**Dynamic Programming – 2**

**Problem 1 : Min Cost Path Problem**

**Send Feedback**

Given an integer matrix of size m\*n, you need to find out the value of minimum cost to reach from the cell (0, 0) to (m-1, n-1).

From a cell (i, j), you can move in three directions : (i+1, j), (i, j+1) and (i+1, j+1).

Cost of a path is defined as the sum of values of each cell through which path passes.

**Input Format :**

Line 1 : Two integers, m and n

Next m lines : n integers of each row (separated by space)

**Output Format :**

Minimum cost

**Constraints :**

1 <= m, n <= 20

**Sample Input 1 :**

3 4

3 4 1 2

2 1 8 9

4 7 8 1

**Sample Output 1 :**

13



PronvnP

**Problem 2 LCS – Problem**

**Send Feedback**

Given two strings S1 and S2 with lengths M and N respectively, find the length of the longest common subsequence.

A subsequence of a string S whose length is K, is a string containing characters in same relative order as they are present in S, but not necessarily contiguous. Subsequences contain all the strings of length varying from 0 to K. For example, subsequences of string "abc" are -- ""(empty string), a, b, c, ab, bc, ac, abc.

**Input Format :**

Line 1: String S1

Line 2: String s2

**Output Format :**

Length of the longest common subsequence.

**Constraints :**

1 <= M <= 100

1 <= N <= 100

Time Limit: 1 sec

**Sample Input 1:**

adebc

dcadb

**Sample Output 1:**

3

**Explanation of Sample Input 1:**

"a", "d", "b", "c", "ad", "ab", "db", "dc" and "adb" are present as a subsequence in both the strings in which "adb" has the maximum length. There are no other common subsequence of length greater than 3 and hence the answer.

**Sample Input 2:**

abcd

acbdef

**Sample Output 2:**

3

**Explanation of Sample Input 2:**

"a", "b", "c", "d", "ab", "ac", "ad", "bd", "cd", "abd" and "acd" are present as a subsequence in both the strings S1 and S2 in which "abd" and "acd" are of the maximum length. There are no other common subsequence of length greater than 3 and hence the answer.

**Problem 3 0 1 Knapsack – Problem**

**Send Feedback**

A thief robbing a store and can carry a maximal weight of W into his knapsack. There are N items and ith item weigh wi and is value vi. What is the maximum value V, that thief can take ?

**Input Format :**

Line 1 : N i.e. number of items

Line 2 : N Integers i.e. weights of items separated by space

Line 3 : N Integers i.e. values of items separated by space

Line 4 : Integer W i.e. maximum weight thief can carry

**Output Format :**

Line 1 : Maximum value V

**Constraints**

1 <= N <= 20

1<= wi <= 100

1 <= vi <= 100

**Sample Input 1 :**

4

1 2 4 5

5 4 8 6

5

**Sample Output :**

13

**Problem 4 Matrix Chain Multiplication**

**Send Feedback**

#### Given a chain of matrices A1, A2, A3,.....An, you have to figure out the most efficient way to multiply these matrices i.e. determine where to place parentheses to minimise the number of multiplications.

#### You will be given an array p[] of size n + 1. Dimension of matrix Ai is p[i - 1]\*p[i]. You need to find minimum number of multiplications needed to multiply the chain.

##### Input Format :

Line 1 : Integer n i.e. number of matrices

Line 2 : n + 1 integers i.e. elements of array p[]

###### Output Format :

Line 1 : Minimum number of multiplication needed

###### Constraints :

1 <= n <= 100

##### Sample Input 1 :

3

10 15 20 25

##### Sample Output :

8000

##### Sample Output Explanation :

#### There are two ways to multiply the chain - A1\*(A2\*A3) or (A1\*A2)\*A3.

#### If multiply in order A1\*(A2\*A3) then number of multiplications required are 15000.

#### If multiply in order (A1\*A2)\*A3 then number of multiplications required are 8000.

#### Thus minimum number of multiplications required are 8000