Angielską wersję tego numeru otrzymali uczestnicy
XVII Międzynarodowej Olimpiady Informatycznej

SPIS TREŚCI
NUMERU 8 (375)

Międzynarodowa Olimpiada Informatyczna dobiega siedemnastki
Stanisław Waligórski str. 2

Zadanie z pierwszej Olimpiady str. 3

Bardzo dziwny algorytm szukania wzorca w słowie
Wojciech Rytter str. 4

Taniec gordyjski
Piotr Chrząstowski str. 6

Zadania str.10

Komputery kwantowe
Jakub Zieliński str.11

Mała Delta:

Ujarzmianie złożoności
Andrzej Walat str.14

Komety, trygonometria i komputer
Grzegorz Sitarski str.18

Samoistne powstawanie korków
Robert Żak str.20

Klub 44 str.22

Patrz w niebo str.24

Sierpień str.24

Kącik biologiczny str.25

W następnym numerze:

Fête de la science

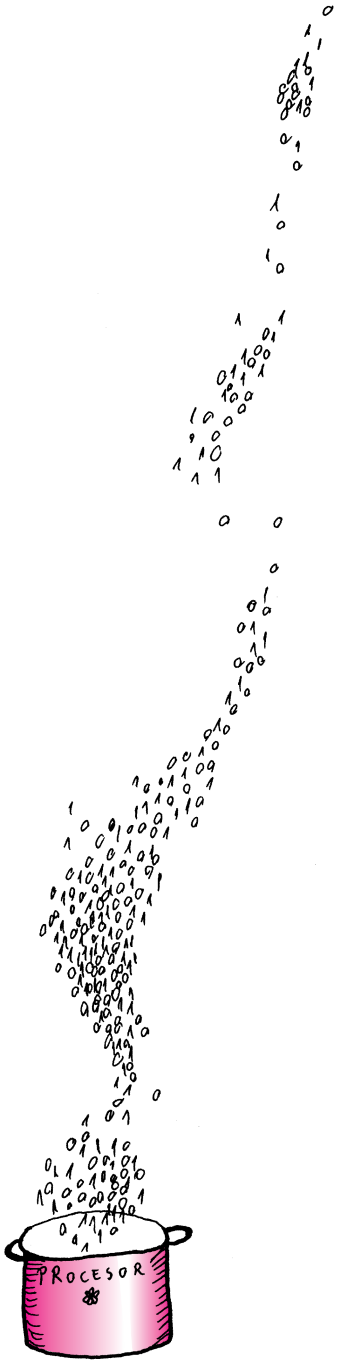
„Delta” – matematyczno-fizyczno-astronomiczny miesięcznik popularny Polskiego Towarzystwa Matematycznego, Polskiego Towarzystwa Fizycznego i Polskiego Towarzystwa Astronomicznego,
wydawany przy poparciu Ministerstwa Edukacji Narodowej i Sportu.

Adres do korespondencji:
Instytut Matematyki UW, Redakcja „Delty”, ul. Banacha 2, pokój 5450, 02-097 Warszawa, e-mail: delta@mimuw.edu.pl, tel.: (22) 55-44-545.

Skład systemem TEX wykonała Redakcja.

Wydrukowano w Drukarni Naukowo-Technicznej S.A. w Warszawie, ul. Mińska 65.

Wydawca: Uniwersytet Warszawski **Cena 1 egzemplarza 4 zł**



Krzysztof Diks, Instytut Informatyki,
Uniwersytet Warszawski

Oddajemy do rąk Czytelników numer *Delty* wydany z okazji XVII Międzynarodowej Olimpiady Informatycznej, która w sierpniu odbywa się w Nowym Sączu. Olimpiada to święto wszystkich młodych informatyków i ich opiekunów. Biorą w niej udział uczniowie szkół średnich, którzy 1 lipca 2005 roku mieli co najwyżej 20 lat. W Nowym Sączu spotykają się reprezentacje kilkudziesięciu krajów z całego świata. Każdy kraj jest reprezentowany przez czterech zawodników-uczniów i dwóch opiekunów. Podczas Olimpiady zawodnicy mają do rozwiązania 6 zadań algorytmiczno-programistycznych, w dwóch pięciogodzinnych sesjach, w każdej po 3 zadania. Rozwiązanie zadania polega na wydobyciu z jego treści rzeczywistego problemu algorytmicznego, ułożeniu stosownego, poprawnego i wydajnego algorytmu oraz zaprogramowaniu go w wybranym języku programowania. Językami programowania dostępnymi podczas Olimpiady są: C, C++ i Pascal. Zawodnicy pracują w środowisku Windows lub Linux. Rozwiązania zawodników są sprawdzane automatycznie, na zestawach testów przygotowywanych przez organizatorów. Za każdy zaliczony test (lub grupę testów) zawodnik uzyskuje pewną liczbę punktów. Zwycięzcą Olimpiady zostaje osoba, która zgromadzi największą liczbę punktów za wszystkie zadania. Jak jest w zwyczaju w tego typu imprezach, połowa uczestników jest nagradzana medalami złotymi, srebrnymi i brązowymi w proporcji 1:2:3. Medaliści to najlepsi młodzi informatycy w świecie w swoim pokoleniu. Przykładem mogą tutaj być reprezentanci Polski, Tomek Czajka, Andrzej Gąsienica-Samek i Krzysztof Onak, którzy byli multimedalistami Międzynarodowych Olimpiad Informatycznych, następnie wygrali prestiżowy konkurs ACM. Są znakomitymi studentami lub pracownikami: Tomek Czajka trzykrotnie zwyciężał w otwartym konkursie TopCoder i jest na studiach doktoranckich w Purdue University, Andrzej Gąsienica-Samek jest dyrektorem laboratorium badawczo-rozwojowego jednej z najlepszych firm informatycznych w Polsce – ComArch (partnera technologicznego tegorocznej Olimpiady), a Krzysztof Onak od września podejmie studia doktoranckie w MIT. Jestem przekonany, że kariery naukowe i zawodowe znakomitej większości medalistów Międzynarodowej Olimpiady Informatycznej potoczyły się (lub potoczą się) równie udanie.

Żeby dać Czytelnikom przedsmak zmagań olimpijskich, przedstawiamy w tym numerze *Delty*, między innymi, teksty czterech osób, które przyczyniły się do popularyzacji informatyki w Polsce i od lat ściśle współpracują z Polską Olimpiadą Informatyczną, są autorami wielu olimpijskich zadań.

Profesor Stanisław Waligórski jest współtwórcą Polskiej Olimpiady Informatycznej, jak i współtwórcą międzynarodowego ruchu olimpijskiego. W swoim artykule wspomina początki Międzynarodowej Olimpiady Informatycznej.

Profesor Wojciech Rytter jest współtwórcą polskiej szkoły algorytmicznej. Jest autorem ponad 150 prac z algorytmiki i autorem 5 podręczników i monografii w tej dziedzinie. W tym numerze opisuje problem ze swojej ulubionej tematyki – przetwarzania tekstów. Czytelnik znajdzie w tym artykule przykład tego, przed jakimi zadaniami stają uczestnicy Polskiej Olimpiady Informatycznej.

Doktor Piotr Chrząstowski to znakomity popularyzator nauki. W swoim artykule pokazuje, w jaki sposób elegancko rozwiązać z pozoru bardzo trudne zadanie, jeśli skorzysta się odpowiednio z (niezbyt trudnego) aparatu matematycznego. Zadanie opisywane przez Piotra pochodzi z programistycznych zawodów internetowych, które wspólnie organizowaliśmy w zeszłym roku.

Doktor Andrzej Walat to znakomity dydaktyk informatyki. Jego artykuł jest przeznaczony dla młodych Czytelników *Delty*. Przedstawia w nim w jaki sposób odpowiednie ustrukturyzowanie zadania algorytmicznego pozwala na uzyskanie wydajnego i poprawnego algorytmu.

Czytelnikom życzę miłej lektury, zachęcam do udziału w konkursach algorytmiczno-programistycznych, a uczestnikom XVII Międzynarodowej Olimpiady Informatycznej życzę sukcesów w zawodach.

Krzysztof DIKS

Międzynarodowa Olimpiada Informatyczna dobiega siedemnastki

Stanisław WALIGÓRSKI

Poprzednia, szesnasta Międzynarodowa Olimpiada Informatyczna (MOI) odbyła się w Atenach od 2 do 9 sierpnia 2004 roku. Wcześniejsze miały miejsce m.in. w stolicy Chin Pekinie (2000), w Tampere w południowej Finlandii (2001), w Yong-In w Korei (2002) i w Wisconsin, USA (2003).

Pomysł MOI przedstawił prof. Błagowest Sendow z Uniwersytetu w Sofii, w latach 1989–1992 przewodniczący IFIP (International Federation for Information Processing) na XXIV Zgromadzeniu Ogólnym UNESCO w Paryżu w październiku 1987 r. UNESCO pomysł zaakceptowała i tak oto I Międzynarodowa Olimpiada Informatyczna rozegrała się w Bułgarii, w miejscowości Pravec, między 16 a 19 maja 1989 roku. Uczestnicy Olimpiady przyjechali z 13 krajów. W drugiej Olimpiadzie, w białoruskim Mińsku, liczba reprezentowanych krajów niemal się podwoiła: było ich już 25.

Zasady

Zgodnie z własnym regulaminem MOI ma charakter corocznych międzynarodowych indywidualnych zawodów informatycznych, do których mogą przystąpić uczestnicy z zaproszonych krajów utrzymujących oficjalne stosunki z ONZ lub z UNESCO – lub które już wcześniej uczestniczyły w MOI.

Wiek zawodnika, liczony 1 lipca roku Olimpiady, nie może przekroczyć 20 lat, a zawodnikiem może być tylko osoba, która była uczniem szkoły średniej w okresie między wrześniem a grudniem poprzedniego roku. Reprezentacja narodowa składa się z co najwyżej czterech zawodników i dwóch dorosłych opiekunów. Ci ostatni, szef i wiceszef delegacji, stają się członkami Zgromadzenia Ogólnego zawodów.

Uczestnicy, a raczej ich sponsorzy, pokrywają sami koszty podróży do miejsca Olimpiady, natomiast organizatorzy biorą na siebie koszty zakwaterowania i utrzymania na miejscu.

Nie ma ogólnie obowiązujących zasad wybierania uczniów, reprezentujących swój kraj na Olimpiadzie, choć zgodnie z regułami MOI podstawowym kryterium powinny być umiejętności. Wiele państw organizuje własne Olimpiady Informatyczne lub podobne konkursy pod innymi nazwami, ale szczegóły związane z ustalaniem składu drużyny narodowej często znacznie się różnią. Odbywa się również kilka regionalnych Olimpiad Informatycznych, a także parę międzynarodowych konkursów rozwiązywania zadań i programowania w Internecie. Każde z tych wydarzeń może w jakiś sposób wpływać na proces przygotowania drużyny olimpijskiej.

