DP Lesson 6

1. Longest Common Subsequence
2. Longest Common Substring
3. Minimum-number-of-deletions-and-insertions
4. Minimum-cost-to-make-two-strings-identical
5. Minimum-deletitions
6. Minimum-number-of-deletions
7. Longest Repeating Subsequence
8. Is Subsequence
9. Form a palindrome | Practice | GeeksforGeeks
10. Find-number-of-times-a-string-occurs-as-a-subsequence
11. k-palindrome

Longest Common Subsequence

Given two sequences, find the length of the longest subsequence present in both of them.

Both the strings are of uppercase.

Example 1:

Input:

A = 6, B = 6

str1 = ABCDGH

str2 = AEDFHR

Output: 3

Explanation: LCS for input Sequences“ABCDGH” and “AEDFHR” is “ADH” of length 3.

Example 2:

Input:

A = 3, B = 2

str1 = ABC

str2 = AC

Output: 2

Explanation: LCS of "ABC" and "AC" is"AC" of length 2.

```cpp
vector<vector<int>> ans;
/*int LCS(int x, int y, string s1, string s2)
{
    if(x==0 || y==0)
        return 0;
    if(ans[x][y]==-1)
    {
        if(s1[x-1]==s2[y-1])
            ans[x][y]=1+LCS(x-1,y-1,s1,s2);
        else
        {
            int k1=LCS(x-1,y,s1,s2);
            int k2=LCS(x,y-1,s1,s2);
            ans[x][y]=max(k1,k2);
        }
    }
    return ans[x][y];
}*/
class Solution
{
  public:  //Function to find the length of longest common subsequence in two strings.
  int lcs(int x, int y, string s1, string s2)
  {
      ans.assign(x+1,vector<int>(y+1,0));
      for(int i=1; i<=x; i++)
      {
          for(int j=1; j<=y; j++)
          {
              if(s1[i-1]==s2[j-1])
                  ans[i][j]=1+ans[i-1][j-1];
              else
                  ans[i][j]=max(ans[i-1][j],ans[i][j-1]);
          }
      }
      return ans[x][y];
  }
};
```

Longest Common Substring

Given two strings. The task is to find the length of the longest common substring.

Example 1:

Input: S1 = "ABCDGH", S2 = "ACDGHR"
Output: 4
Explanation: The longest common substring "CDGH" which has length 4.

Example 2:

Input: S1 = "ABC", S2 "ACB"
Output: 1
Explanation: The longest common substrings are "A", "B", "C" all having length 1.

```
class Solution{
  public:
  int longestCommonSubstr (string S1, string S2, int n, int m)
  {
      int ans[n+1][m+1];
      for(int i=0; i<=n; i++)
          ans[i][0]=0;
      for(int i=0; i<=m; i++)
          ans[0][i]=0;
      int result=0;
      for(int i=1; i<=n; i++)
      {
          for(int j=1; j<=m; j++)
          {
              if(S1[i-1]==S2[j-1])
                  ans[i][j]=1+ans[i-1][j-1];
              else
                  ans[i][j]=0;
              result=max(result,ans[i][j]);
          }
      }
      return result;
  }
};
```

Minimum Cost To Make Two Strings Identical

Given two strings **X** and **Y**, and two values **costX** and **costY**, the task is to find the minimum cost required to make the given two strings identical. You can delete characters from both the strings. The cost of deleting a character from string X is costX and from Y is costY. The cost of removing all characters from a string is the same.

Example 1:

Input: X = "abcd", Y = "acdb", costX = 10   costY = 20.
Output: 30
Explanation: For Making both strings identical we have to delete character 'b' from both the string, hence cost will be = 10 + 20 = 30.
Example 2:

