

**Courses » Design and analysis of algorithms**[Announcements](#)**[Course](#)**[Ask a Question](#)[Progress](#)[FAQ](#)**Course outline**[How to access the portal](#)[Week 1: Introduction](#)[Week 1: Analysis of algorithms](#)[Week 1 Quiz](#)[Week 2: Searching and sorting](#)[Week 2 Quiz](#)[Week 2 Programming Assignment](#)[Week 3: Graphs](#)[Week 3 Quiz](#)[Week 3 Programming Assignment](#)[Week 4: Weighted graphs](#)[Week 4 Quiz](#)[Week 4 Programming Assignment](#)[Week 5: Data Structures: Union-Find and Heaps](#)

# Dividing Sequences

**Due on 2018-10-21, 23:59 IST**

- Select your language (C/C++/Java/Python2/Python3)
- Paste your code into the submission window.
- There are some public test cases and some (hidden) private test cases.
- "Compile and run" will evaluate your submission against the public test cases.
- "Submit" will evaluate your submission against the hidden private test cases and report a score on 100. There are 11 private testcases in all, each with equal weightage.
- Ignore warnings about "Presentation errors".

## Dividing Sequences

*(IARCS OPC Archive, K Narayan Kumar, CMI)*

This problem is about sequences of positive integers  $a_1, a_2, \dots, a_N$ . A subsequence of a sequence is anything obtained by dropping some of the elements. For example,  $3, 7, 11, 3$  is a subsequence of  $6, 3, 11, 5, 7, 4, 3, 11, 5, 3$ , but  $3, 3, 7$  is not a subsequence of  $6, 3, 11, 5, 7, 4, 3, 11, 5, 3$ .

A *fully dividing sequence* is a sequence  $a_1, a_2, \dots, a_N$  where  $a_i$  divides  $a_j$  whenever  $i < j$ . For example,  $3, 15, 60, 720$  is a fully dividing sequence.

Given a sequence of integers your aim is to find the length of the longest fully dividing subsequence of this sequence.

Consider the sequence  $2, 3, 7, 8, 14, 39, 145, 76, 320$

It has a fully dividing sequence of length 3, namely  $2, 8, 320$ , but none of length 4 or greater.

Consider the sequence  $2, 11, 16, 12, 36, 60, 71, 17, 29, 144, 288, 129, 432, 993$ .

It has two fully dividing subsequences of length 5,

- $2, 11, 16, 12, 36, 60, 71, 17, 29, 144, 288, 129, 432, 993$  and
- $2, 11, 16, 12, 36, 60, 71, 17, 29, 144, 288, 129, 432, 993$

and none of length 6 or greater.

## Solution hint

Let the input be  $a_1, a_2, \dots, a_N$ . Let us define  $Best(i)$  to be the length of longest dividing sequence in  $a_1, a_2, \dots, a_i$  that includes  $a_i$ .

Write an expression for  $Best(i)$  in terms of  $Best(i)$  with  $i < i$  with base case

**Week 5: Divide and Conquer**

**Week 5 Quiz**

**Week 6: Data Structures: Search Trees**

**Week 6: Greedy Algorithms**

**Week 6 Quiz**

**Week 7: Dynamic Programming**

**Week 7 Quiz**

**Week 7 Programming Assignment**

**Dividing Sequences**

**Week 8: Linear Programming and Network Flows**

**Week 8: Intractability**

**Week 8 Quiz**

**TEXT TRANSLATION**

Write an expression for  $Best(j)$  in terms of  $Best(i)$  with  $j < i$ , with base case  $Best(1) = 1$ . Solve this recurrence using dynamic programming.

[Full solution](#)

## Input format

The first line of input contains a single positive integer  $N$  indicating the length of the input sequence. Lines  $2, \dots, N+1$  contain one integer each. The integer on line  $i+1$  is  $a_i$ .

## Output format

Your output should consist of a single integer indicating the length of the longest fully dividing subsequence of the input sequence.

## Test Data

You may assume that  $N \leq 10000$ .

## Example:

Here are the inputs and outputs corresponding to the two examples discussed above.

### Sample input 1:

```
9
2
3
7
8
14
39
145
76
320
```

### Sample output 1:

```
3
```

### Sample input 2:

```
14
2
11
16
12
36
60
71
17
29
144
288
```



104  
975  
131  
551  
307  
469  
917  
784  
415  
376  
555  
600  
44  
443  
393  
555  
261  
998  
483  
933  
219  
106  
963  
104  
574  
798  
22  
293  
308  
431



Due Date Exceeded.

7 out of 11 tests passed.

You scored 81.8181818182/100.

Your last recorded submission was :

```

1 def readinput():
2     n = int(input())
3     for j in range(n):
4         nextnum = int(input())
5         insequence.append(nextnum)
6         best.append(0)
7     return
8
9 def solve():
10    for j in range(len(insequence)):
11        prev = [ best[k] for k in range(j) if insequence[j]%insequence[k] == 0 ]
12        if prev:
13            best[j] = 1 + max(prev)
14        else:
15            best[j] = 1
16
17 insequence = []
18 best = []
19 readinput()
20 solve()
21 print(max(best))
22

```

Sample solutions (Provided by instructor)

Select the Language . Python2

End