][j - 1];

}

}

}

return dp[str1.length()][str2.length()];

}