MOI stawia sobie różnorakie cele:

- odkrywanie, zachęcanie, uznanie i kontaktowanie ze sobą młodzieży szczególnie uzdolnionej w zakresie informatyki,
- rozwijanie przyjacielskiej międzynarodowej współpracy między ludźmi zajmującymi się informatyką tak na niwie naukowej, jak i edukacyjnej,
- zachęcenie młodych ludzi do zainteresowania się informatyką,
- promowanie idei konkursów informatycznych dla uczniów szkół średnich.

Zazwyczaj MOI trwa 8 dni, wliczając w to dni przyjazdu i wyjazdu. Same zawody zajmują dwa dni, poprzedzone dniem wolnym; także po zawodach przewidziany jest jeden dzień odpoczynku. Każdego dnia konkursowego uczestnicy dostają kilka zadań o charakterze algorytmicznym (zwykle trzy). Rozwiązaniem zadania musi być program napisany w jednym z powszechnie używanych języków programowania, wybranym z dostarczonej uczestnikom listy, ogłaszanej co najmniej rok przed Olimpiadą. Nie oczekuje się od uczniów znajomości jakiegokolwiek innego oprogramowania.

Ocenę wyników nadzoruje Komitet Naukowy MOI. Dokonywana jest ona automatycznie przez specjalnie opracowany program oceniający; wynik każdego

Stanisław Waligórski, Instytut Informatyki, Uniwersytet Warszawski

uczestnika zawodów jest sumą punktów uzyskanych za poszczególne rozwiązania. Gdy zawodnik lub jego opiekun mają zastrzeżenia do proponowanej oceny, mogą je zgłosić na piśmie do jury zawodów. Ostateczną decyzję podejmuje Zgromadzenie Ogólne w głosowaniu, w którym każda z obecnych delegacji narodowych ma jeden głos. Statystycznie co dwunasty olimpijczyk dostaje złoty medal, co szósty – medal srebrny, co czwarty – medal brązowy. Tak więc mniej więcej połowa zawodników wraca do domu z medalem.

Udział IFIP

Komisja Techniczna IFIP ds. Edukacji, TC3, wspierając MOI, ustanowiła w 1993 roku nagrodę przechodnią IFIP, przyznawaną corocznie podczas Olimpiady. Na posiedzeniu, które odbyło się w trakcie MOI '93, międzynarodowe jury postanowiło przyznać nagrodę uczniowi (lub uczniom) z maksymalną łączną punktacją.

Nagrody przypadały uczestnikom z Republiki Czeskiej, Iranu, Rumunii, Szwecji (wszyscy w 1993 r.), ponownie z Republiki Czeskiej (1996). Uczniowie z Rosji: Wiktor Bargaczew i Władimir Martianow, wygrywali dwukrotnie – pierwszy w latach 1993 i 1994, drugi w 1997 r. i w 1998 r. W tym samym roku (1998) nagrodę przyznano także uczniom z Republiki Południowej Afryki, ponownie z Rumunii oraz z Chin. W 1999 r. nagroda trafiła znów do ucznia z Chin, który stał się w ten sposób jednym z jedenastu zdobywców nagrody IFIP w świecie. Z kolei w 2000 r. jedynym zawodnikiem, który uzyskał maksymalną liczbę, a więc komplet punktów, okazał się Rosjanin Michaił Bautin i to jemu, dwunastemu laureatowi w świecie, wręczył puchar przedstawiciel komisji IFIP TC3 przy MOI, Peter Waker.

Potem, po 2000 roku, ceremonii nagradzania już nie powtarzano. Stało się natomiast zwyczajem, że gospodarze Olimpiady wywieszają jej wyniki na swojej stronie internetowej. Na stronie MOI pod adresem http://olympiads.win.tue.nl/ioi/ można znaleźć wiele materiałów dotyczących Międzynarodowej Olimpiady Informatycznej: zadania z minionych zawodów, literaturę zalecaną do samodzielnego przestudiowania, wskazówki dla uczestników, linki do stron sieciowych regionalnych i narodowych Olimpiad Informatycznych ze szczegółowymi informacjami o zawodach itp., a także listę innych międzynarodowych Olimpiad naukowych. Natomiast na stronie http://mimuw.edu.pl/oi/ioires/ jest dostępna lista zdobywców medali na MOI od pierwszej w 1989 roku do końca XX wieku.

W sieci znajduje się mnóstwo materiałów poświęconych innym zawodom informatycznym. Nie wchodząc w szczegóły, warto tu wspomnieć o trzech międzynarodowych konkursach regionalnych, organizowanych w naszych stronach zgodnie z zasadami MOI, choćby dlatego, że stworzyły one własną dobrą tradycję. Oto one: Olimpiada Informatyczna Państw Europy Środkowej (Central European Olympiad in Informatics), Bałtycka Olimpiada Informatyczna (Baltic Olympiad in Informatics), Bałkańska Olimpiada Informatyczna (Balkan Olympiad in Informatics). Zainteresowany Czytelnik z pewnością odnajdzie ich adresy w Internecie.

Zadanie z pierwszej Olimpiady

W pierwszej Międzynarodowej Olimpiadzie Informatycznej w maju 1989 roku było tylko jedno zadanie (a więc także jedyne). Oto ono.

Mamy $2 * N$ pudełek ustawionych w jednym szeregu; dwa sąsiednie są puste, a pozostałe zawierają $N - 1$ liter „A"i $N - 1$ liter „B".

Przykład dla $N = 5$

A	B	B	A			A	B	A	B

Reguła zamiany. Zawartość dowolnych dwu sąsiednich niepustych pudełek można przełożyć, z zachowaniem kolejności, do dwu pustych pudełek.

Cel. Doprowadzić do konfiguracji, w której wszystkie A są na lewo od wszystkich B; położenie pustych pudełek jest dowolne.

Zadanie. Napisać program, który

1. Wczytuje z klawiatury stan początkowy jako sekwencję liter A, B oraz zer (oznaczających pudełka puste) i wykonuje zamiany.
2. Znajduje dla danego stanu początkowego przynajmniej jeden plan zamian prowadzący do celu, albo melduje, że taki plan nie istnieje. Wyniki (dane wyjściowe) powinny zawierać stan początkowy, stany pośrednie po każdym kroku oraz stan końcowy.
3. Znajduje plan osiągnięcia celu w najmniejszej liczbie kroków.

Bardzo dziwny algorytm szukania wzorca w słowie

Wojciech RYTTER

W artykule pokażemy, jak można „przetłumaczyć” pewien fakt matematyczny na operację algorytmiczną. Problem szukania wzorca (w najprostszej postaci) polega na znajdowaniu wszystkich wystąpień zadanego słowa x (traktowanego jako wzorzec) jako podsłowa innego słowa y. Na przykład dla $x = baa$, $y = abaabaaabaa$ wszystkimi wystąpieniami są $abaabaaabaa$, $abaabaaabaa$ i $abaabaaabaa$, co zapisujemy symbolicznie jako $1, 4, 8$, od numerów pozycji, na której zaczyna się powtórzenie (pierwsza pozycja ma numer 0). Zajmiemy się algorytmem, który działa jednocześnie w czasie liniowym (tzn. gdy problem ma rozmiar n, liczba operacji jest ograniczona przez $c \cdot n$, dla pewnej stałej c) i w pamięci stałej, nie licząc pamięci, w której przechowujemy dane wejściowe (dwie tablice x, y) i która nie jest modyfikowalna. Rozmiarem problemu jest długość n (dłuższego) słowa y. Jeśli chodzi o złożoność czasową, to liczyć będziemy tylko liczbę operacji porównywania symboli w x i y. Zaczniemy od wyszukiwania bardzo specjalnych podsłów.

Dla słowa z oznaczmy przez $p = okres(z)$ najmniejszą liczbę $s \geqslant 1$, taką że s jest okresem z, tzn. $z[i] = z[i-s]$ dla wszystkich i, dla których obie strony są zdefiniowane.

Definicja
Słowo nazwiemy leksykograficznie maksymalnym (w skrócie *lm*), gdy jest ono leksykograficznie maksymalne spośród wszystkich swoich sufiksów. Na przykład słowo $'rytter'$ nie jest *lm* (gdyż $'ytter' >' rytter'$), natomiast $'wojciech'$ jest typu *lm*.

Jak widać, termin *sufiks* (i *prefiks*) jest przez informatyków rozumiany szerzej niż przez filologów – jest to dowolny końcowy (początkowy) fragment słowa.

Dlaczego słowa o własności *lm* są interesujące? Większość szybkich algorytmów szukania podsłów korzysta z okresów p prefiksów słowa. Obliczanie tych okresów w ogólnym przypadku jest „wąskim gardłem” w projekcie algorytmu. Natomiast dla słów typu *lm* obliczanie okresów jest trywialne. Wynika to z następującego faktu, którego dowód zostawiamy Czytelnikowi.

Fakt 1 (Kluczowa własność słów typu *lm*)
Niech x będzie słowem typu *lm* i $j \geqslant 0$. Wtedy

$(*)$ $\quad [(j = 0) \vee (p = okres(x[0 \ldots j-1]) \;\&\; x[j] \neq x[j-p])] \Rightarrow okres(x[0 \ldots j]) = j + 1.$

„Przetłumaczenie” implikacji $(*)$ na odpowiednią instrukcję jest zasadniczą częścią algorytmu.

Przykład
Niech $x = bababaa$, $j = 6$. Słowo $bababa$ ma okres 2. Jednak następny symbol *„psuje”* ten okres. Zatem słowo x ma okres równy 7.

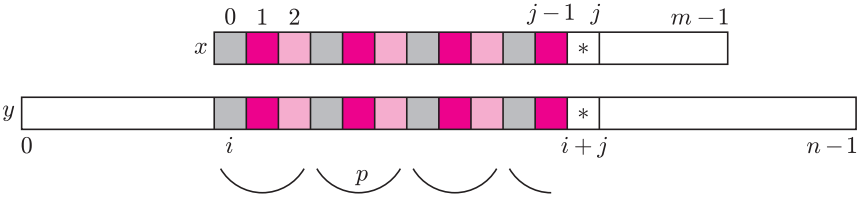
Opiszemy teraz program szukania *lm* podsłowa x w słowie y. Program wczytuje dwa teksty w tablice $x[0 \ldots m-1]$, $y[0 \ldots n-1]$, x jest typu *lm*, a wypisuje wszystkie wystąpienia x w y, tzn. wszystkie takie pozycje i, że $y[i \ldots i+m-1] = x$. Zapisujemy program w języku C++.

