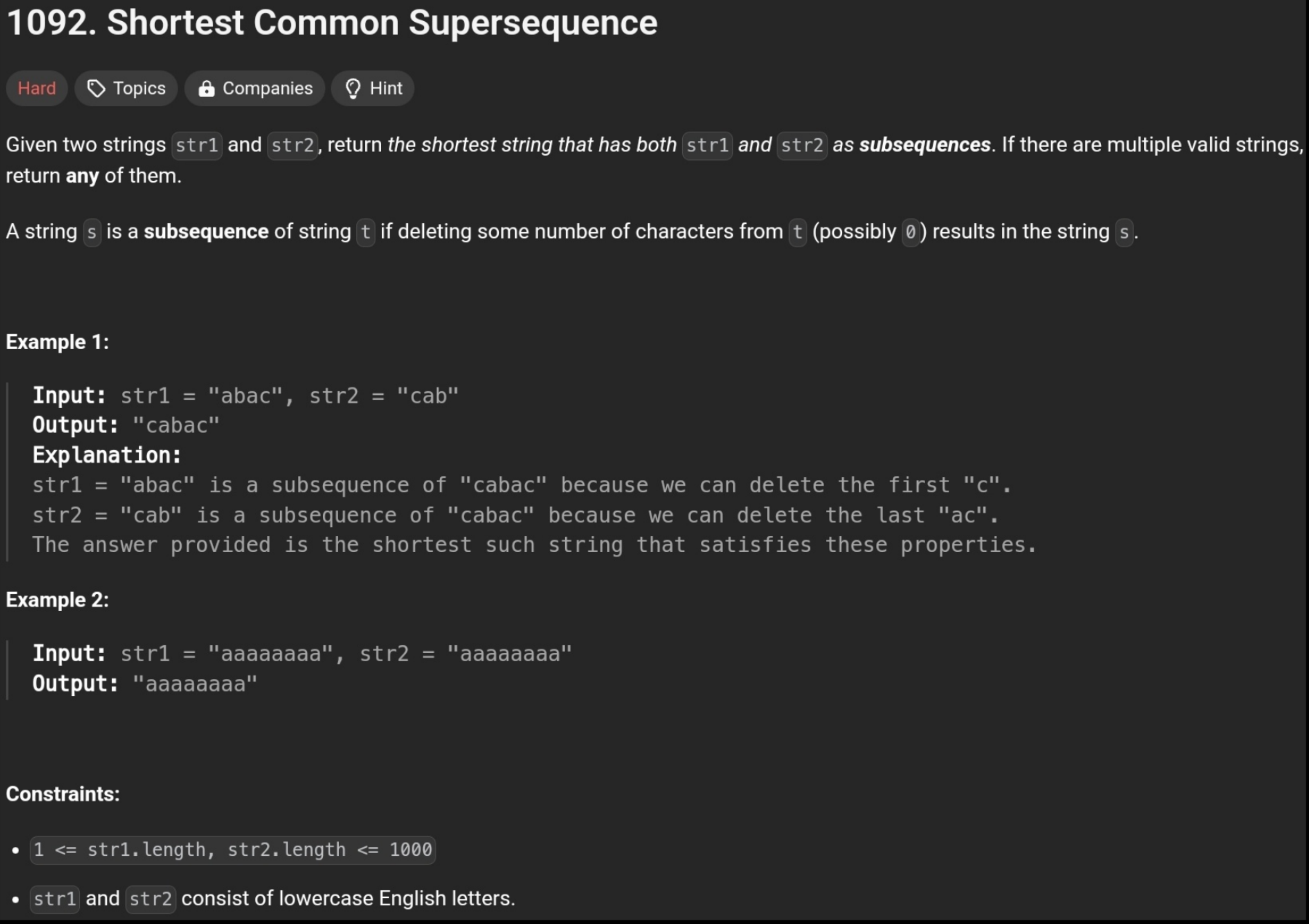


1092. Shortest Common Supersequence

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.

