

[Suggest a Topic](#)[Login](#)[Write an Article](#)

Longest Repeated Subsequence

Given a string, print the longest repeating subsequence such that the two subsequence don't have same string character at same position, i.e., any i'th character in the two subsequences shouldn't have the same index in the original string.

Input : str = "AABEBCDD"

Output : "ABD"

The longest repeating subsequence
is "ABD"

More Examples:

Input: str = "aabb"

Output: "ab"

Input: str = "aab"

Output: "a"

The two subsequence are 'a'(first) and 'a'
(second). Note that 'b' cannot be considered
as part of subsequence as it would be at same
index in both.

Recommended: Please solve it on "[PRACTICE](#)" first, before moving on to the solution.

APNIC Thank
Intern



This problem is just the modification of [Longest Common Subsequence problem](#). The idea is to find the **LCS(str, str)** where **str** is the input string with the restriction that when both the characters are same, they shouldn't be on the same index in the two strings.

We have discussed a solution to find [length of the longest repeated subsequence](#).

C++

```
// Refer https://www.geeksforgeeks.org/longest-repeating-subsequence/
// for complete code.
// This function mainly returns LCS(str, str)
// with a condition that same characters at
// same index are not considered.
int findLongestRepeatingSubSeq(string str)
{
    int n = str.length();

    // Create and initialize DP table
    int dp[n+1][n+1];
    for (int i=0; i<=n; i++)
        for (int j=0; j<=n; j++)
            dp[i][j] = 0;

    // Fill dp table (similar to LCS loops)
    for (int i=1; i<=n; i++)
    {
        for (int j=1; j<=n; j++)
        {
            // If characters match and indexes are
            // not same
            if (str[i-1] == str[j-1] && i != j)
                dp[i][j] = 1 + dp[i-1][j-1];

            // If characters do not match
            else
                dp[i][j] = max(dp[i][j-1], dp[i-1][j]);
        }
    }
    return dp[n][n];
}
```

[Run on IDE](#)
[Copy Code](#)

Python3

```
# Python method for Longest Repeated
# Subsequence

# Refer https://www.geeksforgeeks.org/longest-repeating-subsequence/
# for complete code.
# This function mainly returns LCS(str, str)
# with a condition that same characters at
# same index are not considered.
def findLongestRepeatingSubSeq(str):
    n = len(str)

    # Create and initialize DP table
    dp = [[0 for k in range(n+1)] for l in range(n+1)]

    # Fill dp table (similar to LCS loops)
    for i in range(1, n+1):
        for j in range(1, n+1):
            # If characters match and indices are not same
            if (str[i-1] == str[j-1] and i != j):
                dp[i][j] = 1 + dp[i-1][j-1]
```

```
# If characters do not match
else:
    dp[i][j] = max(dp[i][j-1], dp[i-1][j])
```

```
return dp[n][n]
```

This code is contributed by Soumen Ghosh

[Run on IDE](#)[Copy Code](#)

How to print the subsequence?

The above solution only finds length of subsequence. We can print the subsequence using `dp[n+1][n+1]` table built. The idea is similar to [printing LCS](#).

```
// Pseudo code to find longest repeated
// subsequence using the dp[][] table filled
// above.
```

```
// Initialize result
string res = "";
```

```
// Traverse dp[][] from bottom right
```

```
i = n, j = n;
```

```
while (i > 0 && j > 0)
```

```
{
```

```
    // If this cell is same as diagonally
    // adjacent cell just above it, then
    // same characters are present at
    // str[i-1] and str[j-1]. Append any
    // of them to result.
```

```
    if (dp[i][j] == dp[i-1][j-1] + 1)
```

```
    {
```

```
        res = res + str[i-1];
```

```
        i--;
```

```
        j--;
```

```
    }
```

```
    // Otherwise we move to the side
```

```
    // that that gave us maximum result
```

```
    else if (dp[i][j] == dp[i-1][j])
```

```
        i--;
```

```
    else
```

```
        j--;
```

```
}
```

```
// Since we traverse dp[][] from bottom,
```

```
// we get result in reverse order.
```

```
reverse(res.begin(), res.end());
```

```
return res;
```