Podstawowym niezmiennikiem (patrz rysunek) w programie po każdej iteracji *while* jest:

(A) $j = \max\{j \geqslant 0; j = 0 \text{ lub } x[0 \ldots j-1] = y[i \ldots i+j-1]\}$,

(B) program wypisał wszystkie wcześniejsze wystąpienia $i' < i$,

(C) p jest okresem słowa $x[0 \ldots j-1]$.



Wojciech Rytter, Instytut Informatyki, Uniwersytet Warszawski

Algorytm 1 (dla wzorców x typu lm).

```
#include <iostream.h>
#include <string.h>
#define MAX_LEN 1000
  int i=0,j=0,p=1;
void przesun(void) {
  if (j<2*p) {
    i=i+p; j=0; p=1;
  }
  else {
    j=j-p; i=i+p;
  }
}
void main() {
  char x[MAX_LEN]; char y[MAX_LEN];
  cin>>x>>y;
  while (i <= strlen(y)-strlen(x)) {
    if (j==strlen(x)) {
      cout<<i<<" "; przesun();
    }
    else if (x[j]==y[i+j]) {
      if ((j==0)||(x[j]!=x[j-p])) p=j+1;
      j=j+1;
    }
    else przesun();
  }
}
```

Algorytm sprawdza, czy x zaczyna się od pozycji i-tej. Pewna liczba j symboli wzorca jest zgodna. Następnie algorytm przesuwa wzorzec o wielkość p, która jest okresem segmentu, na którym była zgodność.

Najważniejszą instrukcją w programie jest:

`if ((j==0)||(x[j]!=x[j-p])) p=j+1.`

Instrukcja ta oblicza okres kolejnego prefiksu słowa x, korzystając z Faktu 1 (zapisujemy ten fakt w języku C++). Tak więc poprawność sprowadza się do Faktu 1.

W programie korzystamy ze standardowej funkcji *strlen* zwracającej długość słowa. Operacja $cin \gg x \gg y$ polega na wczytaniu słów x, y do tablic o tych samych nazwach. Operacją dominującą jest porównywanie symboli. Algorytm wykonuje liniową (względem $n = |y|$) liczbę operacji porównywania symboli w słowach – można to udowodnić, obserwując zmiany wartości $2i + j$: wartość ta zwiększa się co najmniej o 1 po każdej iteracji. Jednocześnie $2i + j \leqslant 2n$, gdyż $i + j \leqslant n$.

Algorytm 1 można łatwo zmodyfikować tak, aby znajdował on wystąpienia dowolnego słowa (niekoniecznie typu lm) w czasie liniowym i w stałej pamięci.

Niech $x = uv$, gdzie v jest leksykograficznie maksymalnym sufiksem x. Oznaczmy $r = |u|$. Technicznie informacja o rozkładzie uv sprowadza się do pamiętania r.

Fakt 2
Niech $x = uv$ będzie rozkładem takim, jak wyżej opisany. Wtedy słowo v występuje tylko raz w słowie uv. Jeśli $i' < i$ są początkami wystąpień v, oraz $i - i' < r$, to na pozycji $i - 1$ nie kończy się wystąpienie u.

Z powyższego faktu wynika stosunkowo prosty algorytm szukania x w czasie liniowym i w pamięci stałej. Algorytm ten jest modyfikacją Algorytmu 1, w którym rolę x pełni v.

Algorytm 2

Niech v będzie leksykograficznie maksymalnym sufiksem $x = uv$;

- obliczamy za pomocą Algorytmu 1 kolejne wystąpienia v w y;
- dla każdego wystąpienia i niech i' będzie wystąpieniem poprzednim; jeśli $i - i' \geqslant |u|$, to sprawdź, czy u występuje na lewo od pozycji i; (sprawdzanie to wykonujemy w sposób *naiwny*);
- jeśli występuje, to wypisz kolejne wystąpienie całego wzorca x.

Twierdzenie
Problem szukania słowa x w słowie y można rozwiązać w czasie liniowym i w pamięci (dodatkowej) stałej, jeśli znamy początkową pozycję r leksykograficznie maksymalnego sufiksu v słowa x.

Dekompozycję uv, zapamiętaną na $r = |u|$, można obliczyć w czasie liniowym i w pamięci stałej, korzystając z modyfikacji Algorytmu 1 (odsyłamy Czytelnika po szczegóły do pracy

W. Rytter, *On maximal suffices and constant space versions of KMP algorithm*, Theoretical Computer Science 1-3(302): 211–222 (2003),

z której pochodzą Algorytm 1 i Algorytm 2).

Taniec gordyjski

Piotr CHRZĄSTOWSKI

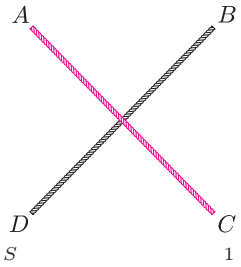
Instytut Informatyki UW od paru lat organizuje otwarte internetowe konkursy programistyczne. Do 2004 roku nazywały się one „Pogromcy Algorytmów”, w tym roku konkurs ten miał nową nazwę „Potyczki Algorytmiczne”. Zadania, z którymi mierzą się zawodnicy, są zróżnicowanej trudności, ale te z ostatniego dnia każdych zawodów są zazwyczaj dość trudne, na poziomie olimpiad informatycznych. W 2004 roku zaproponowałem zadanie, które Komitet Główny Olimpiady Informatycznej uznał za nieco zbyt trudne na olimpiadę i tak stało się ono jednym z zadań finałowej rundy „Pogromców”.

Taniec gordyjski to tradycyjny bajtocki taniec tańczony przez dwie pary tancerzy. Początkowo tancerze stoją w wierzchołkach kwadratu $ABCD$, w dwóch parach: A–B i C–D. Każda z par rozciąga między sobą sznurek. Tak więc na początku oba sznurki są rozciągnięte poziomo i równolegle do siebie (rys. 1).

Rys. 1

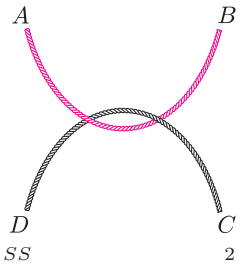
Taniec składa się z ciągu ruchów, z których każdy może być ruchem następującego rodzaju:

- (S) Tancerze stojący w punktach B i C zamieniają się miejscami (nie puszczając swoich sznurków) w ten sposób, że tancerz stojący w punkcie B podnosi rękę ze sznurkiem do góry i idąc do punktu C, przepuszcza tancerza idącego z punktu C do B przed sobą, pod swoją ręką (rys. 2).
- (R) Wszyscy tancerze wykonują obrót o 90 stopni w prawo, nie puszczając sznurków, czyli tancerz, który stał w punkcie A, idzie do punktu B, ten, który stał w punkcie B, idzie do punktu C, ten, który stał w punkcie C, idzie do punktu D, a ten, który stał w punkcie D, idzie do punktu A.



Rys. 2

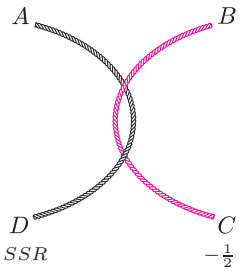
W trakcie tańca sznurki plączą się ze sobą, jednak na koniec tańca powinny zostać rozplątane i znowu być rozciągnięte poziomo i równolegle do siebie. Tancerze nie muszą przy tym stać na tych samych miejscach, na których stali na początku. Taniec ten wymaga od tancerzy dużej wprawy, gdyż w trakcie tańca sznurki mogą być bardzo splątane i ciąg ruchów, który prowadziłby do ich rozplątania i rozciągnięcia poziomo i równolegle do siebie, może być trudny do odgadnięcia.



Rys. 3

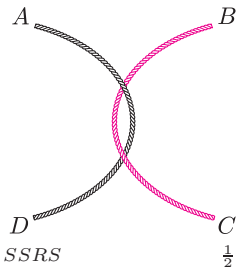
Na podstawie ciągu już wykonanych ruchów Twój program powinien wyznaczyć minimalną liczbę ruchów pozwalających zakończyć taniec.

Przykładowo, po wykonaniu ciągu ruchów SS otrzymujemy konfigurację z rysunku 3.



Rys. 4

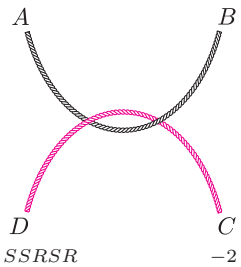
Najkrótszy ciąg ruchów, który pozwala zakończyć taniec, ma długość 5 i jest nim $RSRSS$ (rysunki 4–8).



Rys. 5

Napisz program, który:

- wczyta ze standardowego wejścia opis ciągu wykonanych ruchów w tańcu,
- wyznaczy minimalną liczbę ruchów potrzebnych do rozplątania sznurków i rozciągnięcia ich poziomo i równolegle do siebie (po wykonaniu tych ruchów tancerze nie muszą znajdować się na swoich początkowych pozycjach),
- wypisze wynik na standardowe wyjście.

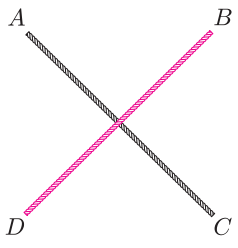


Rys. 6

W pierwszym wierszu standardowego wejścia zapisana jest jedna dodatnia liczba całkowita n równa liczbie wykonanych ruchów, $1 \leqslant n \leqslant 1\,000\,000$. W drugim wierszu zapisane jest jedno słowo długości n złożone z liter S i/lub R. Kolejne litery tego słowa reprezentują kolejno wykonane ruchy w tańcu.

Twój program powinien wypisać na standardowe wyjście w pierwszym i jedynym wierszu jedną liczbę całkowitą – minimalną liczbę ruchów (S i/lub R)

Piotr Chrząstowski, Instytut Informatyki, Uniwersytet Warszawski



Rys. 7

Rys. 7

koniecznych do rozplątania sznurków i rozciągnięcia ich poziomo i równolegle do siebie.

Postępując w sposób opisany w treści zadania, można nieźle zaplątać sznurki. Miłe z kolei jest to, że wykonując w odpowiedniej kolejności wspomniane operacje, zawsze jesteśmy w stanie dojść do stanu wyjściowego, co wcale nie jest oczywiste i co wkrótce wykażemy. Pamiętajmy tu od razu o zastrzeżeniu czynionym w treści zadania, że przy identyfikowaniu węzłów to, kto trzyma sznurki, jest nieistotne, podobnie jak to, który sznurek jest gdzie i w którą stronę jest skierowany. Mówiąc ściślej, *abstrahujemy* od nieistotnych różnic – interesuje nas jedynie rodzaj węzła. Tak samo wyglądające węzły uznajemy za *równoważne* niezależnie od tego, kto, gdzie i jak trzyma końce sznurków. Za to zwracamy uwagę na to, czy węzeł jest ułożony pionowo, czy poziomo.

