# Homework: Dynamic Programming

This document defines the **homework assignments** for the ["Algortihms" course @ Software University](https://softuni.bg/trainings/1194/Algorithms-September-2015). Please submit a single zip / rar / 7z archive holding the solutions (source code) of all below described problems.

## 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. **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 conntected 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 i 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 stays unchanged at all times. 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 doesn’t actually change the character.

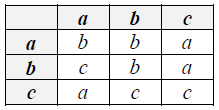
The goal is to find the sequence of operations which will produce s2 from s1 with **minimal cost**.

Examples

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comments** |
| cost-replace = 3  cost-insert = 2  cost-delete = 1  s1 = abracadabra  s2 = mabragabra | Minimum edit distance: 7  INSERT(0, m)  DELETE(3)  DELETE(4)  REPLACE(6, g) | 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. |
| cost-replace = 5  cost-insert = 2  cost-delete = 1  s1 = nqma bira  s2 = ima bira | Minimum edit distance: 4  DELETE(0)  DELETE(1)  INSERT(1, i) | We can obtain s2 with two operations – DELETE(0) + REPLACE(1, i), but the cost of the REPLACE operation is high, that’s why the solution involves three operations, their total cost is smaller.  The INSERT can be performed also at index 0 and index 2. |
| cost-replace = 3  cost-insert = 3  cost-delete = 3  s1 = equal  s2 = equal | Minimum edit distance: 0 |  |
| cost-replace = 1  cost-insert = 1  cost-delete = 1  s1 = equal  s2 = different | Minimum edit distance: 8  INSERT(0,d)  INSERT(1,i)  INSERT(2,f)  INSERT(3,f)  REPLACE(1,r)  REPLACE(2,e)  REPLACE(3,n)  REPLACE(4,t) |  |

## \* Symbol Multiplication

We have an **alphabet** of k symbols (a finite number) and a **multiplication table** showing the result of multiplying each two symbols of the alphabet. E.g., the alphabet is {a, b, c} and the multiplication table is:



This shows that a\*a = b, a\*b = b, b\*a = c, etc. As shown in the example, multiplication is **not commutative or associative** – a\*b != b\*a, therefore, the order of multiplication is essential.

We have a string S comprised of characters from the alphabet. The task is to find whether we can obtain the symbol **'a'** by inserting brackets in the string – all symbols in brackets are multiplied. If so, print the string with the brackets inserted. Print "No solution" otherwise. Assume 'a' will always be in the alphabet.

Examples

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comments** |
| Alphabet = {a,b,c}  Table =  bba  cba  aac  S = abc | ((a\*b)\*c) | ((a\*b)\*c) = (b\*c) = a |
| Alphabet = {a,b,c}  Table =  bba  cba  aac  S = bacacbcabbbcacab | (((b\*a)\*(c\*a))\*(((c\*(b\*c))\*a)\*((b\*((b\*b)\*(c\*a)))\*(c\*(a\*b))))) |  |
| Alphabet = {a,b}  Table =  bb  bb  S = abbbaaba | No solution | No combination of two symbols produces 'a' after multiplication. |