S_1 = "groot" S_2 = "brute"

To get the shortest Supersequence, we should consider the same characters of S_1 and S_2 only once. So first we get the LCS

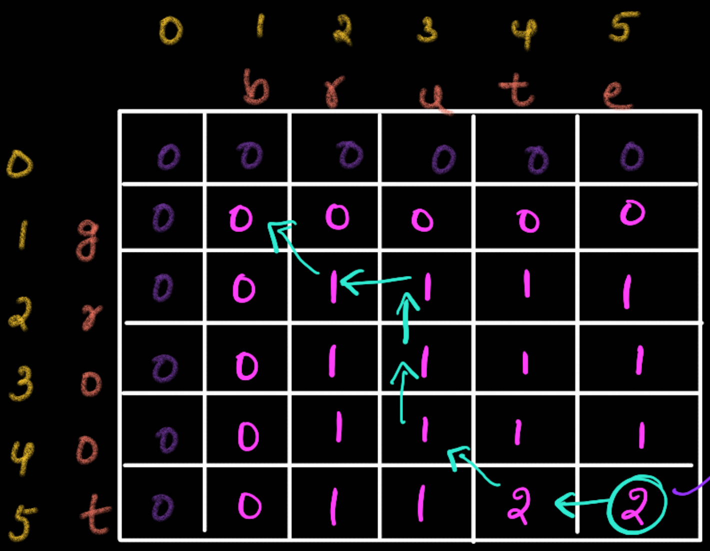
LCS = "rt"

length of shortest SuperSequence

= m + n − LCS

i.e. remove one occurrence of LCS characters bcoz they are counted twice in m and n.

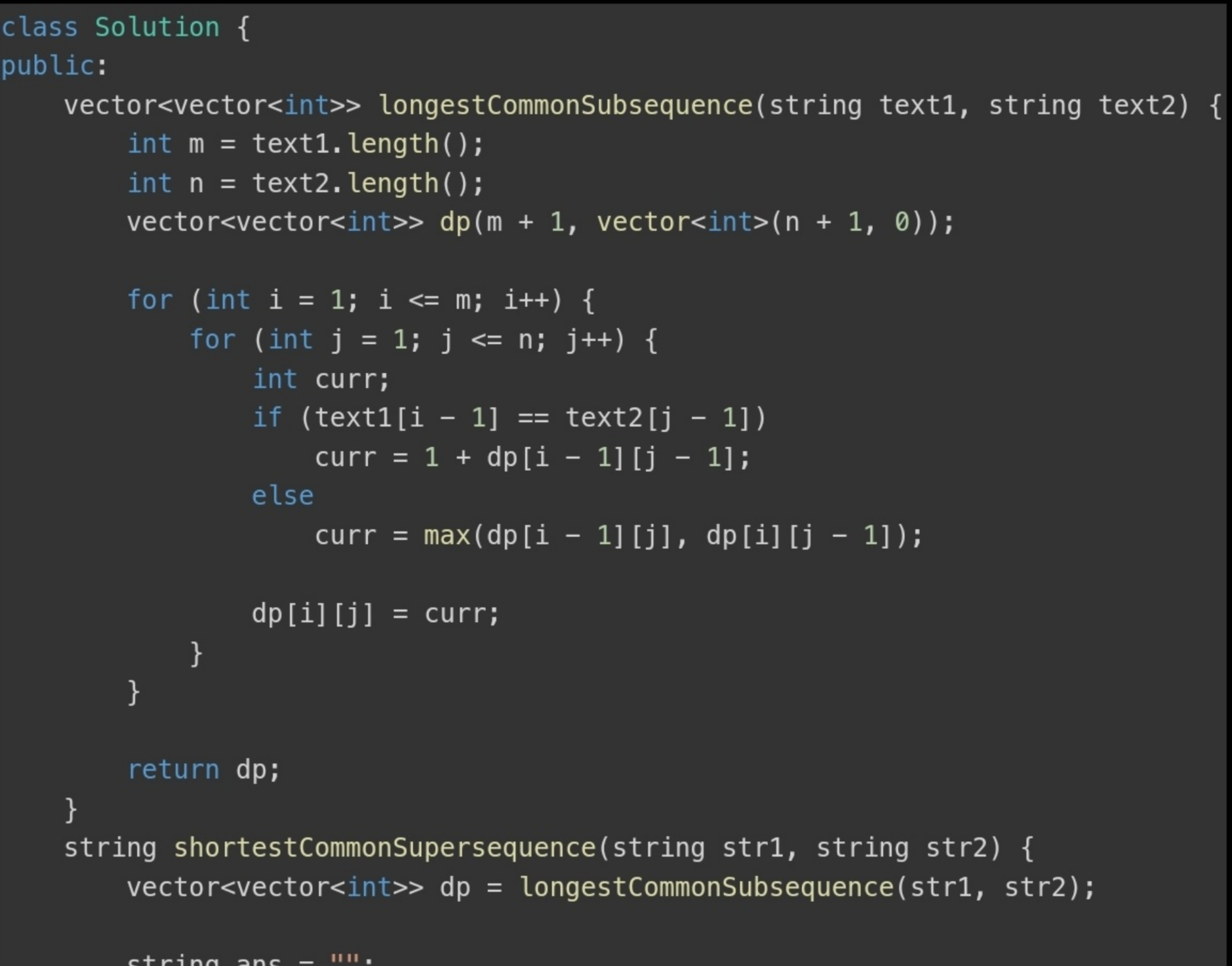
To get the string, we can use the dp table constructed in finding LCS constructed in finding LCS:



