UniLecs #137_1. Multiplication of digits

UniLecs • November 06, 2018

Задача: дано натуральное число N. Необходимо найти наименьшее натуральное число, в ктр произведение его цифр было бы равно N. Если такого числа не существует, вывести -1.

Входные данные: N - натуральное число от 1 до 10^9.

Вывод: наименьшее натуральное число, произведение цифр ктр было бы равно N. -1 - если такого числа нет.

Пример:

N = 11; Answer = -1

N = 12; Answer = 26

N = 14; Answer = 27

Season Cup 5:

UniLecs, Season Cup 5							
№	Имя	UserName	Языки П.	Баллы	#133	#135	#137
1	Евгений	@jinxonik	много	6,2	1,3	1,1	3,8
2	Костя	@mrmeison	JS	1	1		
3		@abdujabbor1987	Python	1	1		
4		@FutorioFranklin	Python	1	1		
5		@TEXHIK	Julia	2	1		1
6	Саша	@pakrulin	JS	1	1		
7	Andrew Bystrov		Java	2	1		1
8	Алексей	@akolosov364	JS	3,1	1	1	1,1
9		@Shoxrux_NUU	C++	1	1		
10	Дмитрий	@my_diamonds_dancing	Python, C#	1,8	1,1	0,7	
11		@LexxWanderlust	Swift	1	1		
12	Антон		Rust, Haskell	3	1	1	1
13		@P9oSi	C#	1	1		
14	Роман	@LostInKadath	Python, C++	2	1		1
15	Егор	@egormasharskii	C++, Python	5,9	1	3,1	1,8
16	Рустем	@rustem_b	много	1,4	1,4		
17	Кирилл	@kirillmotrichkin	Python	3,5	1	1	1,5
18	Михаил	@lPestl	C#, C++, F#	5	1,7	2,3	1
19	Михаил	@voodoo_woodpecker	Python	4,2	1,3	1,5	1,4
20		@kupp1	Python	1			1
21		@DaurenAbd	C++	1			1
22		@Rusik666	C++	1			1

Season Cup 5: Турнирная таблица

Реализация:

1. **Andrew Bystrov**, Java: 1 балл

> *Идея заключается в том, что необходимо разложить данное число N так, чтобы все его сомножители были меньше 9 (т.е. произвести факторизацию числа так, чтобы все сомножители были <= 9) (ну и очевидно, больше 2) Иначе может получится ситуация, если сомножитель будет больше чем 9 (т.е. число вида ab) и тогда, мы можем предположить, что результат у нас должен быть в виде q*w*e*...*ab. Но это не сработает, т.к. мы должы по условию делать q*w*e*...*a*b. Логично, что для простых чисел мы всегда вернем -1. Соответственно, если мы в какой то момент времени видим, что числе не делится на число из диапазона 2-9, значит выводим -1. Иначе*

> *запоминаем все цифры и выводим их в порядке возрастания. Начнем делить мы с 9 до 2, т.к. у нас может возникнуть ситуация (если мы будем делить с 2 по 9), например, с числом 100 что мы поделим 100 /2 = 50, и потом 50/2 = 25. Значит у нас будет уже цифры 2 и 2, хотя если бы мы делили в обратном порядке, то мы сразу бы дошли до 5 (100 / 5 = 20), потом снова до 5 (20 / 5 = 4) и уже полчили 4 => ответ 455*

https://gist.github.com/AndrewBystrov/bcf5089e9a2bdc4b70ecb6d7e109eb5d

2. **@akolosov364**, JS, Rust: 1.1 балла

> *Задача сводится к поиску всех делителей числа. Если число имеет простой делитель больше 10, то задача не имеет решения. Имеет смысл искать делители, начиная с 9. Это сократит количество чисел в записи числа: 9=3x3, 8=2x4, 6=2x3, 4=2x2. Как только все делители были найдены, достаточно их просто перевернуть чтобы получить ответ.*

```
function solve(num) {
    if (num < 10) {
        return num;
    }
    const digits = [];

    let current = 9;
    while(current > 1) {
        if (num % current === 0) {
            digits.push(current);
            num /= current;
        } else {
            current--;
        }
    }

    if (num > 9) {
        return -1;
    }

    return +digits.reverse().join("");
}
```

@akolosov364, JS

https://gist.github.com/KolosovAO/057af7e0d6e082595c4547deadf47965

Test:

https://repl.it/@AlieksieiKoloso/task137

Rust:

https://gist.github.com/KolosovAO/66302ad6e15d76f0d2090b2f64443867

Rust Test:

https://repl.it/@AlieksieiKoloso/task137rust

3. **@jinxonik**, 24 ЯП (1 + 0.5 балла за доп.метод с помощью разложения числа на простые множители + 23 *0.1 доп.реализации на различных ЯП) = 3.8 балла

