Longest Common Subsequence

E-OLYMP **1618. The longest Common Subsequence** Two sequences are given. Find the length of their longest common subsequence. A subsequence is a sequence derived from the original sequence by deleting some elements while preserving the order of the remaining elements.

Input. The first line contains the length n ($1 \le n \le 1000$) of the first sequence. The second line contains the elements of the first sequence – integers not exceeding 10^4 in absolute value.

The third line contains the length m ($1 \le m \le 1000$) of the second sequence. The fourth line contains the elements of the second sequence – integers not exceeding 10^4 in absolute value.

Output. For the given two sequences, print the length of their longest common subsequence.

Sample input	**Sample output**
3 1 2 3 4 2 1 3 5	2

► A ***subsequence*** of a sequence is a set of elements that appear in left-to-right order, but not necessarily consecutively. A subsequence can be derived from a sequence only by deletion of some elements.

For example, consider the sequence {2, 1, 3, 5}. Then

- {1, 5}, {2}, {2, 3, 5} are subsequences;
- {5, 1}, {2, 3, 1} are not subsequences;

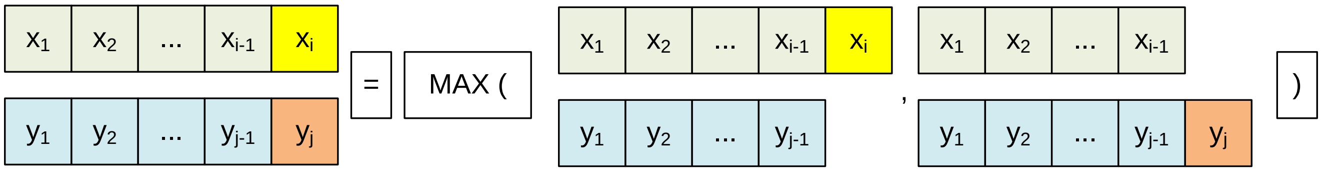
A ***common subsequence*** of two sequences is a subsequence that appears in both sequences. A ***longest common subsequence*** (***lcs***) is a common subsequence of maximal length.

For example, the longest common subsequence of {1, 2, 3} and {2, 1, 3, 5} can be {1, 3} or {2, 3}. The length of lcs is 2.

Let f(i, j) be the length of the longest common subsequence of sequences $x_1x_2\ldots x_i$ and $y_1y_2\ldots y_j$.

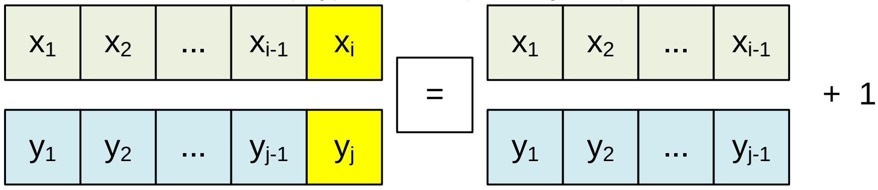
If $x_i \ne y_j$, then we find ***lcs*** among $x_1x_2\ldots x_i$ and $y_1y_2\ldots y_{j-1}$, and also among $x_1x_2\ldots x_{i-1}$ and $y_1y_2\ldots y_j$. Return the biggest value:

$$f(i, j) = \max(\ f(i, j-1),\ f(i-1, j)\ )$$



If $x_i = y_j$, then we find ***lcs*** among $x_1x_2\ldots x_{i\text{-}1}$ and $y_1y_2\ldots y_{j\text{-}1}$:

$$f(i, j) = 1 + f(i-1, j-1)$$

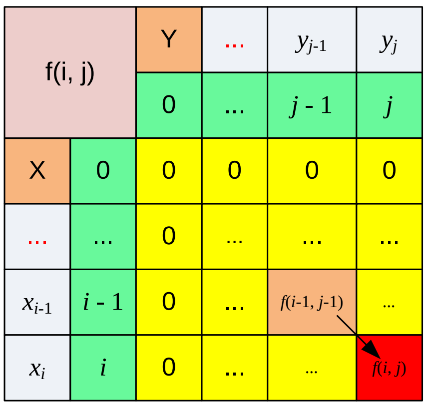


If one of the sequences is empty, then their lcs is empty:

