# Exercise: Dynamic Programming

This document defines an exercise for ["Algorithms with Python" course @Software University](https://softuni.bg/opencourses/algorithms-with-python)".

Please submit your solutions (source code) to all below-described problems in [Judge](https://judge.softuni.org/Contests/3473).

## Binomial Coefficients

Write a program that finds the [binomial coefficient](https://en.wikipedia.org/wiki/Binomial_coefficient) for given non-negative integers n and k. The coefficient can be found recursively by adding the two numbers above using the formula:

  \binom nk = \binom{n-1}{k-1} + \binom{n-1}k \quad \text{for all integers }n,k : 1\le k\le n-1,

However, this leads to calculating the same coefficient multiple times (a problem that also occurs when solving the Fibonacci problem recursively). Use memoization to improve performance.

You can check your answers using the picture below (row and column indices start from 0):



### **Examples**

|  |  |
| --- | --- |
| **Input** | **Output** |
| 3  2 | 3 |
| 4  0 | 1 |
| 6  2 | 15 |
| 49  10 | 8217822536 |

## Word Differences

Write a program that finds all the **differences** between two strings. You have to determine the **smallest set** of **deletions** and **insertions** to make the **first** string **equal** to the **second**. Finally, you have to print the **count** of the minimum **insertions** and **deletions**.

### Input

* You will receive the **two strings** on **separate lines**

### Output

* Print the minimum **amount** of **deletions** and **insertions** as described below

### Examples

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comment** |
| YMCA  HMBB | Deletions and Insertions: 6 | One solution will be to remove "Y" and add "H" to the first: HMCA  "M" matches in both strings  Remove "C" and "A" from the first: HM  Add two "B"'s and now both strings match |
| JFEIOWHGOW  GHEWQHFEWQ | Deletions and Insertions: 12 |  |

## Connecting Cables

We are in a rectangular room. On opposite sides of the room, there are sets of n cables (n < 1000). The cables are indexed from 1 to n.

On each side of the room, there is a permutation of the cables, e.g. on one side we always have ordered {1, 2, 3, 4, 5} and on the other side, we have some permutation {5, 1, 3, 4, 2}. We are trying to connect each cable from one side with the corresponding cable on the other side – connect 1 with 1, 2 with 2, etc. **The cables are straight and should not overlap!**

The task is to find the maximum number of pairs we can connect given the restrictions above.

### **Examples**

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comments** |
| 2 5 3 8 7 4 6 9 1 | Maximum pairs connected: 5 |  |
| 4 3 2 1 | Maximum pairs connected: 1 | Any other pair can be connected as well. |
| 1 2 3 | Maximum pairs connected: 3 |  |

## Minimum Edit Distance

We have two strings, s1 and s2. The goal is to obtain s2 from s1 by applying the following operations:

* replace(i, x) – in s1, replaces the symbol at index with the character x
* insert(i, x) – in s1, inserts the character x at index i
* delete(i) – from s1, removes the character at index i

We are only allowed to modify s1, s2 always stays unchanged. Each of the three operations has a certain **cost** associated with it (positive integer number).

**Note**: the cost of the replace(i, x) operation is 0 if it does not change the character.

The goal is to find the sequence of operations which will produce s2 from s1 with **minimal cost**.

### Input

* The input consists of **five lines**.
* The **first** line is the **replacement cost**.
* The **second** line is the **insert** **cost**.
* The **third** line is the **delete** **cost**.
* After that on the next two lines are the two strings **s1** and **s2.**

### **Examples**

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comments** |
| 3  2  1  abracadabra  mabragabra | Minimum edit distance: 7 | Indices refer to the original s1 string – DELETE(3) deletes the symbol at index 3 from abracadabra, not from the modified string mabracadabra after the INSERT(0, m) operation. |
| 3  3  3  equal  equal | Minimum edit distance: 0 |  |
| 1  1  1  equal  different | Minimum edit distance: 8 |  |

## Longest String Chain

Given a list of strings, write a program that returns the longest string chain that can be built from those strings.

A string chain is defined as follows: subsequence of a given sequence in which the subsequence's elements are in sorted order (string length), lowest to highest, and in which the subsequence is as long as possible.

If several sequences with equal length exist, find the left-most of them.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| a ab abcd abc | a ab abcd |
| a ab abcd abc abcd abcde | a ab abc abcd abcde |
| abde abc abd abcde ade ae 1abde abcdef | abde abcde abcdef |

## Longest Zigzag Subsequence

A zigzag sequence is one that alternately increases and decreases. More formally, such a sequence has to comply with one of the two rules below:

1. Every even element is smaller than its neighbors and every odd element is larger than its neighbors, or
2. Every odd element is smaller than its neighbors and every even element is larger than its neighbors

1 3 2 is a zigzag sequence, but 1 2 3 is not. Any sequence of one or two elements is zigzag.

Find the longest zigzag subsequence in a given sequence.

### **Examples**

|  |  |
| --- | --- |
| **Input** | **Output** |
| 8 3 5 7 0 8 9 10 20 20 20 12 19 11 | 8 3 5 0 20 12 19 11 |
| 1 2 3 | 1 2 |
| 1 3 2 | 1 3 2 |
| 24 5 31 3 3 342 51 114 52 55 56 58 | 24 5 31 3 342 51 114 52 55 |