> *Метод 1: Через последовательное деление на разные цифры*

> *Метод 2: Разложение числа n на простые множители. Возвращает словарь формата {множитель: степень}*

```c
#include <stdio.h>
// Получение минимального натурального числа, произведение
// цифр которого равно n, либо -1, если такого числа нет.
// Значение radix задаёт систему счисления.
long long mul_of_digits(int n, int radix)
{
  // Системы счисления < 2 не годятся
  if (radix < 2) return -1;
  // Для n < radix результатом будет n (n <= 0 представить произведением цифр нельзя)
  if (n < radix) { return n > 0 ? n : -1; }
  // mul - множитель (его десятичный логарифм - это номер цифры с конца, 0 = последняя)
  long long result = 0, mul = 1;
  // Перебираем все цифры i от radix-1 до 2
  for (int i = radix-1; i > 1; --i) {
    while (n % i == 0) {
      // Если n делится на i, добавляем цифру в начало
      result += i * mul;
      // Делим n на i и увеличиваем номер цифры
      n /= i;
      mul *= radix;
    }
  }
  // Если остаток == 1, число не содержит множителей > 9,
  // иначе такой множитель нельзя представить одной цифрой
  return n == 1 ? result : -1;
}

int main()
{
  int n[] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 34, 72, 81, 100,
    168, 216, 324, 330, 432, 648, 1000, 2520, 4212, 7350, 11155, 21600, 30240, 1000000};

  for (int i = 0; i < sizeof(n)/sizeof(*n); ++i) {
    long long result10 = mul_of_digits(n[i], 10);  // в десятичной системе счисления
    long long result16 = mul_of_digits(n[i], 16);  // в 16-ричной системе счисления
    char *str = "%d: %lld / %llX (hex)\n";
    if (result16 == -1) { str = "%d: %lld / %lld (hex)\n"; }  // для корректного вывода result16 == -1
    printf(str, n[i], result10, result16);
  }
}
```

@jinxonik, C

https://gist.github.com/jin-x/4af215a71f8c6693812e71da1d6b54a5

Компилируемые:

1. C : Test

2. C++ [изменённый метод в сравнении с C]

2a. C++ [шаблон + constexpr: расчёт на этапе компиляции, программа только выводит результаты]

3. C#

4. Java

5. Delphi 7

6. PascalABC.NET

7. Rust

8. Go

9. Visual Basic .NET

Ассемблеры:

10. fasm 1 for Windows x64

10a. fasm 1 for Windows x86 (32-bit) [расчёты через макросы во время компиляции, программа только выводит результаты]

11. MASM32 for Windows x86 (32-bit) [алгоритм, возвращающий строку, а не число]

12. GNU Assembler (GAS, AT&T syntax) for Linux x64

13. TASM for MS-DOS (.COM) [исполняемый код - 318 байт, включая 140 байт данных]

Скриптовые:

14. Python 3

15. PHP

16. JavaScript

17. Ruby

18. Perl

19. Lua

20. VBScript

21. VBA for Excel

22. CMD for Windows (+ модицикация)

23. Bash for Linux

24. GolfScript (+ модицикация) [эзотерический]

4. **@TEXHIK**, Java: 1 балл

> *Чем больше множителИ, тем меньше множителЕЙ. А чем меньше множителей тем меньше порядок. Так что факторизуем бьём начиная с самых больших цифр.*

https://repl.it/@levon1550/T137

5. **Антон**, Rust: 1 балл

Подробный разбор смотрите в gist файле:

https://gist.github.com/AnthonyMikh/750c4d9bec3be66bcce28197494367de

Test:

https://play.rust-lang.org/?gist=5150d56ba5b6781dfe662bd61718a575

6. **@kupp1**, Python: 1 балл

> *Пусть n - введенное число. Пусть p - число, которое требуется составить. Подобную задачу можно свести к более строгому условию: необходимо найти такое сочетание цифр от 2 до 9, произведение которых даст какое-то определенное число n. Далее необходимо составить из найденных цифр минимально возможное число p. Я исключаю из этого диапазона 1 и 0, так как 1 результат произведения не изменит, а 0 - превратит его в 0. По основной теореме арифметики есть только один способ разложить число на простые делители. Этим и воспользуемся: найдем простые делители числа n факторизацией и будем пробовать составить число p из этих простых чисел. Логично, что перемножение простых делителей числа n даст нам само число n, остается только определить, что если хотя бы один из простых делителей не однозначен (т.е двухзначен, трехзначен и т.д.) то по такому числу n число p составить невозможно. Таким образом, мы можем работать только с однозначными простыми делителями, т.е. только с делителями 2, 3, 5, 7 - это единственные однозначные простые числа. У нас есть определение случая, когда задача не решаема. Когда задача решаема, у нас есть набор простых однозначных чисел, из который надо составить число p. Всего у нас может получится 8 цифр в числе p:*

