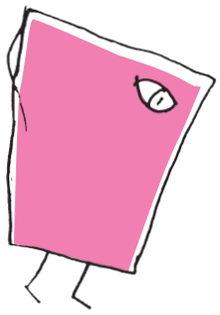
Rozwiązanie zadania F 991.
Jako model serca możemy rozważyć cylinder z tłokiem. Moc takiego urządzenia otrzymamy, mnożąc siłę, z jaką działamy na tłok przez jego prędkość. Siła jest równa iloczynowi ciśnienia p i powierzchni tłoka, a iloczyn powierzchni tłoka przez jego prędkość jest równy objętości krwi pompowanej w jednostce czasu. Otrzymujemy więc:

$$L = nVp \approx 1{,}3\,\mathrm{W}.$$



i matematykę tylko w ramach zbiorów mniej licznych niż ona. Nigdy wtedy jej nie napotkamy! To, czy ona istnieje, nie będzie miało dla nas wtedy żadnego znaczenia. Doprecyzowanie tego rozumowania prowadzi do dowodu, że z aksjomatów nie da się wykazać, że taka liczba istnieje.

W drugą stronę jest jeszcze trudniej. Da się bowiem udowodnić, że w ramach aksjomatów nie da się udowodnić, że nie da się udowodnić, że taka „nieskończoność" nie istnieje! Gdyby bowiem było to możliwe, dawałoby to dowód niesprzeczności aksjomatów, co przeczyłoby twierdzeniu Gödla o niezupełności (patrz odcinek „Rozmyślania o myślakach"). Istnienie takich „nieskończoności" jest więc jeszcze bardziej ulotne niż istnienie nieskończoności pomiędzy licznością liczb naturalnych a licznością liczb rzeczywistych.

W takim razie wszechświat zbiorów wygląda następująco: najmniejszym zbiorem jest zbiór pusty, potem są coraz większe zbiory skończone, aż w końcu najmniejsza nieskończona liczba kardynalna $\aleph_0$, czyli liczność zbioru liczb naturalnych. Potem spotkamy zbiory, których liczności to coraz większe „nieskończoności", które możemy skonstruować, poczynając od $\aleph_0$ i korzystając z metod następnikowej, podzbiorów i sumy. I dalej jest horyzont, za którym być może nic nie ma, a być może jest pierwsza silnie nieosiągalna liczba kardynalna. Z tej „nieskończoności" znów moglibyśmy konstruować coraz większe liczby kardynalne, ale mniejsze niż kolejny hipotetyczny horyzont, którego przekroczyć nie możemy, a który stanowi kolejna silnie nieosiągalna liczba kardynalna. I tak dalej. To jest podróż w nieskończoność i za nieskończoność.

Przygodę tę wypada zakończyć, cytując jednego z bohaterów, od których zaczęliśmy nasze rozważania, czyli Davida Hilberta. „Nieskończoność! Żadne inne pytanie nie poruszyło tak głęboko duszy człowieka".

Złożoność algorytmów teorioliczbowych

Rozważmy następujący algorytm:

```
function czynnik(int n)
{
   int i = 2;
   while (i < n)
   {
      if (n % i) == 0 return i;
      i = i + 1;
   }
   return 0;
}
```

Jak łatwo sprawdzić, podany kod zwraca najmniejszy nietrywialny czynnik podanej na wejściu liczby n, bądź 0 – gdy takiego czynnika nie ma (bo np. n jest liczbą pierwszą).

Zastanówmy się, jaka jest złożoność podanego programu. Pętla `while` wykona co najwyżej $O(n)$ obrotów. W pojedynczym jej wykonaniu niemal wszystkie operacje mają koszt stały, poza operacją `%` (reszta z dzielenia), której koszt możemy bardzo zgrubnie oszacować z góry przez $O(n)$ (w rzeczywistości można łatwo osiągnąć $O(\log(n))$). Oznacza to, że cały algorytm działa w czasie nie większym niż $O(n^2)$.

Czy oznacza to, że właśnie pokazaliśmy wielomianowy algorytm na szukanie nietrywialnych czynników? Rozkład nawet dużych liczb na czynniki pierwsze już nie jest dla nas kłopotem?

Oczywiście, nic z tych rzeczy.

Nieporozumienie bierze się z problemu z ustaleniem, co jest parametrem, względem którego liczymy złożoność programów komputerowych. Otóż parametrem jest dla nas zawsze *rozmiar danych*, ale rozumiany jako długość *napisu* reprezentującego wejście. Konkretniej: 1 000 000 000 ma dla nas rozmiar „dziesięć", a nie „miliard"...

Wracając do naszego algorytmu: jeśli rozmiar danych to k, to wówczas liczba n, którą reprezentują te dane jest rzędu 10^k. Skoro więc oszacowaliśmy czas działania algorytmu przez $O(n^2)$, to w języku rozmiaru danych przekłada się to na $O((10^k)^2) = O(100^k)$, czyli jest jednak wykładniczy od k.

Potencjalny algorytm wielomianowy na rozkład na czynniki musiałby więc być pewnie nieco bardziej wyrafinowany...

Tomasz KAZANA