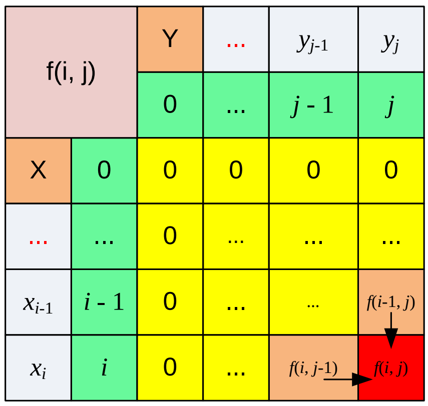
$$f(0, j) = f(i, 0) = 0$$

Let's summarize the recurrence relation:

$$f(i,j) = \begin{cases} \max(f(i,j-1), f(i-1,j)), & x_i \neq y_j \\ f(i-1,j-1)+1, & x_i = y_j \\ 0, & i = 0 \ or \ j = 0 \end{cases}$$

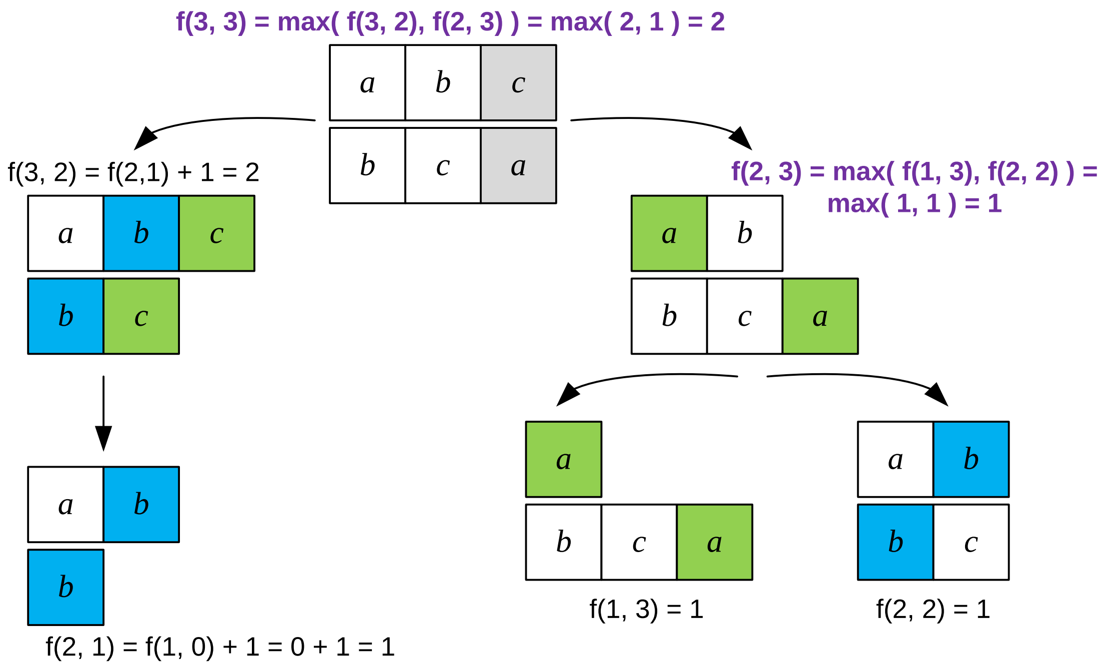


$x_i = y_j$
$f(i, j) = f(i - 1, j - 1) + 1$



$x_i \neq y_j$
$f(i, j) = \max(f(i, j - 1), f(i - 1, j))$

Consider an example of calculations:



The values f(i, j) will be stored in array m[0..1000, 0..1000], where m[0][i] = m[i][0] = 0. Each next line of array m[i][j] is calculated through the previous one. Therefore, to find the answer, it is enough to keep in memory only two lines of length 1000.

Example

Let X = *abcdgh*, Y = *aedfhr*. The longest common subsequence is *adh*, its length equals to f(6, 6) = 3.

f(i, j)		X	a	b	c	d	g	h
		0	1	2	3	4	5	6
Y	0	0	0	0	0	0	0	0
a	1	0	1(a)	1	1	1	1	1
e	2	0	1	1	1	1	1	1
d	3	0	1	1	1	2(d)	2	2
f	4	0	1	1	1	2	2	2
h	5	0	1	1	1	2	2	3(h)
r	6	0	1	1	1	2	2	3

f(6, 6) = max(f(6, 5), f(5, 6)) = max(2, 3) = 3, because Y[6] = $r \neq h$ = X[6].

f(5, 6) = 1 + f(4, 5) = 1 + 2 = 3, because Y[5] = *h* = X[6].