- 9 - 3*3

- 8 - 2^3
- 7 - есть изначально
- 6 - 2*3
- 5 - есть изначально
- 4 - 2^2
- 3 - есть изначально
- 2 - есть изначально

> *Наша задача сначала найти количество 9, потом 8, и т.д. от большего у меньшему. Изначально в цикле мы просто подсчитываем количество цифр 2,3,5,7. Далее, вычисляем количество 9: это количество пар троек (не забываем уменьшить кол-во троек) и так далее. Потом выводим в одну строчку все наши возможные цифры от меньшей к большей. Число p вычислено.*

https://gist.github.com/kupp1/4e2f32320a2cbc68dcd9e52bebcbb46a

7. **@egormasharskii**, C++: (1 + 0.5 балла за Метод 3 + 0.3 балла за Метод 2 с оптимизацией 1го метода) = 1.8 балла

> *Метод 1: Brute force. Раскладываем число на все возможные множители от 2 до 9 и выбираем наименьшее число составленное из них.*

> *Метод 2: Brute force optimization. Поскольку в разложении числа перестановка множителей ничего не меняет, мы можем использовать этот факт и в предыдущем решении использовать кэширование.*

> *Метод 3: Можно заметить, что чем больший делитель мы выбираем, тем меньше становится оставшаяся часть и тем меньше разрядов будет в ответе. Поэтому можно избавиться от перебора и искать делители не от 2 до 9, а от 9 до 2. В таком случае мы получим минимальное по количеству разрядов число в ответе при первом же разложении. Единственное что там останется сделать - это поменять порядок цифр в ответе чтобы максимальные делители ушли в младшие разряды.*

http://cpp.sh/9l6xa

8. **@DaurenAbd**, C++: 1 балл

Смотри разбор в gist файле:

https://gist.github.com/DaurenAbd/8foad2410ec5531df1d5336b6527f356

9. **@LostInKadath**, C++: 1 балл

> *По сути задача сводится к факторизации числа - нахождению его множителей. Но вводится дополнительное ограничение - множители не должны превышать 10, ибо они выражаются цифрами.Здесь реализован самый простой вариант факторизации - перебор. Перебор делителей осуществляется с бОльших цифр к меньшим.Полученные делители складываются в вектор и сортируются в возрастающем порядке для получения минимального числа.Если на каком-то шаге алгоритм не находит делитель в интервале [2, 9], происходит возвращение к предыдущему шагу и проба другого делителя.*

https://gist.github.com/LostInKadath/a246e89e5cae17fa534166d59e68a096

10. **@Rusik666**, C++: 1 балл

Разбор решения смотри в gist-файле:

https://github.com/ruslan1979/unilecs/blob/master/task137.cpp

11. **@voodoo_woodpecker**, Python: (1 + 0.3 балла за доп.решение с помощью оптимизации первого алгоритма + 0.1 балла за доп.ЯП) = 1.4 балла

> *Метод 1. Сначала разбиваем число на простые множители-цифры (2,3,5,7)# Потом комбинируем множители в наиболее эффективные комбинации.*

> *Метод 2. Чтобы не мучиться с заменой комбинаций множителей на втором этапе# просто ищем их в оптимальном порядке: 9(3x3) > 8 (2x2x2) > 6 (2x3) > 4 (2x2) > 3 > 2, остальное (5, 7) неважно, ибо не комбинируется.*

https://gist.github.com/MikePeleah/8dbafe95acc7cd6f9b4663279531c9fc

Test:

https://repl.it/@MikePeleah/UniLecs137Py

JS:

https://gist.github.com/MikePeleah/a8c7e0ebca4251a2ce2ee049037142fb

Test JS:

https://repl.it/@MikePeleah/UniLecs137JS

12. **@kirillmotrichkin**, Python: 1.5 балла

> *Метод 1. Будем делить заданное число на числа меньше 10, то есть цифры. Если будет получаться деление, то записываем делитель в ответ и продолжаем ту же работу с частным.*

> *Метод 2. Будем делить заданное число на числа меньше 10, то есть цифры. Если будет получаться деление, то записываем делитель в ответ и продолжаем ту же работу с частным. Рекурсия здесь выглядит красивее!*

Метод 1:

https://gist.github.com/superkiria/1c4d1743272d2b0ff9053652f6334a38

Test:

https://repl.it/@superkiria/unilecs137-1multiplicationofdigits

Метод 2:

https://gist.github.com/superkiria/b2d3c08ff924323b68816df38e7ca445

Test:

https://repl.it/@superkiria/unilecs137-2multiplicationofdigits

13. **@lPestl**, C#: 1 балл

Смотри подробный разбор в gist-файле:

https://gist.github.com/lpestl/0b8912a2aa45d377ec10ab4a622854d4