O klasyfikację wszystkich takich węzłów, jakie zdefiniowano w zadaniu, jak również wielu innych, pokusił się znakomity matematyk John H. Conway (inicjał drugiego imienia jest o tyle istotny, że żyje w tej chwili dwóch matematyków o tym samym pierwszym imieniu i nazwisku – to dla tych, którzy zechcą w Internecie poszukać więcej o działalności „naszego” Conwaya, a warto!). John H. Conway znany jest szeroko jako twórca gry LIFE, najpopularniejszego chyba automatu komórkowego. Jego twierdzenie o charakteryzacji tangli (tak nazwał węzły występujące w naszym zadaniu) jest niezwykłej wprost urody. A że może być podstawą rozwiązania, przedstawimy je tu z radością.

Do tego potrzebne będzie nam pojęcie *izomorfizmu*, jedno z podstawowych pojęć w całej matematyce. Intuicyjnie oznacza ono nierozróżnialność pewnych obiektów, jeśli ograniczymy się do obserwowania, co zadany zestaw operacji (lub ogólniej relacji), które będą określone na obiektach dwóch zbiorów, z tymi obiektami wyczynia. *Algebrą* $\mathcal{A} = \langle A, o_1, \ldots, o_n\rangle$ nazwiemy zbiór A wraz z zestawem operacji $o_1, \ldots, o_n$. Zbiór A nazwiemy *nośnikiem algebry* $\mathcal{A}$. Dwie algebry mogą być różne, mimo że ich nośniki są identyczne. Na przykład algebry $\langle\mathbb{R}, +\rangle, \langle\mathbb{R}, -\rangle, \langle\mathbb{R}, +, *\rangle$ są wszystkie różne. Dla dwóch zadanych algebr dobrze jest wiedzieć, czy zachowują się tak samo. Oczywiście, aby algebry uznać za tak samo się zachowujące, konieczne jest, aby operacji w każdej z nich było tyle samo i żeby odpowiadające sobie operacje miały po tyle samo argumentów. Będziemy też żądali, żeby nośniki algebr były tak samo liczne, czyli żeby istniała funkcja różnowartościowa odwzorowująca jeden zbiór na drugi.

Jeżeli, na przykład, przyjęlibyśmy za nośnik jednej algebry zbiór liczb rzeczywistych dodatnich, a drugiej ujemnych, a w obu algebrach byłaby określona jedna operacja dodawania, to można by znaleźć funkcję $f : \mathbb{R}_+ \to \mathbb{R}_-$, która tak sparuje elementy tych dwóch zbiorów, że wykonanie dodawania w jednym zbiorze da wynik, który ta funkcja przeniesie na wynik dodawania obrazów argumentów. Innymi słowy

$$f(x_1 +_1 x_2) = f(x_1) +_2 f(x_2).$$

Indeksy przy znakach dodawania podkreślają to, że dodawanie odbywa się w innych dziedzinach, choć z punktu widzenia całego zbioru liczb rzeczywistych jest to to samo, co zwykłe dodawanie. Taka funkcja to np. $f(x) = -x$. Równie dobra byłaby zresztą funkcja $g(x) = -2x$. Zauważmy też, że gdybyśmy dołączyli jeszcze operację odejmowania (nie zawsze określoną, bo czasami wynik ucieka poza dziedzinę), to również nie sposób byłoby odróżnić wspomniane algebry od siebie: operacja odejmowania byłaby określona w zbiorze $\mathbb{R}_+$ dla argumentów x_1, x_2 wtedy i tylko wtedy, gdy byłaby określona w zbiorze $\mathbb{R}_-$ dla argumentów $f(x_1), f(x_2)$ i w takim przypadku, podobnie jak dla dodawania, dawałaby izomorficzne wyniki dla izomorficznych argumentów, czyli $f(x_1 -_1 x_2)$ byłoby równe $f(x_1) -_2 f(x_2)$. Ustalając izomorfizm dwóch algebr, musimy wskazać, które elementy nośnika pierwszej algebry odpowiadają którym elementom drugiej (u nas odpowiedniość tę wyznacza np. funkcja $f(x) = -x$) oraz którym operacjom pierwszej algebry odpowiadają które operacje drugiej (u nas operacji $+_1$ odpowiada operacja $+_2$, a operacji $-_1$ operacja $-_2$. Powiemy zatem, że algebry $\langle\mathbb{R}_+, +_1, -_1\rangle$ oraz $\langle\mathbb{R}_-, +_2, -_2\rangle$ są izomorficzne, gdyż ustaliliśmy stosowne odpowiedniości między nośnikami i operacjami obu algebr, które dają odpowiadające sobie wyniki.

Gdybyśmy dołączyli do zbioru operacji w obu algebrach mnożenie, to takie algebry nie byłyby izomorficzne, gdyż w algebrze drugiej mnożenie byłoby operacją niedającą rezultatu dla żadnych argumentów – wynik wykraczałby poza dziedzinę. Wystarczyłoby wówczas zapytać o wynik mnożenia dla odpowiadających sobie elementów, np. w pierwszej algebrze $1 * 2$ byłoby równe 2, natomiast w drugiej $(-1) * (-2)$ byłoby działaniem nieokreślonym, zatem usłyszawszy obie odpowiedzi, wiedzielibyśmy, o której algebrze rozmawiamy. Te algebry byłyby zatem nieizomorficzne.

Tutaj mieliśmy do czynienia z dość prostą funkcją ustalającą izomorfizm. „Światy” tych dwóch algebr były jak gdyby wzajemnym lustrzanym odbiciem względem funkcji $f(x) = -x$. Czasem taki izomorfizm nie jest oczywisty. Zauważmy, że algebry $\langle\mathbb{R}_+, *\rangle$ oraz $\langle\mathbb{R}, +\rangle$

są izomorficzne, a funkcją ustalającą taki izomorfizm jest np. funkcja $f(x) = \log_2 x$. Funkcja logarytm jest bowiem różnowartościowa i „na", a ponadto zachodzi wzór

$$f(x * y) = f(x) + f(y).$$

Zapytajmy jeszcze, po co w ogóle rozważać izomorfizmy. Okazuje się, że nawet gdy dwie algebry są izomorficzne, może się okazać, że wykonywanie operacji w jednej z nich jest technicznie prostsze niż w drugiej. Zatem możemy zyskać możliwość wykonywania operacji w tej z algebr, w której się to robi prościej. Jeśli tylko umielibyśmy w prosty sposób tłumaczyć z nośnika jednej algebry w nośnik drugiej i na odwrót, to mając do wykonania działanie w algebrze „trudniejszej", moglibyśmy przetłumaczyć argumenty za pomocą funkcji f ustalającej odpowiedniość między nośnikami, wykonać prostsze działanie i za pomocą funkcji f^{-1}, odwrotnej do f, uzyskać wynik w oryginalnej algebrze. Pamiętajmy, że funkcja odwrotna zawsze istnieje: wszak f musi być 1-1 i „na".

Wszyscy się chyba zgodzimy, że dodaje się prościej, niż mnoży. Zatem, jeśli umiemy szybko logarytmować i antylogarytmować, to zamiast żmudnego mnożenia moglibyśmy postąpić następująco. Szukając wyniku mnożenia x przez y, znajdujemy ich logarytmy, dodajemy je i szukamy liczby, której logarytmem byłby otrzymany wynik dodawania. Tak zresztą postępowano przez parę wieków, od czasu gdy Napier wynalazł logarytmy i ułożył ich tablice, które pozwalały szybko znajdować zarówno logarytmy, jak i antylogarytmy. Działania te wykonywano w sposób przybliżony, ale do wielu celów wystarczająco dokładny. Na tej zasadzie działał też suwak logarytmiczny – podstawowe narzędzie inżynierów epoki przedkalkulatorowej.

Ten cały wstęp był potrzebny, aby przedstawić rozwiązanie naszego zadania za pomocą tej samej techniki. Zauważmy bowiem, że węzły tworzą algebrę: nośnikiem są wszystkie rodzaje węzłów, a operacjami dwie figury taneczne, które przekształcają węzły w węzły – obie są operacjami jednoargumentowymi. Wykorzystamy tu twierdzenie Conwaya, które skomplikowaną algebrę węzłów „tłumaczy" w prostą algebrę liczb wymiernych z nieskończonością i z dwiema szczególnymi operacjami. Pierwszy krok rozwiązania polega na znalezieniu odpowiedników figur tanecznych w lepiej wyobrażalnej algebrze liczb wymiernych. Rozszerzmy więc zbiór liczb wymiernych o nieskończoność i rozważmy, jako nośnik naszej algebry, zbiór $\mathcal{Q}_\infty = Q \cup \{\infty\}$, gdzie Q jest zbiorem liczb wymiernych. Zbiór $\mathcal{Q}_\infty$ nazwiemy zbiorem liczb superwymiernych. Operacjami superwymiernymi, odpowiadającymi operacjom S i R z algebry węzłów, będą $s(x) = x + 1$ i $r(x) = -\frac{1}{x}$. Zakładamy tu, że $r(0) = \infty$, $r(\infty) = 0$ oraz $s(\infty) = \infty$. Początkowy układ poziomych węzłów odpowiadał będzie liczbie 0. Nasza algebra $\mathcal{Q}_\infty = \langle \mathcal{Q}_\infty, r, s\rangle$ okazuje się być izomorficzna z algebrą węzłów, w której nośnikiem są wszystkie rodzaje węzłów możliwych do otrzymania za pomocą operacji R i S, a te dwie jednoargumentowe operacje są jedynymi rozważanymi. To jest właśnie treścią twierdzenia Conwaya, którego dowodu nie będziemy tu przytaczali. Wykonując w odpowiedniej kolejności operacje s i r, możemy z zera uzyskać dowolną liczbę superwymierną. Wykonując analogiczne ciągi operacji S i R, możemy uzyskać wszystkie węzły zgodnie z treścią zadania. Każdy z tych węzłów może więc uzyskać unikalny numer zwany numerem Conwaya i będący odpowiadającą mu liczbą superwymierną. Różnym węzłom odpowiadają różne numery Conwaya i na odwrót: różnym liczbom superwymiernym odpowiadają różne węzły. Zauważmy przy okazji, że wykonanie dwóch kolejnych operacji r po sobie na liczbach superwymiernych jest identycznością, podobnie jak wykonanie dwóch operacji R na węzłach: dwukrotny obrót o 90 stopni jest symetrią środkową, która nie zmienia węzła, a jedynie zamienia miejscami położenia końców oraz tych, którzy te końce trzymają.