Exercise

Fill the table:

f(i, j)		X	d	f	c	a	b	a
		0	1	2	3	4	5	6
Y	0							
f	1							
d	2							
c	3							
c	4							
a	5							
a	6							

Algorithm realization

Arrays x and y contain input sequences, *n* and *m* are their lengths. Array mas contains two last lines of dynamic calculations.

```
#define SIZE 1010
int x[SIZE], y[SIZE], mas[2][SIZE];
```

Main part of the program. Read input sequences to arrays, starting from the first index. Then read the data into x[1..*n*] and y[1..*m*].

```
scanf("%d",&n);
for(i = 1; i <= n; i++) scanf("%d",&x[i]);
scanf("%d",&m);
for(i = 1; i <= m; i++) scanf("%d",&y[i]);
```

Fill array mas with zeroes. Dynamically calculate the values f(*i*, *j*). Initially mas[0][*j*] contains the values f(0, *j*). Then put into mas[1][*j*] the values f(1, *j*). Since to calculate f(2, *j*) it is enough to have the values of the previous row of mas array, the values of f(2, *j*) can be stored in mas [0][*j*], the values of f(3, *j*) in mas [1][*j*] and so on.

```
memset(mas,0,sizeof(mas));
for(i = 1; i <= n; i++)
for(j = 1; j <= m; j++)
  if (x[i] == y[j])
    mas[i%2][j] = 1 + mas[(i+1)%2][j-1];
  else
    mas[i%2][j] = max(mas[(i+1)%2][j],mas[i%2][j-1]);
```

Print the answer, that is located in the cell mas[n][m]. Take the first argument modulo 2.

```
printf("%d\n",mas[n%2][m]);
```

Algorithm realization – recursion

```
#include <stdio.h>
#include <string.h>
#define SIZE 1002

int x[SIZE], y[SIZE], dp[SIZE][SIZE];
int n, m, i, j, res;

int max(int i, int j)
{
  return (i > j) ? i : j;
}

int lcs(int *x, int *y, int m, int n)
{
  if (m == 0 || n == 0)
    return 0;
  if (dp[m][n] != -1) return dp[m][n];

  if (x[m] == y[n])
    return dp[m][n] = 1 + lcs(x, y, m - 1, n - 1);
  else
    return dp[m][n] = max(lcs(x, y, m, n - 1), lcs(x, y, m - 1, n));
}

int main(void)
{
  scanf("%d", &n);
  for (i = 1; i <= n; i++) scanf("%d", &x[i]);
  scanf("%d", &m);
  for (i = 1; i <= m; i++) scanf("%d", &y[i]);

  memset(dp, -1, sizeof(dp));
  res = lcs(x, y, n, m);
  printf("%d\n", res);
  return 0;
}
```

E-OLYMP **1079. Removing the letters** You are given two words (each word consists of upper-case English letters). Delete some letters from each word so that the resulting words become equal.

Find the maximum possible length of the resulting word.

► The answer to the problem is the length of the ***longest common subsequence*** (**LCS**) of input sequences of uppercase Latin letters.

Let f(i, j) be the longest common subsequence of sequences $x_1x_2\ldots x_i$ and $y_1y_2\ldots y_j$.

If $x_i \neq y_j$, then find LCS between $x_1x_2\ldots x_{i-1}$ and $y_1y_2\ldots y_j$, and also between $x_1x_2\ldots x_i$ and $y_1y_2\ldots y_{j-1}$. Return the largest of them:

$$f(i, j) = \max(f(i, j - 1), f(i - 1, j))$$

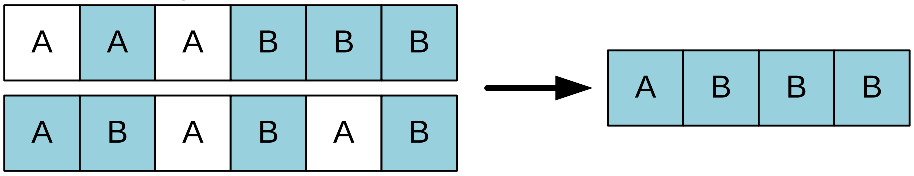
If $x_i = y_j$, then find LCS between $x_1x_2\ldots x_{i-1}$ and $y_1y_2\ldots y_{j-1}$:

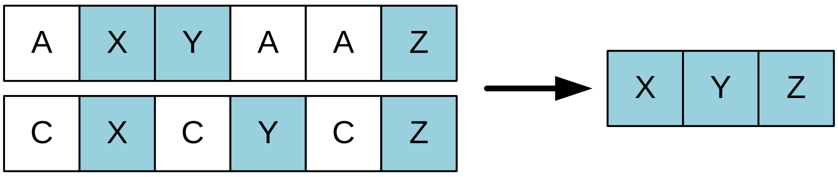
$$f(i, j) = 1 + f(i - 1, j - 1)$$

The values f(i, j) will be stored in array m[0.. SIZE, 0.. SIZE], where m[0][i] = m[i][0] = 0. Since the length of words is no more than 200 characters then assign SIZE = 201.

