Shortest Common Supersequence

Given two strings str1 and str2, find the shortest string that has both str1 and str2 as subsequences.

Examples:

```
Input:   str1 = "geek",  str2 = "eke"
Output: "geeke"

Input:   str1 = "AGGTAB",  str2 = "GXTXAYB"
Output:  "AGXGTXAYB"
```

We strongly recommend you to minimize your browser and try this yourself first.

This problem is closely related to longest common subsequence problem. Below are steps.

1) Find Longest Common Subsequence (lcs) of two given strings. For example, lcs of “geek” and “eke” is “ek”.

2) Insert non-lcs characters (in their original order in strings) to the lcs found above, and return the result. So “ek” becomes “geeke” which is shortest common supersequence.

Let us consider another example, str1 = “AGGTAB” and str2 = “GXTXAYB”. LCS of str1 and str2 is “GTAB”. Once we find LCS, we insert characters of both strings in order and we get “AGXGTXAYB”

How does this work?

We need to find a string that has both strings as subsequences and is shortest such string. If both strings have all characters different, then result is sum of lengths of two given strings. If there are common characters, then we don’t want them multiple times as the task is to minimize length. Therefore, we fist find the longest common subsequence, take one occurrence of this subsequence and add extra characters.

```
Length of the shortest supersequence  = (Sum of lengths of given two strings) -
                                        (Length of LCS of two given strings)
```

Below is C implementation of above idea. The below implementation only finds length of the shortest supersequence.

```
/* C program to find length of the shortest supersequence */
#include<stdio.h>
#include<string.h>

/* Utility function to get max of 2 integers */
int max(int a, int b) { return (a > b)? a : b; }

/* Returns length of LCS for X[0..m-1], Y[0..n-1] */
int lcs( char *X, char *Y, int m, int n);

// Function to find length of the shortest supersequence
// of X and Y.
int shortestSuperSequence(char *X, char *Y)
{
  int m = strlen(X), n = strlen(Y);

  int l = lcs(X, Y, m, n); // find lcs

  // Result is sum of input string lengths - length of lcs
  return (m + n - l);
}

/* Returns length of LCS for X[0..m-1], Y[0..n-1] */
int lcs( char *X, char *Y, int m, int n)
{
   int L[m+1][n+1];
   int i, j;

   /* Following steps build L[m+1][n+1] in bottom up fashion.
      Note that L[i][j] contains length of LCS of X[0..i-1]
      and Y[0..j-1] */
   for (i=0; i<=m; i++)
   {
     for (j=0; j<=n; j++)
     {
       if (i == 0 || j == 0)
         L[i][j] = 0;

       else if (X[i-1] == Y[j-1])
         L[i][j] = L[i-1][j-1] + 1;

       else
         L[i][j] = max(L[i-1][j], L[i][j-1]);
     }
   }

   /* L[m][n] contains length of LCS for X[0..n-1] and
      Y[0..m-1] */
   return L[m][n];
}

/* Driver program to test above function */
int main()
{
  char X[] = "AGGTAB";
  char Y[] = "GXTXAYB";
  printf("Length of the shortest supersequence is %d\n",
         shortestSuperSequence(X, Y));
  return 0;
}
```

Run on IDE

Output:

```
Length of the shortest supersequence is 9
```

Below is **Another Method** to solve the above problem.
A simple analysis yields below simple recursive solution.

```
Let X[0..m-1] and Y[0..n-1] be two strings and m and be respective
lengths.

  if (m == 0) return n;
  if (n == 0) return m;

  // If last characters are same, then add 1 to result and
  // recur for X[]
  if (X[m-1] == Y[n-1])
     return 1 + SCS(X, Y, m-1, n-1);

  // Else find shortest of following two
  //  a) Remove last character from X and recur
  //  b) Remove last character from Y and recur
  else return 1 + min( SCS(X, Y, m-1, n), SCS(X, Y, m, n-1) );
```

Below is simple naive recursive solution based on above recursive formula.

```
/* A Naive recursive C++ program to find length
   of the shortest supersequence */
#include<bits/stdc++.h>
using namespace std;

int superSeq(char* X, char* Y, int m, int n)
{
    if (!m) return n;
    if (!n) return m;

    if (X[m-1] == Y[n-1])
        return 1 + superSeq(X, Y, m-1, n-1);

    return 1 + min(superSeq(X, Y, m-1, n),
                   superSeq(X, Y, m, n-1));
}

// Driver program to test above function
int main()
{
    char X[] = "AGGTAB";
    char Y[] = "GXTXAYB";
    cout << "Length of the shortest supersequence is "
         << superSeq(X, Y, strlen(X), strlen(Y));
    return 0;
}
```

Run on IDE

Output:

```
Length of the shortest supersequence is 9
```

Time complexity of the above solution exponential $O(2^{min(m, n)})$. Since there are overlapping subproblems, we can efficiently solve this recursive problem using Dynamic Programming. Below is Dynamic Programming based implementation. Time complexity of this solution is O(mn).

```
/* A dynamic programming based C program to find length
   of the shortest supersequence */
#include<bits/stdc++.h>
using namespace std;

// Returns length of the shortest supersequence of X and Y
int superSeq(char* X, char* Y, int m, int n)
{
    int dp[m+1][n+1];

    // Fill table in bottom up manner
    for (int i = 0; i <= m; i++)
    {
        for (int j = 0; j <= n; j++)
        {
           // Below steps follow above recurrence
           if (!i)
               dp[i][j] = j;
           else if (!j)
               dp[i][j] = i;
           else if (X[i-1] == Y[j-1])
                dp[i][j] = 1 + dp[i-1][j-1];
           else
                dp[i][j] = 1 + min(dp[i-1][j], dp[i][j-1]);
        }
    }

    return dp[m][n];
}

// Driver program to test above function
int main()
{
    char X[] = "AGGTAB";
    char Y[] = "GXTXAYB";
    cout << "Length of the shortest supersequence is "
         << superSeq(X, Y, strlen(X), strlen(Y));
    return 0;
}
```

Run on IDE

Output:

```
Length of the shortest supersequence is 9
```

Thanks to Gaurav Ahirwar for suggesting this solution.

Exercise:

Extend the above program to print shortest supersequence also using function to print LCS.

References:

https://en.wikipedia.org/wiki/Shortest_common_supersequence

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

GATE CS Corner Company Wise Coding Practice

Dynamic Programming LCS subsequence

Recommended Posts:

Compute sum of digits in all numbers from 1 to n
Collect maximum points in a grid using two traversals
Dynamic Programming | Set 4 (Longest Common Subsequence)
Count possible ways to construct buildings
Printing Shortest Common Supersequence

(Login to Rate and Mark)

3.1 Average Difficulty : **3.1/5.0**
Based on **34** vote(s)

Add to TODO List
Mark as DONE

Writing code in comment? Please use ide.geeksforgeeks.org, generate link and share the link here.

Share this post!