Zobaczmy na kilku przykładach, jak to wszystko działa. Układ początkowy to 0. Wykonanie operacji R prowadzi nas do węzła „nieskończoność", jako że $r(0) = \infty$. Powtórne wykonanie R oznacza powrót do zera, bo $(r(\infty) = 0)$. Zatem węzły całkowicie rozplątane poziome i pionowe odpowiadają liczbom 0 i ∞. Przy okazji: dla układu pionowego wykonanie operacji S nie zmienia go, gdyż $s(\infty) = \infty$.

Jak wygląda sytuacja po wykonaniu pojedynczego S w stanie wyjściowym na węzłach? Ano węzeł, który uzyskujemy, ma numer 1, bo $s(0) = 1$. Zgodnie z twierdzeniem Conwaya tylko jeden węzeł ma numer jeden i aby go rozplątać, należy za pomocą operacji s i r przekształcić jedynkę w zero. Zauważmy, że w tym celu wystarczy wykonać ciąg $r \cdot s$, bo $s(r(1)) = s(-1) = 0$. Łatwo sprawdzić, że faktycznie wykonanie kolejno operacji R, a potem S doprowadzi nas do węzła wyjściowego.

Gdybyśmy wykonali dwa razy S, tak jak w przykładzie z treści zadania, otrzymalibyśmy węzeł o numerze 2. Najprostsza metoda rozwikłania go będzie taka, która za pomocą minimalnej liczby operacji r i s przekształci dwójkę w zero. Zaczynamy więc od

$$2 \xrightarrow{r} -\frac{1}{2} \xrightarrow{s} \frac{1}{2} \xrightarrow{r} -2 \xrightarrow{s} -1 \xrightarrow{s} 0.$$

Ponieważ ciąg operacji $rsrss$ w algebrze liczb superwymiernych przekształca liczbę $2 = s(s(0))$ w 0, więc ciąg operacji $RSRSS$ w algebrze węzłów doprowadzi nas od węzła $S(S(0))$ do punktu wyjściowego.

Teraz rozplątać możemy każdy węzeł, korzystając z izomorfizmu naszych algebr. Mając ciąg zaplątań złożony z operacji S i R, obliczamy numer otrzymanego węzła przez wykonanie analogicznego ciągu operacji s i r na liczbach superwymiernych, a następnie przekształcamy otrzymany numer możliwie krótką

sekwencją operacji s i r doprowadzającą do zera. Rozwiązaniem będzie długość tej sekwencji.

Sprowadziliśmy zatem nasz problem do następującego: jak mając daną liczbę superwymierną, możliwie szybko uzyskać zero za pomocą operacji s i r? Zachłanny algorytm, który liczby dodatnie będzie natychmiast zamieniał na ujemne za pomocą operacji r, a na liczbach ujemnych będzie wykonywał operację s tak długo, aż otrzymamy liczbę nieujemną, okazuje się być tu optymalny. Wiadomo, że z wyjątkiem sytuacji, gdy zaczynamy od węzła ∞, czyli pionowych sznurków, ostatnim ruchem rozwiązania musi być s. Zatem będziemy atakowali zero od dołu: naszym celem powinno być osiągnięcie w miarę szybko ujemnej liczby całkowitej, a następnie wykonanie odpowiedniej liczby s-ów. Zrobimy to w taki sposób, że mianowniki kolejnych liczb ujemnych, powstałych po wykonaniu operacji r, będą malały do jedności. Ostatnia liczba ułamkowa dodatnia w rozwiązaniu powinna być postaci $\frac{1}{k}$ dla pewnego k i końcówka algorytmu będzie wyglądała tak, że po uzyskaniu $-k$ wykonamy k-krotnie operację s. Z kolei liczbę $\frac{1}{k}$ można uzyskać tylko za pomocą operacji s (inaczej bezsensownie wyskoczylibyśmy z „dobrej" ujemnej liczby całkowitej). Niżej pokażemy, że podana metoda daje zawsze minimalną liczbę ruchów rozplątującą węzeł.

Możemy więc zastosować następujący algorytm. Niech l/m będzie liczbą wymierną lub minus nieskończonością reprezentującą zaplątanie sznurków.

1. `if` $l = 0$ `then exit`

2. `else if` $(l/m > 0) \vee (m = 0)$ `then` $(l, m) := (-m, l)$ // (operacja R)

3. `else` $l := l \bmod m$ //(ciąg operacji S)

Udowodnimy, że ten algorytm prowadzi do rozplątania sznurków za pomocą minimalnej liczby ruchów.

Wzór 1. Dla liczb naturalnych $l > 0$ i $m > 0$

$$l = (-m) \bmod (l + m) \tag{1}$$

Dowód wzoru 1

$$l = (-m) \bmod (l+m) = -m - \left\lfloor \frac{-m}{l+m} \right\rfloor (l+m)$$

$$l + m = -\left\lfloor \frac{-m}{l+m} \right\rfloor (l+m)$$

$$1 = -\left\lfloor \frac{-m}{l+m} \right\rfloor$$

$$1 = \left\lceil \frac{m}{l+m} \right\rceil$$

Lemat 1. Algorytm się zatrzymuje.

Dowód lematu 1. Wykażemy, że dla dowolnej liczby wymiernej ujemnej postaci $-\frac{l}{m}$, dla $l, m > 0$, wykonywanie kolejnych kroków algorytmu doprowadzi nas w końcu do wartości $x = 0$. Jeśli $m = 1$, to po l-krotnym wykonaniu s dostaniemy zero. Jeśli zaś $m > 1$, to wykonując kroki algorytmu, dojdziemy do innej liczby wymiernej $-l'/m'$, takiej że $m' < m$. Niech bowiem $x = -l/m$ oraz $l > 0$ i $m > 1$. Dla $x < 0$ wykonujemy ciąg operacji s i po jego wykonaniu dojdziemy do sytuacji, gdy x po raz pierwszy stanie się większe od zera. Niech wtedy $x = l_1/m$, gdzie $m > l_1 > 0$. Teraz wykonujemy R i mamy $x = -m/l_1$. Mianownik zmalał, a ułamek jest ujemny. Zatem zawsze po wykonaniu ciągu S, aż do uzyskania liczby dodatniej, zakończonego jednym R, dostaniemy ułamek ujemny o mniejszym mianowniku. Nie można w nieskończoność zmniejszać dodatniego mianownika. Zatem po skończonej liczbie kroków mianownik stanie się równy 1, a to, jak wiemy, doprowadza do zatrzymania całego algorytmu.

Lemat 2. Algorytm wykonuje minimalną liczbę kroków.

Dowód lematu 2. Możliwe są następujące ruchy dla $l \neq 0, m \geqslant 0$.

- $m = 0$ i S – zły oczywiście, bo nie zmienia węzła,
- $m = 0$ i R – dobry, bo w jednym ruchu doprowadza nas do zera. Szybciej nie można.
- $l, m > 0$ i S – zły.

Wykonajmy S, a potem tak, jak każe algorytm, i pokażmy, że ta sekwencja nie jest najkrótsza. Mamy bowiem

$$\begin{aligned} l/m &\to (l+m)/m \to \\ &\to -m/(l+m) \to \\ &\to ((-m) \bmod (l+m))/(l+m) \overset{\text{(ze wzoru (1))}}{=} \\ &= l/(l+m) \to \\ &\to -(l+m)/l \to \\ &\to ((-(l+m)) \bmod l)/l = \\ &= ((-m) \bmod l)/l, \end{aligned}$$

czyli 5 kroków. Lepszą sekwencją, po której dochodzimy do tej samej liczby, jest

$$l/m \to -m/l \to ((-m) \bmod l),$$

co wymaga 2 kroków.

- $l, m > 0$ i R – dobry,
- $m > 0,\ l < 0$ i S – dobry,
- $m > 0,\ l < 0$ i R – zły.

Niech bowiem $l_1 = -l > 0$, po wykonaniu R mamy $x = m/l_1$. Teraz wykonanie R nie ma sensu, gdyż wrócimy do stanu poprzedniego, wykonanie S jest złe $(m/l_1 > 0)$.

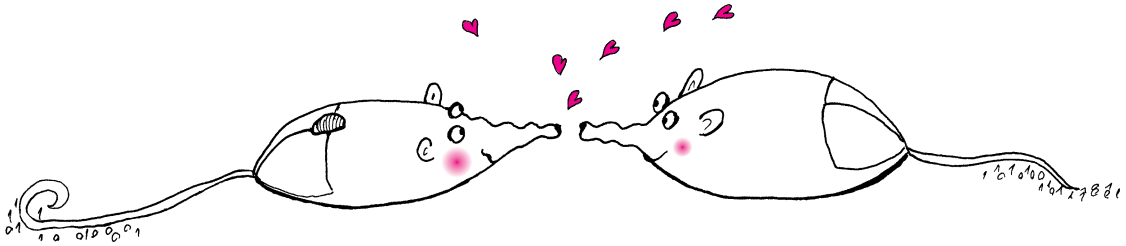
Zatem wystarczy jeszcze wykazać, że jakikolwiek zły krok nie prowadzi do optymalnego rozwiązania. Jeśli będziemy wykonywać tylko S dla liczby dodatniej, to nigdy nie dojdziemy do rozplątania, bo będziemy zwiększać liczbę dodatnią i nie osiągniemy zera. Załóżmy więc, że po złym kroku S (dla liczby dodatniej) kiedyś wykonamy R, czyli tak jak nakazuje algorytm. Wykonanie dobrego kroku po złym to już

przypadek rozpatrywany wcześniej – zawsze można lepiej wyjść na niewykonywaniu tego ostatniego S, tylko pójściu na skróty. Jeśli więc S było wykonane dla liczby dodatniej, to tylko zwiększyło niepotrzebnie liczbę kroków konieczną do rozplątania. Dla operacji R jest jeszcze łatwiej, gdyż R^{2k} nic nie daje, a $R^{2k+1} = R$. Sensowne jest zatem wykonywanie jedynie pojedynczych R przetykanych S-ami. Podsumowując: R-y należy wykonywać tylko jednokrotnie, a S-y tylko dla liczb ujemnych. I tak właśnie działa nasz algorytm, więc dochodzi najszybciej do stanu końcowego.

Wiemy już, jak dojść do rozwiązania końcowego, pozostaje więc zliczyć, ile potrzebujemy ruchów. Jeśli algorytm wykonuje operację R, to zwiększamy licznik o 1, gdy zaś wykonuje S (właściwie ciąg operacji S), dodajemy do licznika $||\frac{l}{m}]|$ (ułamek jest ujemny). Rzecz jasna, zakresy danych wymuszały implementację własnej arytmetyki umożliwiającej wykonanie tych działań.

Przy wykorzystywaniu dowodu lematu 2 widać, iż podczas zaplątywania operacja S zwiększa liczbę kroków algorytmu o co najwyżej 3. Liczba operacji S w ciągu wejściowym jest oczywiście nie większa niż jego długość. Złożoność czasowa algorytmu wynosi więc $O(n)$, gdzie n jest długością ciągu podanego na wejściu.

