Liczba Grahama. Rozszerzamy notację strzałkową, będziemy pisali

$$\uparrow^n = \underbrace{\uparrow \ldots \uparrow}_{n},$$

gdzie

$$a \uparrow^n b = \begin{cases} ab, & \text{dla } n = 0, \\ 1 & \text{dla } n \geqslant 0 \text{ oraz } b = 0, \\ a \uparrow^n b = \underbrace{a \uparrow^{n-1} (a \uparrow^{n-1} (\ldots \uparrow^{n-1} a))}_{b \text{ kopii } a} & \text{w przeciwnym wypadku.} \end{cases}$$

$3 \uparrow^3 3 = 3 \uparrow^2 (3 \uparrow^2 3)$
$3 \uparrow^2 3 = 3^{3^3} = 3^{27} = 7\,625\,597\,484\,987$
$a \uparrow^4 b = \underbrace{a \uparrow^3 (a \uparrow^3 (\ldots \uparrow^3 a))}_{b \text{ kopii } a}$

Niech teraz $f(n) = 3 \uparrow^n 3$, wtedy *liczba Grahama* zdefiniowana jest jako $G = f^{64}(4)$, gdzie potęga przy funkcji oznacza liczbę powtórzeń. Liczba ta wystąpiła w badaniach nad uogólnionym problemem Ramseya.

Ciekawostka: G jest „znacznie" większe od liczby Mosera. Pokazano, że ta ostatnia nie jest większa od $f^3(4)$.

Posłowie I: Naprawdę duże liczby

Jaką największą liczbę może zwrócić program długości co najwyżej k (o co najwyżej k znakach)?

Pojęciem dualnym do wspominanego obok jest *złożoność Kołmogorowa* (od Andrieja Kołmogorowa) liczby $n \in \mathbb{N}$ – długość najkrótszego programu, który zwraca liczbę n.

Przez gener(k) oznaczmy odpowiedź na powyższe pytanie ($k \in \mathbb{N}$). Program, który ma 100 znaków, może zwrócić liczbę zdecydowanie większą niż 100. Program złożony z 1000 znaków może zwrócić liczbę bardzo zdecydowanie większą niż 1000, ale jak bardzo zdecydowanie? Jak szybko rosną możliwości programu wraz ze wzrostem liczby jego znaków?

Znaków, których możemy użyć do napisania programów, jest skończenie wiele. Stąd programów o długości co najwyżej k również jest skończenie wiele, a tylko niektóre z nich będą działać poprawnie i zwracać liczby. Niewątpliwie któryś z nich wygeneruje liczbę największą.

Może zdarzyć się, że żaden program o długości co najwyżej k nie zwraca liczby. W takiej sytuacji ustalmy, że gener$(k) = 0$. To się zdarzy tylko dla małych k.

Okazuje się, że gener(i) dla niedużych $i \in \mathbb{N}$ są naprawdę pokaźne, przy nich liczby Mosera lub Grahama to zupełne mikrusy. Co ciekawe, kiedy wybierzemy konkretne i, wartość gener(i) jest *nieobliczalna*, nie da się obliczyć jej dokładnej wartości – wyjaśnienie w tekście na sąsiedniej stronie. Można natomiast pokazać, od jakiej liczby gener(i) na pewno jest większe – stąd śmiałość w nazywaniu liczby Mosera mikrusem. Przyjrzyjmy się poniższemu programowi.

```
1  Function arrow(a, n, b, k)
2      if (k > 1) then
3          return arrow(a, n, arrow(a, n, b, k − 1), 1)
4      else if (n == 0) then
5          return ab
6      else if (b == 0) then
7          return 1
8      else
9          return arrow(a, n − 1, a, b − 1)
10 return arrow(3, 4, 3, 64)
```

Funkcja $arrow(a, n, b, k)$ zwraca liczbę $\underbrace{a \uparrow^n (a \uparrow^n (\ldots (a \uparrow^n b) \ldots))}_{k \text{ strzałek}}$. Przyjrzyjmy się dlaczego tak jest. Linie 4-9 rozpisują definicję notacji strzałkowej, opisanej we wcześniejszym artykule. Co się dokładniej w programie dzieje?

- Linie 2-3 – rozważamy przypadek, gdy liczba strzałek k jest większa niż 1. Wtedy trzeba zrobić jedną strzałkę, a potem jeszcze $k - 1$.
- Zachodzi $a \uparrow^0 b = ab$, co obsługują linie 4-5 oraz $a \uparrow^n 0 = 1$, co obsługują linie 6-7.
- $a \uparrow^n b = \underbrace{a \uparrow^{n-1} (a \uparrow^{n-1} (\ldots (a \uparrow^{n-1} a) \ldots))}_{b-1 \text{ strzałek}}$, co implementują linie 8-9.

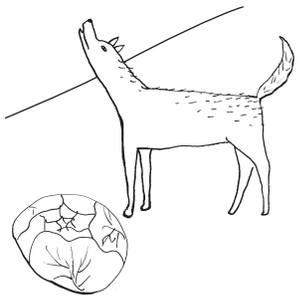
W powyższym programie, po zdefiniowaniu funkcji *arrow*, w ostatniej linii pytamy program o wartość $arrow(3,4,3,64)$, czyli o liczbę Grahama. Program ma dokładnie 201 znaków. A więc wiadomo, że $\text{gener}(201) \geqslant G$ – być może istnieje program o takiej długości zwracający większą liczbę. A co powiecie o $\text{gener}(G)$?