Below is implementation of above steps.

```

// C++ program to find the longest repeated
// subsequence
#include <bits/stdc++.h>
using namespace std;

// This function mainly returns LCS(str, str)
// with a condition that same characters at
// same index are not considered.
string longestRepeatedSubSeq(string str)
{
    // THIS PART OF CODE IS SAME AS BELOW POST.
    // IT FILLS dp[][]
    // https://www.geeksforgeeks.org/longest-repeating-subsequence/
    // OR the code mentioned above.
    int n = str.length();
    int dp[n+1][n+1];
    for (int i=0; i<=n; i++)
        for (int j=0; j<=n; j++)
            dp[i][j] = 0;
    for (int i=1; i<=n; i++)
        for (int j=1; j<=n; j++)
            if (str[i-1] == str[j-1] && i != j)
                dp[i][j] = 1 + dp[i-1][j-1];
            else
                dp[i][j] = max(dp[i][j-1], dp[i-1][j]);

    // THIS PART OF CODE FINDS THE RESULT STRING USING DP[][]
    // Initialize result
    string res = "";

    // Traverse dp[][] from bottom right
    int i = n, j = n;
    while (i > 0 && j > 0)
    {
        // If this cell is same as diagonally
        // adjacent cell just above it, then
        // same characters are present at
        // str[i-1] and str[j-1]. Append any
        // of them to result.
        if (dp[i][j] == dp[i-1][j-1] + 1)
        {
            res = res + str[i-1];
            i--;
            j--;
        }

        // Otherwise we move to the side
        // that that gave us maximum result
        else if (dp[i][j] == dp[i-1][j])
            i--;
        else
            j--;
    }

    // Since we traverse dp[][] from bottom,
    // we get result in reverse order.
    reverse(res.begin(), res.end());

    return res;
}

// Driver Program
int main()
{
    string str = "AABEBCDD";
    cout << longestRepeatedSubSeq(str);
    return 0;
}

```

Run on IDE

Copy Code



Python3

```

# Python3 program to find the
# longest repeated subsequence

# This function mainly returns LCS(str, str)
# with a condition that same characters
# at same index are not considered.
def longestRepeatedSubSeq(str):
    # This part of code is same as
    # below post it fills dp[][]
    # https://www.geeksforgeeks.org/longest-repeating-subsequence/
    # OR the code mentioned above
    n = len(str)
    dp = [[0 for i in range(n+1)] for j in range(n+1)]

    for i in range(1, n + 1):
        for j in range(1, n + 1):
            if (str[i-1] == str[j-1] and i != j):
                dp[i][j] = 1 + dp[i-1][j-1]
            else:
                dp[i][j] = max(dp[i][j-1], dp[i-1][j])

    # This part of code finds the result
    # string using dp[][] Initialize result
    res = ''

    # Traverse dp[][] from bottom right
    i = n
    j = n
    while (i > 0 and j > 0):
        # If this cell is same as diagonally
        # adjacent cell just above it, then
        # same characters are present at
        # str[i-1] and str[j-1]. Append any
        # of them to result.
        if (dp[i][j] == dp[i-1][j-1] + 1):
            res += str[i-1]
            i -= 1
            j -= 1

        # Otherwise we move to the side
        # that gave us maximum result.
        elif (dp[i][j] == dp[i-1][j]):
            i -= 1
        else:
            j -= 1

    # Since we traverse dp[][] from bottom,
    # we get result in reverse order.
    res = ''.join(reversed(res))

    return res

# Driver Program
str = 'AABEBCDD'
print(longestRepeatedSubSeq(str))

# This code is contributed by Soumen Ghosh

```

Run on IDE

Copy Code

Output:

ABD


Time Complexity : $O(n^2)$

Auxiliary Space : $O(n^2)$

This article is contributed by **Kartik**. If you like GeeksforGeeks and would like to contribute, you can also write an article using contribute.geeksforgeeks.org or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

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

IQ Test: What is your IQ?

 Answer 20 questions to find out! test-iq.org

Recommended Posts:

- [A Space Optimized Solution of LCS](#)
- [Longest Repeating Subsequence](#)
- [Program for nth Catalan Number](#)
- [Longest Common Substring | DP-29](#)
- [Cutting a Rod | DP-13](#)
- [Longest Palindromic Subsequence | DP-12](#)
- [0-1 Knapsack Problem | DP-10](#)
- [Coin Change | DP-7](#)
- [Min Cost Path | DP-6](#)
- [Edit Distance | DP-5](#)
- [Longest Common Subsequence | DP-4](#)
- [Longest Increasing Subsequence | DP-3](#)
- [Maximum size square sub-matrix with all 1s](#)
- [Ugly Numbers](#)
- [Largest Sum Contiguous Subarray](#)

Article Tags : [Dynamic Programming](#) [LCS](#)

Practice Tags : [Dynamic Programming](#) [LCS](#)