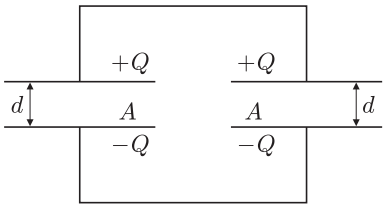
Zadanie to pokazuje, jak przywykli do myślenia abstrakcyjnego (liczby są wszak abstrakcyjne), a nie przywykli do plątania realnych węzłów, lepiej poruszamy się w świecie abstrakcji, choć przecież mogłoby być zupełnie odwrotnie: jakieś plemię biegłe w tańcach gordyjskich mogłoby tłumaczyć dzieciom liczby wymierne w drugą stronę. Patrz synku! Liczba $\frac{1}{2}$ to tak, jakbyś zrobił $SSRS\ldots$



Zadania

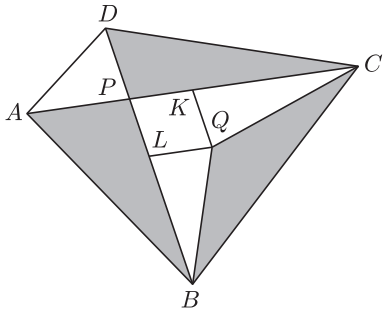
Redaguje Mikołaj KORZYŃSKI

F 649. Zasada zachowania energii oraz pędu zabrania zamiany fotonu w parę elektron – pozyton. Taka przemiana jest jednak możliwa, jeśli w reakcji uczestniczy inna, ciężka cząstka o masie M. Jaka minimalna energia fotonu jest potrzebna, by reakcja zaszła? Masę elektronu m_e traktować jako daną.
Rozwiązanie na str. 22



Rys. 1

F 650. Dwa identyczne kondensatory płaskie o powierzchni płytek A i ładunku Q oraz odległości między płytkami d połączone są jak na rysunku 1. Zmniejszamy odległość między płytkami jednego z kondensatorów. Jak zmienia się energia całkowita układu? Porównać z sytuacją pojedynczego naładowanego kondensatora. Ile wynosi siła przyciągania okładek kondensatora?
Rozwiązanie na str. 22



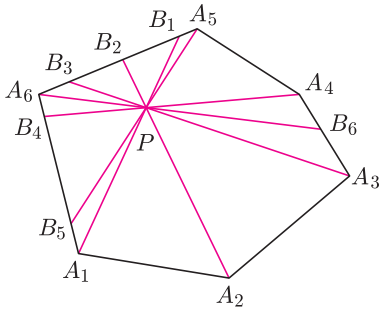
Rys. 2

Redaguje Waldemar POMPE

M 1105. Przekątne AC i BD czworokąta wypukłego $ABCD$ przecinają się w punkcie P. Punkty K i L są odpowiednio środkami przekątnych AC i BD (rys. 2). Punkt Q jest takim punktem, że czworokąt $KPLQ$ jest równoległobokiem. Udowodnić, że

$$[ABP] + [CDP] = 2\,[BCQ]\,,$$

gdzie $[XYZ]$ oznacza pole trójkąta XYZ.
Rozwiązanie na str. 18



Rys. 3

M 1106. Danych jest dziesięć liczb naturalnych dwucyfrowych. Dowieść, że spośród tych liczb można wyłonić dwa takie różne podzbiory, że sumy liczb w obu podzbiorach są jednakowe.
Rozwiązanie na str. 22

M 1107. Punkt P leży wewnątrz $(2n)$-kąta wypukłego $A_1A_2\ldots A_{2n}$ i nie należy do żadnej z jego przekątnych. Proste $A_1P, A_2P, \ldots, A_{2n}P$ przecinają obwód danego wielokąta po raz drugi w punktach $B_1, B_2, \ldots, B_{2n}$ (zob. rys. 3 dla $n = 3$). Wykazać, że na pewnym boku danego $(2n)$-kąta nie leży żaden z punktów B_i.
Rozwiązanie na str. 24

Komputery kwantowe

Jakub ZIELIŃSKI

Wszyscy wiemy, że wiele problemów nie daje się rozwiązać za pomocą komputera tylko z powodu ich zbyt dużej „złożoności obliczeniowej". Pod tym poważnym stwierdzeniem kryje się prosta i smutna prawda. Nawet najszybsze komputery są zbyt wolne, aby uporać się z niejednym zadaniem. Wiemy też, że nie da się w nieskończoność zwiększać szybkości komputerów. Rozmiary atomów wyznaczają możliwą do wyobrażenia skalę miniaturyzacji. Jako że żaden sygnał nie może rozchodzić się szybciej niż światło w próżni, czas potrzebny na przesłanie informacji między fragmentami procesora też jest ograniczony od dołu. Czy komputer kwantowy może stać się remedium na powyższy problem?

Pierwszy raz z nazwą „komputer kwantowy" zetknąłem się 9 lat temu, na pierwszym roku studiów. Wydała mi się nieco podejrzana. Przecież zasada nieoznaczoności Heisenberga wyklucza dokładne pomiary obiektów kwantowych, powinna więc uniemożliwiać działanie takiego urządzenia. Tak naprawdę to wymienione ograniczenia są, obok gwałtownie rosnącego zużycia energii, główną przeszkodą przy tworzeniu coraz mniejszych i szybszych komputerów „klasycznych". Celowo użyłem cudzysłowu, bowiem w obecnie produkowanych procesorach prawa mechaniki kwantowej odgrywają już znaczącą rolę – tak naprawdę trudno opisać klasycznie działanie pojedynczego tranzystora...

Na czym więc ma polegać owa kwantowość? Na zupełnie odmiennym sposobie przetwarzania informacji. We współczesnych komputerach informacja jest zapamiętywana jako ciąg bitów, czyli zer i jedynek. Obliczenia zatem sprowadzają się do odpowiedniego zamieniania jednych bitów na inne.

W komputerze kwantowym bity są zastąpione przez qubity (quantum bits). Każdy qubit Ψ może być w tak zwanej superpozycji pomiędzy zerem i jedynką. Zapisujemy to w ten sposób:

$$|\Psi\rangle = \alpha|0\rangle + \beta|1\rangle.$$

Jeśli dokonamy pomiaru wartości qubitu, to możemy otrzymać $|0\rangle$ lub $|1\rangle$ z prawdopodobieństwami równymi odpowiednio: $P_0 = |\alpha|^2$ oraz $P_1 = |\beta|^2$. Przykładem tu może być elektron ze spinem skierowanym w górę lub w dół.

Takie uogólnienie bitu można jednak osiągnąć klasycznie. Wystarczy zbudować klasyczny komputer analogowy. Każdy bit będzie przybierał wartości z przedziału $[0, 1]$ i gotowe.

Nie chodzi jednak o operowanie wartościami między 0 i 1, lecz o wykorzystanie praw mechaniki kwantowej, pozwalających na superpozycję stanów jednego lub więcej qubitów.

Zobaczmy, jak to wygląda na przykładzie dwóch qubitów. Najprostsza baza układu dwóch qubitów to oczywiście: $|0\rangle|0\rangle$, $|0\rangle|1\rangle$, $|1\rangle|0\rangle$ oraz $|1\rangle|1\rangle$. Przykładowy stan to pewna kombinacja liniowa wektorów bazy; np.

$$|\Psi\rangle = \frac{1}{\sqrt{2}}(|0\rangle|1\rangle - |1\rangle|0\rangle).$$

Pierwszy qubit może być zerem lub jedynką. Obie możliwości są jednakowo prawdopodobne. Drugi tak samo. Wiemy jednak, że zawsze będą miały przeciwne wartości. Zaczyna robić się ciekawie, ale po chwili zastanowienia możemy zaproponować klasyczny odpowiednik tej sytuacji. Weźmy dwie kule: białą i czarną. Włóżmy je, nie patrząc, do dwóch kieszeni. Nie wiemy jednak, gdzie znajduje się któraś z nich, ale wiemy, że kule w kieszeniach są różne.

Aby pokazać różnicę między mechaniką kwantową a klasyczną statystyką, zamienię oznaczenia. Zamiast $|0\rangle$ będę pisał $|\downarrow\rangle$, natomiast $|1\rangle$ zastąpię przez $|\uparrow\rangle$. Wprowadzę również dwa nowe stany, będące superpozycjami powyższych: $|\leftarrow\rangle = \frac{1}{\sqrt{2}}(|\uparrow\rangle + |\downarrow\rangle)$ oraz $|\rightarrow\rangle = \frac{1}{\sqrt{2}}(|\uparrow\rangle - |\downarrow\rangle)$. Są to stany spinu skierowanego odpowiednio w lewo i prawo. Prosty rachunek (zachęcam!) pokazuje:

$$|\Psi\rangle = \frac{1}{\sqrt{2}}(|\downarrow\rangle|\uparrow\rangle - |\uparrow\rangle|\downarrow\rangle) = \frac{1}{\sqrt{2}}(|\leftarrow\rangle|\rightarrow\rangle - |\rightarrow\rangle|\leftarrow\rangle).$$

Wynik jest dość dziwny. Wygląda bowiem na to, że elektrony „nie wiedzą", czy zostały „złożone" z elektronów o spinach góra-dół, czy też lewo-prawo. Co gorsza, my też tego nie wiemy! To tak, jakbyśmy po sprawdzeniu, że kulka z lewej kieszeni jest czerwona, wiedzieli, że ta z prawej jest zielona. Co więcej, moglibyśmy wybrać dowolną parę kolorów dopełniających. Jest tylko jedno ograniczenie: takiego „magicznego" pomiaru możemy dokonać tylko raz. Po pierwszym pomiarze owo połączenie czy też „splątanie" – jak mawiają fizycy – pomiędzy elektronami znika. Każdy z nich ma już określony stan.

Owa przedziwna własność stanów splątanych jest kluczowa – jest tym, co naprawdę odróżnia komputer kwantowy od klasycznego. Aby ją lepiej zrozumieć, musimy najpierw dowiedzieć się, na czym polega pomiar w mechanice kwantowej.

Nie możemy „zapytać", jak skierowany jest spin elektronu. Możemy spytać się jedynie, czy jest skierowany w górę czy w dół. Możemy także wybrać inny pomiar: w lewo czy w prawo albo w przód czy w tył. Ale nie możemy zadać tych pytań naraz. Tego typu zakazy wynikają z zasady nieoznaczoności. Ograniczenie powyższe nie wynika z niedoskonałości przyrządów pomiarowych. Popatrzmy na definicję spinów skierowanych w bok. Widać, że elektron o spinie skierowanym w prawo niejako składa się z elektronów o spinach skierowanych w górę i w dół. Zatem jeśli mamy przyrząd, który rozróżnia elektrony $|\uparrow\rangle$ od $|\downarrow\rangle$, to elektron $|\rightarrow\rangle$ „musi się zdecydować" na jedną

Jakub Zieliński, Instytut Fizyki Teoretycznej, Uniwersytet Warszawski

z tych możliwości. Nie mamy i mieć nie możemy na to wpływu. Każde urządzenie, które wykrywa wszystkie elektrony $|\uparrow\rangle$, musi wykryć co najmniej połowę elektronów $|\rightarrow\rangle$. To, czy wykryje połowę czy więcej, zależy tylko od tego, jak reaguje na elektrony $|\downarrow\rangle$.

