Crack-a-Hack

Cost of Merging strings.

Name: Akhila Joshi.

Usn:01FE15BCS017.

Problem: Github reconciles multiple changes made to a collection of files using the Longest Common Subsequence (LCS). At a basic level, this operation boils down to merging a pair of strings if the length of their LCS is at least k. For e.g. the strings "apple" and "pie" have LCS "pe" for k=2. You are given two strings x & y with the goal of performing the merge for a given k, by altering some characters in x with minimum cost.

For e.g. let's say you have "abble" and "pie" with k=2. You can change "abble" to "abple" to achieve the result.

However, there's a cost in making a change to a character from one to another, defined by f(originalcharacter)^f(newcharacter) where ('a') = 0, ('b') = 1, ('c') = 2, .. ('z') = 25. For e.g. if you changed 'a' to 'z', it would cost you and denotes the xor operation.

Approach:

1.Modification to Longest Common Subsequence.

2.Identify the constraints :

1. if k>min(n,m), then it's impossible to attain LCS of atleast length k.

2. Let dp[n][m][k] store the minimum cost to achieve LCS of length k, in x[0..i] and y[0..j].

3. dp[n][m][0] = 0 , because we can achieve LCS of length 0 with 0 cost.

4. For all i < 0 or j < 0, dp[n][m][k] = infinity; (We can never achieve LCS of length p > 0 in such case).

3.Solution

Else there are 3 cases

1. Convert x[i] to y[j]

2.Skip i th character from x.

3. Skip j th character from y.

2.If you convert x[i] to y[j], then cost=f(x[i])^f(y[j]) will be added and LCS will decrease by 1.

3.If you skip i th character from x then i will be decreased by 1, no cost will be added and LCS will remain the same.

4.If you skip j th character from y then j will be decreased by 1, no cost will be added and LCS will remain the same.

5.Thus dp[i][j][k]=min(cost+ dp[i-1][j-1][k-1], dp[i-1][j][k], dp[i][j-1][k])

6. The minimum cost to make the length of their LCS atleast k is dp[n-1][m-1][k] .

Code:

```
#include <stdio.h>
 #include<stdlib.h>
 #define N 350
 const int MAX = 1e9;
 char x[N],y[N];
 int n,m,k; int dp[N][N][N];
  int solve(int i,int j,int k)
        {
                if(k == 0)
                        return 0;
               if(i==-1 || j == -1)
                        return MAX;
                if(dp[i][j][k]>=0)
                        return dp[i][j][k];
                int temp1 = solve(i-1,j,k);
               int temp2 = solve(i,j-1,k);
               int temp3 = solve(i-1,j-1,k-1)+((x[i]-97)^(y[j]-97));
               int ans = temp1;
                if(ans>temp2)
```

```
                ans = temp2;
        if(ans>temp3)
                ans = temp3;
        dp[i][j][k]=ans;
                return ans;
}
int main()
 {
 memset(dp,~0,sizeof(dp));
scanf("%d %d %d",&n,&m,&k);
scanf("%s",x);
scanf("%s",y);
 int answer = solve(n-1,m-1,k);
if(answer >=MAX)
        printf("-1\n");
else
        printf("%d",answer);
return 0;
}
```

Refrences:

- https://www.geeksforgeeks.org/dynamic-programming-set-5-edit-distance
- https://www.hackerrank.com/contests/adobe-codiva/challenges/cost-of-merging-strings