

LCS(Longest common subsequence).

Problem Statement

Given two strings S1 and S2.

The Longest Common Subsequence (LCS) problem is to find the length of the longest string which is subsequence of both strings S1 and S2.

Input: S1=BCDAACD

S2=ACDBAC

Output: Length of LCS = 4

The longest common subsequence is CDAC

- Given $S1=BCDAACD$ and $S2=ACDBAC$ be two sequences.
- Then, common Subsequences are $BC, CDAC, DAC, AAC, AC, CD, \dots$
- Among these subsequences $CDAC$ is the longest common subsequence with $Length=4$.
- Let us try to find this longest common subsequence using dynamic programming.

Recursive Code of LCS

```
1  #include<bits/stdc++.h>
2  using namespace std;
3  string S1,S2;
4  int dp[1000][1000];
5  int LCS(int i,int j)
6  {
7
8      if(i==0|| j==0)
9      {
10         return 0;
11     }
12     if(dp[i][j]!=-1)return dp[i][j];
13     if(S1[i-1]==S2[j-1])
14     {
15         return dp[i][j]=1+LCS(i-1,j-1);
16     }
17     else
18     {
19         return dp[i][j]=max(LCS(i,j-1),LCS(i-1,j));
20     }
21
22
23     int main()
24     {
25         cin>>S1>>S2;
26         memset(dp, -1, sizeof(dp));
27         cout<<LCS(S1.size(),S2.size());|
28     }
29
30
```

Iterative Code of LCS

```
31
32
33 int LCS_iterative()
34 {
35
36     int n=S1.size();
37     int m=S2.size();
38
39     for(int i=0;i<=n;i++)
40     {
41         for(int j=0;j<=m;j++)
42         {
43             if(i==0 || j==0)
44                 dp[i][j]=0;
45             else if(S1[i-1]==S2[j-1])
46             {
47                 dp[i][j]=dp[i-1][j-1]+1;
48             }
49             else
50                 dp[i][j]=max(dp[i-1][j],dp[i][j-1]);
51         }
52     }
53     return dp[n][m];
54 }
55
56 int main()
57 {
58     cin>>S1>>S2;
59
60     cout<<LCS_iterative()<<endl;
61 }
62
63
```

How LCS can be printed?

	\$	A	C	D	B	A	C
\$	0	0	0	0	0	0	0
B	0	0	0	0	1	1	1
C	0	0	1	1	1	1	2
D	0	0	1	2	2	2	2
A	0	1	1	2	2	3	3
A	0	1	1	2	2	3	3
C	0	1	2	2	2	3	4
D	0	1	2	3	3	3	4

cont.....

- Time complexity of both solution?
- LCS with better space complexity. Think in terms of linear space.