Budując algorytmy kwantowe, mamy do dyspozycji dodatkowe stany. Wydawać by się mogło, że z łatwością powinno dać się skonstruować mnóstwo algorytmów znacznie efektywniejszych niż klasyczne. Problemem jest jednak brak możliwości rozróżniania dowolnych stanów. Co z tego, że otrzymamy wspaniały rezultat, jeśli nie będziemy go w stanie odczytać. Dlatego obecnie znamy tak naprawdę tylko dwa algorytmy kwantowe.

Pierwszym jest algorytm Shora. Nie będę tutaj tłumaczył całego algorytmu, gdyż jego opis jest długi i raczej żmudny. Pozwala on rozłożyć liczbę na czynniki pierwsze. Jak wiadomo, jest to kluczowy element łamania tak zwanych szyfrów z kluczem publicznym RSA (używanych powszechnie do szyfrowania informacji w Internecie, np. przez banki). Nie ma wątpliwości, że to jedna z ważniejszych przyczyn zainteresowania informatyką kwantową.

W algorytmie zaproponowanym przez Shora najpierw sprowadza się problem rozłożenia liczby N na czynniki pierwsze do problemu znalezienia okresu pewnej funkcji. Jest ona zadana dla N wartości. Procedura ta jest w pełni klasyczna – pomijam ją tutaj. Po szczegóły odsyłam do oryginalnej pracy [1]. Standardową metodą znajdywania okresu funkcji jest obliczenie jej transformaty Fouriera. W wyniku tej operacji znajdujemy tak zwane widmo. Gdyby funkcja opisywała falę dźwiękową, to dzięki transformacji Fouriera dostalibyśmy informację, jakie tony zawiera dźwięk. Podobnie, dla światła, byłby to rozkład na fale płaskie o określonych częstościach (czyli barwach). Transformacja Fouriera pozwala po prostu wyrazić funkcję przez szereg sinusów i kosinusów o różnych okresach. W wyniku transformacji funkcji, określonej dla N punktów, dostajemy funkcję (również N-punktową), której każdy punkt dostarcza informacji, „ile jest danej częstości".

I tak oto dostajemy wynik – rozkład liczby na czynniki pierwsze. Zaniepokoić powinny nas dwie rzeczy. Po pierwsze, transformatę Fouriera umiemy obliczać klasycznie – gdzie więc jest mechanika kwantowa? Po drugie, czas potrzebny na jej obliczenie jest proporcjonalny do $N \log N$. Widać więc, że jest to fatalny algorytm. Najprostszy pomysł na rozłożenie N to sprawdzać wszystkie liczby od 2 do $\sqrt{N}$.

Rzecz w tym, że transformatę Fouriera można obliczyć bardzo szybko kwantowo. Realizuje to odpowiedni układ „bramek kwantowych", czyli po prostu można wykonać stosowne obliczenia za pomocą manipulacji na spinach elektronów. Jak każde rachunki, można je przeprowadzić na różne sposoby. Jednak niezależnie od tego, jak byśmy się starali, zawsze musimy uwzględnić operacje „splątujące", a więc takie, które z dwóch nieskorelowanych elektronów zrobią splątaną parę. Stanowi to ogromną trudność eksperymentalną.

A jak duży musi być komputer kwantowy, aby poradzić sobie z rozkładem liczby N? Przede wszystkim liczba ta musi zmieścić się w jego „pamięci". Oznacza to, że $N < 2^n$, gdzie n jest właśnie liczbą qubitów. To bardzo ważne, bowiem w wielu przypadkach obowiązuje „zasada zachowania trudności": zwiększenie szybkości algorytmu osiągane jest za cenę większej „pamięciożerności".

Rzeczywiście, transformację Fouriera możemy wykonać bardzo łatwo, kodując funkcję za pomocą fali elektromagnetycznej, a widmo uzyskać natychmiast, przepuszczając światło przez pryzmat lub siatkę dyfrakcyjną. Problemem jest tu jednak ogromna liczba częstości, jakie musielibyśmy rozróżniać. Liczby pierwsze stosowane w kryptografii mają często po kilkaset cyfr. Nie ma najmniejszych szans na rozróżnianie częstości światła z tak ogromną precyzją. W algorytmach klasycznych czas potrzebny na rozłożenie liczby rośnie wykładniczo z liczbą jej cyfr (jej logarytmem). Przy zastosowaniu algorytmu Shora i optyki klasycznej wykładniczo rośnie liczba potrzebnych częstości światła. W algorytmie kwantowym, dzięki splątaniu, czas obliczeń i ilość qubitów rosną potęgowo wraz z logarytmem rozkładanej liczby.

Widać więc, że algorytm Shora to w zasadzie szukanie okresu funkcji. Nie jest to raczej problem fascynujący sam w sobie. Zawdzięcza on swoją popularność temu, że Shor potrafił powiązać ten pozornie czysto akademicki problem z kryptografią. To uczy, że bardzo ryzykowne jest „spisywanie na straty" jakiegoś działu nauki, mówiąc, że „to się do niczego nie przyda".

Na koniec jedna uwaga. Dlaczego upieram się, że komputer kwantowy ma szukać okresu funkcji, a nie po prostu obliczyć transformatę Fouriera? Ten ostatni problem jest ważny w niezliczonych zastosowaniach. Od diagnostyki medycznej, poprzez analizę zdjęć satelitarnych, na symulacjach układów chemicznych kończąc. Otóż komputer kwantowy potrafi obliczyć transformatę Fouriera zawsze, ale wynik możemy odczytać tylko jeśli funkcja jest okresowa. Powód jest prosty: tylko dla funkcji okresowej widmo zawiera niewiele częstości. Jeśli będziemy próbować obliczać transformatę Fouriera z funkcji nieokresowej, to otrzymamy widmo złożone z niezliczonej ilości długości fal. A podczas pomiaru możemy dostać tylko jedną wartość częstości...

Drugim zaproponowanym algorytmem kwantowym [2] jest sposób przeszukania zbioru N-elementowego w poszukiwaniu jednego „dobrego" elementu. W wielu

przypadkach jest tak, że łatwo sprawdzić, czy zaproponowane rozwiązanie jest dobre. Trudno je jednak znaleźć. Często pozostaje nam jedynie sprawdzanie po kolei wszystkich możliwych przypadków. Kłopot zaczyna się, gdy jest ich dużo.

Klasyczny średni czas szukania jednego „dobrego" elementu spośród N jest proporcjonalny do $\frac{N}{2}$. Czas potrzebny dla komputera kwantowego to tylko $\sqrt{N}$.

Wszystkie N sprawdzanych odpowiedzi możemy ponumerować. Od tej pory będziemy sprawdzać, która z N liczb jest „dobra". N liczb zakodujemy za pomocą 2^n qubitów. Pokażę to na prostym przykładzie $N = 4$ ($N_i = \{0, 1, 2, 3\}$):

$$|0\rangle = |00\rangle, \;\; |1\rangle = |01\rangle, \;\; |2\rangle = |10\rangle, \;\; |3\rangle = |11\rangle.$$

Kodowanie jest więc takie, jak w komputerze klasycznym. Następnie tworzymy stan będący superpozycją wszystkich możliwych odpowiedzi:

$$|\Psi\rangle = \frac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle).$$

Zwracam uwagę, że jest to stan bardzo silnie splątany. Sprawdzanie, czy dana liczba jest dobra, można zdefiniować jako obliczenie funkcji. Jeśli liczba jest dobra, to funkcja ma wartość 1, a jeśli zła, to -1. Oczywiście, możemy przypisać inne wartości obu odpowiedziom – byleby tylko były różne.

Widać więc, że wszystkie $N - 1$ złych stanów będzie zawsze traktowane tak samo, a inaczej jeden stan dobry. Możemy więc ogólnie zapisać

$$|\Psi\rangle = \sqrt{\frac{1}{N}}(|0\rangle + |1\rangle + \ldots + |N\rangle) =$$

$$= \sqrt{\frac{1}{N}}(\sqrt{N-1}|\text{zły}\rangle + |\text{dobry}\rangle).$$

Jak widać, jeśli zmierzymy, w którym stanie jest układ N qubitów, to z prawdopodobieństwem $\frac{N-1}{N}$ dostaniemy „złą" odpowiedź, a „dobrą" jedynie z prawdopodobieństwem $\frac{1}{N}$. Jest to dość oczywiste – nic jeszcze nie zrobiliśmy i szansa wylosowania stanu o szukanym numerze jest znikoma.

Zwróćmy uwagę, że otrzymany stan jest nieco podobny do pojedynczego spinu. Jeśli przyjmiemy

$$\alpha = \sqrt{\frac{N-1}{N}} \;\; \text{oraz} \;\; \beta = \sqrt{\frac{1}{N}},$$

to otrzymamy:

$$|\Psi| = \alpha|0\rangle + \beta|1\rangle,$$

a więc spin skierowany prawie idealnie do dołu. Spin możemy obrócić o 180 stopni i dostaniemy elektron o spinie skierowanym niemal idealnie do góry. Obrotu pojedynczego spinu można dokonać, np. włączając jednorodne pole magnetyczne lub pole magnetyczne połączone z falą elektromagnetyczną. Jest to technika powszechnie stosowana w obrazowaniu i spektroskopii, opartych na zjawisku rezonansu magnetycznego (zazwyczaj spiny pochodzą tam od jąder, a nie od elektronów – jest to dla nas bez znaczenia. Elektron jest tylko przykładem cząstki ze spinem. Do budowy komputera można użyć innych cząstek).

Oczywiście, w przypadku układu N spinów nie jest tak prosto. Co więcej, problemem jest to, że nie znamy dokładnie stanu $|\Psi\rangle$. Po prostu nie wiemy, co oznacza $|\text{dobry}\rangle$, gdyż to jest nasza niewiadoma. Ale znamy nasz stan z bardzo dobrym przybliżeniem, dlatego też możemy z niezłym przybliżeniem go „obrócić". Zajmuje to $\sqrt{N}$ kroków. Niestety, nie możemy obrócić układu w jednym kroku. Musimy „drobić", bo gdybyśmy „robili zbyt duże kroki", moglibyśmy pójść nieco za daleko. To cena, jaką płacimy za niedokładną znajomość stanu początkowego.