W. C.

Posłowie II: Dowód nierozstrzygalności

Spróbujmy ściśle uzasadnić, że funkcja gener, o której mowa wyżej, jest faktycznie *nieobliczalna*. Bardziej formalnie: chcemy udowodnić, że nie może istnieć program komputerowy P taki, który dla danej na wejściu liczby n obliczy $\text{gener}(n)$.

Nasze rozumowanie prowadzić będziemy nie wprost. Zakładamy więc, że jednak taki program P istnieje. Jego długość (okaże się bardzo ważna) oznaczmy przez k. (To znaczy: *kod* programu P jest zapisany jako ciąg k znaków.)



Rozważmy teraz program (a raczej *szablon* programu) Q_n zdefiniowany następująco:

```
1 Function Z(i)
2 |  kod programu P
3 return Z([wpisana na sztywno stała n])
```

Dla jasności czym jest szablon, napiszmy kod *konkretnego* programu Q_{83}:

```
1 Function Z(i)
2 |  kod programu P
3 return Z(83)
```

Programy z szablonu Q_n są dziwaczne: mają wbudowany program P jako wewnętrzną funkcję Z; nie przyjmują żadnego wejścia oraz w swoim kodzie mają wpisaną „na sztywno” stałą n. Zastanówmy się, jaką długość ma sam program Q_n. W kod Q_n wchodzi kod P, więc tutaj zużywamy k znaków. Poza tym mamy jakąś niewielką liczbę innych znaków (jak „r”, „e”, „t”, „u”, „r”, „n” itd.) oraz stałą, której zapis zużyje $\lceil \log_{10} n \rceil$ znaków „0”, „1”, „2”, . . . , „9”. Łącznie: $k + c + \lceil \log_{10} n \rceil$ znaków dla pewnego $c \in \mathbb{N}$.

Liczba n ma $\lceil \log_{10} n \rceil$ cyfr w zapisie dziesiętnym.

Jak widać, program Q_n oblicza liczbę $P(n)$. A zatem mamy, że $\text{gener}(k + c + \lceil \log_{10} n \rceil) \geqslant P(n)$ oraz, oczywiście, $\text{gener}(n) = P(n)$. Jednak funkcja gener jest niemalejąca, czyli musiałoby być $k + \lceil \log_{10} n \rceil + c \geqslant n$ dla dowolnego n, a to jest, oczywiście, sprzeczność dla dostatecznie dużych n.

T. K.

Posłowie III: Ortografia

W tekście „Naprawdę duże liczby” zostało użyte następujące rozumowanie: *Skoro znaków używanych do napisania programu jest skończenie wiele i skoro program ma ograniczoną długość, to liczb definiowanych przez ten program może być też tylko skończenie wiele, a więc wśród nich musi być największa.* Powstaje pytanie, jakie dodatkowe założenia muszą zostać użyte w tym rozumowaniu, by było ono poprawne i dawało prawdziwy wniosek.

O tym, że takie pytanie ma rację bytu świadczy następujący przykład rozumowania dość podobnego do przytoczonego wyżej: *W języku polskim używamy skończonej liczby liter, więc utworzonych z nich napisów ograniczonej długości jest skończenie wiele. W szczególności skończenie wiele jest napisów używających co najwyżej stu liter. Znacznie mniej (czyli też skończenie wiele) jest wśród nich napisów sensownych, a jeszcze mniej tych, które oznaczają liczbę. Liczb dających się tak zapisać jest zatem skończenie wiele, a więc jest też największa taka liczba.*

Jak to pogodzić z faktem, że napis *liczba co najmniej o jeden większa od każdej, którą można zapisać za pomocą co najwyżej stu liter* ma tylko osiemdziesiąt liter?

Owe dodatkowe założenia użyte w cytowanym tekście to reguły ortograficzne (tak, ortograficzne!), których musimy przestrzegać przy pisaniu programów.

M. K.

Rozwiązanie zadania F 929.
Niech jednej fali odpowiadają drgania wektora pola elektrycznego $\mathbf{E}_1 = \mathbf{E}_1 0 \cos \omega t$, a drugiej $\mathbf{E}_2 = \mathbf{E}_2 0(\cos \omega t + \delta)$, gdzie δ jest różnicą faz między drganiami. Drganie wypadkowe będzie dane wzorem

$$\mathbf{E} = \mathbf{E}_1 + \mathbf{E}_2 = \mathbf{E}_{10} \cos \omega t + \mathbf{E}_{20} \cos(\omega t + \delta),$$

a stąd

$$\mathbf{E}_2 = \mathbf{E}_{10}^2 \cos^2 \omega t + \mathbf{E}_{20}^2 \cos^2(\omega t + \delta) + \\ + 2\mathbf{E}_{10}\mathbf{E}_{20} \cos \omega t \cos(\omega t + \delta).$$

Natężenie drgań będzie równe średniej względem czasu wartości $\mathbf{E}_2$, czyli

$$I = 1/2\mathbf{E}_{10}^2 + \tfrac{1}{2}\mathbf{E}_{20} + \\ + 2\mathbf{E}_{10}\mathbf{E}_{20} \langle \cos \omega t \cos(\omega t + \delta) \rangle,$$

gdzie $\langle\ \rangle$ oznacza średnią względem czasu. Wyrażenie to będzie równe zero niezależnie od wartości przesunięcia fazowego δ jeżeli iloczyn skalarny $\mathbf{E}_{10}\mathbf{E}_{20}$ będzie równy zero, a to oznacza, że kierunki drgań muszą być względem siebie prostopadłe.