$$\begin{array}{c|c|cccccc}
 & & 0 & 1 & 2 & 3 & 4 & 5 \\
 & & & b & r & u & t & e \\
\hline
0 & & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & g & 0 & 0 & 0 & 0 & 0 & 0 \\
2 & r & 0 & 0 & 1 & 1 & 1 & 1 \\
3 & o & 0 & 0 & 1 & 1 & 1 & 1 \\
4 & o & 0 & 0 & 1 & 1 & 1 & 1 \\
5 & t & 0 & 0 & 1 & 1 & 2 & \boxed{2}
\end{array}$$

you are moving left means you are leaving a char of 'brute' so add it to ans.

ans = "gbruoote"



```
class Solution {
public:
    vector<vector<int>> longestCommonSubsequence(string text1, string text2) {
        int m = text1.length();
        int n = text2.length();
        vector<vector<int>> dp(m + 1, vector<int>(n + 1, 0));

        for (int i = 1; i <= m; i++) {
            for (int j = 1; j <= n; j++) {
                int curr;
                if (text1[i - 1] == text2[j - 1])
                    curr = 1 + dp[i - 1][j - 1];
                else
                    curr = max(dp[i - 1][j], dp[i][j - 1]);

                dp[i][j] = curr;
            }
        }

        return dp;
    }
    string shortestCommonSupersequence(string str1, string str2) {
        vector<vector<int>> dp = longestCommonSubsequence(str1, str2);

        string ans = "";
        string ans = "";

        int i = str1.length();
        int j = str2.length();

        while (i > 0 && j > 0) {
            if (str1[i - 1] == str2[j - 1]) {   ↖
                ans = str1[i - 1] + ans;
                i--, j--;
            } else if (dp[i - 1][j] >= dp[i][j - 1]) {   ↑
                ans = str1[i - 1] + ans;
                i--;
            } else {   ←
                ans = str2[j - 1] + ans;
                j--;
            }
        }

        while (i > 0) {
            ans = str1[i - 1] + ans;
            i--;
        }
        while (j > 0) {
            ans = str2[j - 1] + ans;
            j--;
        }

        return ans;
    }
};
```

$T(n)$: $T(n)$ of LCS $+$

$O(m+n)$

$S(n)$: $S(n)$ of LCS