Input : X = "ef", Y = "gh", costX = 10   costY = 20.
Output: 60
Explanation: For making both strings identical, we have to delete 2-2 characters from both the strings, hence cost will be = 10 + 10 + 20 + 20 = 60.

```
vector<vector<int>> ans;
/*int findMin(int x, int y, string X, string Y, int CX, int CY)
{
   if(x==-1 && y==-1)
      return 0;
   if(x==-1)
   {
      return (y+1)*CY;
   }
   if(y==-1)
   {
      return (x+1)*CX;
   }
   if(ans[x][y]!=-1)
      return ans[x][y];
   if(X[x]==Y[y])
      ans[x][y]=findMin(x-1,y-1,X,Y,CX,CY);
   else
   {
        int k1=findMin(x-1,y,X,Y,CX,CY)+CX;
        int k2=findMin(x,y-1,X,Y,CX,CY)+CY;
        ans[x][y]=min(k1,k2);
    }
    return ans[x][y];
}*/

class Solution{
        public:
        int findMinCost(string X, string Y, int costX, int costY)
        {
            // Your code goes here
            int x=X.length()+1;
            int y=Y.length()+1;
            ans.clear();
            ans.assign(x,vector<int>(y,0));
            for(int i=0; i<y; i++)
                ans[0][i]=i*costY;
            for(int i=0; i<x; i++)
                ans[i][0]=i*costX;
            for(int i=1; i<x; i++)
            {
                for(int j=1; j<y; j++)
                {
                    if(X[i-1]==Y[j-1])
                        ans[i][j]=ans[i-1][j-1];
                    else
                        ans[i][j]=min(ans[i-1][j]+costX,ans[i][j-1]+costY);
                }
            }
            return ans[x-1][y-1];
        }


};
```

Longest Repeating Subsequence

Given a string str, find the length of the longest repeating subsequence such that it can be found twice in the given string. The two identified subsequences A and B can use the same ith character from string str if and only if that ith character has different indices in A and B.

Example 1:

Input:
str = "axxxy"
Output: 2
Explanation:
The given array with indexes looks like
a x x x y
0 1 2 3 4

The longest subsequence is "xx". It appears twice as explained below.

subsequence A
x x
0 1 <-- index of subsequence A
------
1 2 <-- index of str

subsequence B
x x
0 1 <-- index of subsequence B
------
2 3 <-- index of str

We are able to use character 'x' (at index 2 in str) in both subsequences as it appears on index 1 in subsequence A and index 0 in subsequence B.

Example 2:

Input:
str = "aab"
Output: 1
Explanation:
The longest repeating subsequence is "a".

```
vector<vector<int>> ans;
class Solution {
    public:
        int LongestRepeatingSubsequence(string str){
            // Code here
            int l=str.length();
            ans.assign(l+1,vector<int>(l+1,0));
            for(int i=1; i<=l; i++)
            {
                for(int j=1; j<=l; j++)
                {
                    if(i==j || str[i-1]!=str[j-1])
      {
         int x=ans[i-1][j];
         int y=ans[i][j-1];
         ans[i][j]=max(x,y);
      }
      else
         ans[i][j]=1+ans[i-1][j-1];
                }
            }
            return ans[l][l];
        }
};
```

String Subsequence

Given two strings **S1** and **S2**, find the number of times the second string occurs in the first string, whether continuous or discontinuous.

Example 1:

Input:
S1 = geeksforgeeks
S2 = gks
Output: 4
Explanation: For the first 'g' there are 3 ways and for the second 'g' there is one way. Total 4 ways.

```
vector<vector<int>> ans;
/*int occur(string &S1, string &S2, int l1, int l2)
{
    if(l2==0)
        return 1;
    if(l1==0)
        return 0;
    if(ans[l1][l2]==-1)
    {
        if(S1[l1-1]==S2[l2-1])
        {
            int x=occur(S1,S2,l1-1,l2-1);
            int y=occur(S1,S2,l1-1,l2);
            ans[l1][l2]=x+y;
        }
        else
            ans[l1][l2]=occur(S1,S2,l1-1,l2);
    }
    return ans[l1][l2];
}*/
class Solution{
public:
  int countWays(string S1, string S2){
      // code here
```

```
        int l1=S1.length();
        int l2=S2.length();
        ans.assign(l1+1,vector<int>(l2+1,0));
        for(int i=0; i<=l1; i++)
            ans[i][0]=1;
        for(int i=1; i<=l1; i++)
        {
            for(int j=1; j<=l2; j++)
            {
                if(S1[i-1]==S2[j-1])
                    ans[i][j]=ans[i-1][j-1]+ans[i-1][j];
                else
                    ans[i][j]=ans[i-1][j];
            }
        }
        return ans[l1][l2];
    }
};
```