Each next line of array m[i][j] is calculated through the previous one. Therefore, to find the answer, it is enough to keep only two lines in memory.

Here is given the largest common subsequences for samples.





E-OLYMP **4260. LCS - 2** Two strings are given. Find and print their longest common subsequence.

► In the problem you must find the largest common subsequence (LCS) of two strings and print it.

Construct an array dp, where dp[i][j] is the length of LCS of strings $x_{[1\ldots i]}$ and $y_{[1\ldots j]}$. The value of dp[n][m] equals to the length of LCS of input strings ($|x| = n$, $|y| = m$). Move through the matrix dp from the cell (n, m) to (0, 0). For the current position (i, j) we have:

- If symbols x_i and y_j are the same, then this character must be present in the LCS, store it into the resulting string *res*. Move in array dp from cell (i, j) to cell ($i - 1$, $j - 1$), that is, then construct LCS ($x_{[1\ldots i-1]}$, $y_{[1\ldots j-1]}$).
- If symbols x_i and y_j are different, then we can move in array dp from cell (i, j) either to cell (i, $j - 1$) or to cell ($i - 1$, j). Since the largest subsequence is being built, the transition should be made to the cell where the value is greater. If dp[$i - 1$][j] = dp[i][$j - 1$], we can go to any of the specified cells.

Find the LCS for two string given in a sample.

dp[i][j]		X	a	b	a	c	a	b	a
		0	1	2	3	4	5	6	7
Y	0	0	0	0	0	0	0	0	0
d	1	0	0	0	0	0	0	0	0
a	2	0	1	1	1	1	1	1	1
c	3	0	1	1	1	2	2	2	2
a	4	0	1	1	2	2	3	3	3
b	5	0	1	2	2	2	2	4	4
c	6	0	1	2	2	3	3	4	4

We start to search the LCS from position $(i, j) = (6, 7)$.

$y[6] \neq x[7]$, move to any adjacent cell with the maximum value. For example to (5, 7).

$y[5] \neq x[7]$, move to (5, 6).

$y[5] = x[6]$ = '*b*', move diagonally, include letter '*b*' to LCS.

Declare the inut strings *x* and *y*. To find their LCS declare dp array.

```
#define MAX 1001
int dp[MAX][MAX];
string x, y, res;
```

Read the input lines. Add a space to them so that the indexing will start from 1.

```
cin >> x; n = x.length(); x = " " + x;
cin >> y; m = y.length(); y = " " + y;
```

Finding the longest common subsequence.

```
for (i = 1; i <= n; i++)
for (j = 1; j <= m; j++)
  if (x[i] == y[j])
    dp[i][j] = dp[i - 1][j - 1] + 1;
  else
    dp[i][j] = max(dp[i - 1][j], dp[i][j - 1]);
```

Construct the LCS in the string *res*.

```
i = n; j = m;
while (i >= 1 && j >= 1)
  if (x[i] == y[j])
  {
    res = res + x[i];
```

```
    i--; j--;
  }
  else
  {
    if (dp[i - 1][j] > dp[i][j - 1])
      i--;
    else
      j--;
  }
```

Invert and print the resulting string.

```
reverse(res.begin(), res.end());
cout << res << endl;
```

E-OLYMP **1765. Three sequences** Three sequences of integers are given. Find the length of their longest common subsequence.

► Let a, b, c be three input sequences. Let f(i, j, k) be the length of LCS of sequences a[1..i], b[1..j] and c[1..k]. The value f(i, j, k) we shall keep in dp[i][j][k].

If a[i] = b[j] = c[k], then

$$f(i, j, k) = 1 + f(i - 1, j - 1, k - 1)$$

Otherwise

$$f(i, j, k) = \max(\ f(i - 1, j, k), f(i, j - 1, k), f(i, j, k - 1)\ )$$