Niestety, pomimo ogromnego wysiłku, nie udało się dotąd zbudować działającego komputera kwantowego i nie wiadomo, czy będzie to kiedykolwiek możliwe. Jak już wspomniałem, nie musimy koniecznie używać elektronów. Próbowano rozmaitych schematów, w tym jądrowego rezonansu magnetycznego.

W tej chwili najbardziej obiecujące wydają się próby z jonami umieszczonymi w pułapce [3, 4]. Można już w sposób kontrolowany umieścić jeden za drugim kilka jonów w próżni. Ich pozycja jest kontrolowana za pomocą odpowiednio uformowanego pola elektromagnetycznego. Co więcej, możliwe jest dokonywanie manipulacji na wybranym jonie oraz ich splątywanie! Niestety, problemem jest tak zwana dekoherencja: po czasach rzędu milisekund w niezwykle przecież delikatny układ wkrada się jakieś zaburzenie. Powoduje to utratę informacji. Nie wiadomo, czy kiedykolwiek uporamy się praktycznie z tym problemem. Wraz ze wzrostem komputera, problem będzie się nasilał. Już jednak zbudowanie układu kilku qubitów i precyzyjna nimi manipulacja pokazuje, jak niesamowity rozwój technik eksperymentalnych nastąpił w ostatnich latach.

Z pewnością komputer kwantowy nie będzie kolejną zabawką stojącą na biurku, lecz urządzeniem służącym do wykonywania najbardziej zaawansowanych obliczeń, umieszczonym w najlepszych światowych laboratoriach. Trzeba jednak pamiętać, że taką samą przyszłość wieszczono klasycznym komputerom...

[1] Peter Shor, *Proceedings 35th Annual Symposium on Foundations of Computer Science*, IEEE Comput. Soc. Press (1994), 124-134.

[2] Lov K. Grover, *Proceedings of the 28th Annual ACM Symposium on the Theory of Computing*, (1996), 212-219.

[3] D. Kielpinski, C.R. Monroe and D.J. Wineland, *Nature* 417, 709-711 (2002).

[4] D. Leibfried et al., *Nature* 422, 412-415 (2003).

mała delta

Ujarzmianie złożoności

Andrzej WALAT

Logo jest środowiskiem, w którym można znakomicie uczyć się, jak radzić sobie ze złożonością. W tym artykule omówię trzy rozwiązania zadania o wadze szalkowej z finału Mazowieckiego Konkursu Informatycznego 2000 (dla uczniów dawnej szkoły podstawowej) i na tym przykładzie przedstawię techniki ujarzmiania złożoności.

Andrzej Walat, Ośrodek Edukacji Informatycznej i Zastosowań Komputerów w Warszawie

Zadanie o wadze szalkowej

Jeśli mamy po jednym odważniku o masie 1 kg, 3 kg, 9 kg itd. aż do 3^n kg, to możemy na wadze szalkowej odważyć każdy ciężar, którego masa w kg jest liczbą całkowitą z zakresu od 0 do $1+3+\ldots+3^n$ i to w dodatku tylko w jeden sposób. Na przykład, żeby zrównoważyć ciężar 11 kg, trzeba na jednej szali położyć dany ciężar i odważnik 1kg, a na drugiej szali – odważniki 9 kg i 3 kg.

Zdefiniuj funkcję `ZRW :mc`, która dla danej masy ciężaru, będącej liczbą całkowitą dodatnią, wyznacza odważniki, które trzeba położyć na szalach wagi, żeby zrównoważyć dany ciężar. Wynikiem funkcji ma być dwuelementowa lista. Pierwszym jej elementem powinna być lista mas odważników, które należy położyć na jednej szali z danym ciężarem, a drugim elementem – lista mas odważników, które należy położyć na przeciwnej szali. Obie te listy powinny być uporządkowane malejąco.

Przykłady

Poprawnym wynikiem wyrażenia:	jest:
`ZRW 101`	`[[9 1] [81 27 3]]`
`ZRW 282`	`[[] [243 27 9 3]]`

Rozwiązanie 1

Używając 3 odważników o masie 1, 3 oraz 9 kg, można zrównoważyć na wadze szalkowej każdy ciężar, którego masa `mc` w kg jest liczbą całkowitą z zakresu od 0 do 13 włącznie. Na przykład:

- ciężar 10 kg można zrównoważyć, kładąc na przeciwnej szali odważniki [9 1],
- ciężar 11 kg można zrównoważyć, kładąc na tej samej szali odważnik 1, a na przeciwnej [9 3].

Największy ciężar, jaki można zrównoważyć za pomocą czterech odważników 1, 3, 9, 27, to $1+3+9+27=40$. Ale czy każdy ciężar o masie `mc` od 0 do 40 da się odważyć, jeśli mamy tylko te 4 odważniki?

- Każdy od 0 do 13 kg tak, bo wtedy wystarczą 3 odważniki: 1, 3 oraz 9.
- Każdy od 14 do 27 kg tak, bo po położeniu na przeciwnej szali odważnika 27 kg trzeba będzie zrównoważyć $27-\texttt{mc}\leqslant 13$; a to można osiągnąć, używając odważników 1, 3 oraz 9.
- Każdy od 28 do 40 kg tak, po położeniu na przeciwnej szali odważnika 27 kg i wtedy trzeba będzie zrównoważyć $\texttt{mc}-27\leqslant 13$ kg; a to można osiągnąć, używając odważników 1, 3 oraz 9.

Podobnie można wykazać, że za pomocą 5 odważników 1, 3, 9, 27, 81 można zrównoważyć na wadze szalkowej każdy ciężar, którego masa `mc` w kg jest liczbą całkowitą nie większą niż 121, itd.

Problem zrównoważenia ciężaru o danej masie jest prosty, jeśli ta masa jest nieduża, np. nie większa niż 13 albo jeszcze lepiej 4. Jednak jeśli masa ciężaru jest bardzo duża, problem staje się złożony. Sposobem na złożoność jest **redukcja złożoności**. Problem zrównoważenia ciężaru o dużej masie można zredukować do problemu zrównoważenia ciężaru o odpowiednio mniejszej masie. Trzeba w tym celu znaleźć najcięższy (maksymalny) odważnik `mo`, jakiego musimy użyć, by zrównoważyć dany ciężar (to nie jest takie trudne) i położyć go na przeciwnej szali.

- Jeśli $\texttt{mc}=\texttt{mo}$, to koniec, ciężar jest zrównoważony.
- Jeśli $\texttt{mc}>\texttt{mo}$, to trzeba jeszcze zrównoważyć mniejszą masę $\texttt{mc}-\texttt{mo}$.

- Jeśli mo > mc, to trzeba zrównoważyć mniejszą masę mo − mc, nadmiar na przeciwnej szali.

Ten pomysł redukcji złożonego przypadku do prostszego został wykorzystany w rozwiązaniu, na które składają się 4 procedury przedstawione na poniższym wydruku.

```
OTO ZRW :mc
  WYNIK BALANS :mc MODW :mc
JUŻ

OTO BALANS :mc :mo
  JEŚLI :mc = :mo [WYNIK LISTA [] ( LISTA :mo )]
  JEŚLI :mc > :mo [WYNIK DOPW :mo ZRW :mc - :mo]
  WYNIK DOPW :mo WSPAK ZRW :mo - :mc
JUŻ
```

```
OTO MODW :mc
  NIECH "PTG3 1
  NIECH "SPTG3 1
  DOPÓKI [:mc > :SPTG3][PRZYP "PTG3 3 * :PTG3
  PRZYP "SPTG3 :SPTG3 + :PTG3]
  WYNIK :PTG3
JUŻ

OTO DOPW :EL :PARA
  WYNIK LISTA PIERW :PARA NAP :EL OST :PARA
JUŻ
```

Pozioma kreska oddziela dwie podstawowe procedury określające ogólny plan działań oraz dwie pomocnicze procedury – środki realizacji planu.

Procedura BALANS określa sposób redukcji zagadnienia zrównoważenia ciężaru do prostszego zagadnienia zrównoważenia mniejszego ciężaru. Musi ona mieć dwie dane: mc – masę ciężaru oraz mo – masę maksymalnego odważnika, jakiego trzeba użyć, by zrównoważyć dany ciężar.

Funkcja – ZRW znajduje wynik, wywołując BALANS z dwiema danymi: mc – masa ciężaru oraz MODW :mc – masa maksymalnego odważnika, jakiego trzeba użyć, by zrównoważyć dany ciężar.

Trzeba jeszcze wyjaśnić, w jaki sposób funkcja MODW wyznacza swoją wartość – masę maksymalnego odważnika, jakiego trzeba użyć, by zrównoważyć dany ciężar. Wyjaśnię to na przykładzie. Załóżmy, że chcemy wyznaczyć masę maksymalnego odważnika, jakiego trzeba użyć, by zrównoważyć ciężar o masie 101 kg. W tym celu będziemy obliczali kolejne potęgi liczby 3 (wartości PTG3) oraz kolejne sumy potęg liczby 3 (wartości SPTG3) tak długo, dopóki dana masa 101 będzie większa niż wartość SPTG3.

PTG3	SPTG3
1	1
3	4
9	13
27	40
81	121

Końcowa wartość PTG3 – w tym przypadku 81 – to szukany maksymalny odważnik. W taki sam sposób funkcja MODW znajduje wynik w dowolnym przypadku. Trzeba dodać, że funkcję MODW, jak prawie wszystko, można zdefiniować na wiele sposobów. Można np. wykorzystać spostrzeżenie, że kolejna wartość SPTG3 – to zawsze potrojona poprzednia wartość plus 1.

```
OTO MODW :mc
  NIECH "SPTG3 1
  DOPÓKI [:mc > :SPTG3][ PRZYP "SPTG3 3 * :SPTG3
  + 1]
  WYNIK :SPTG3 - (:SPTG3 - 1 ) / 3
JUŻ
```

Funkcję MODW możemy zdefiniować jak tylko nam się spodoba (byle poprawnie), nie ma to żadnego wpływu na nasz ogólny plan – treść funkcji ZRW oraz BALANS. Tak samo, jak zmiana elementu złożonej konstrukcji, np. okien, nie wpływa na ogólny plan budowli (pod warunkiem, że elementy zachowują ustalone parametry, np. rozmiary). Taka separowalność części rozwiązania zadania jest cechą dobrego programowania.

Ostatnia z czterech funkcji składających się na rozwiązanie DOPW musi mieć dwie dane: jakiś nowy element EL oraz listę dwóch list PARA. Jej wynikiem jest dana lista dwóch list z nowym elementem dopisanym na początek drugiej listy.

Rozwiązanie 2

Prawie każde zadanie można rozwiązać na 1001 różnych sposobów. Myśl o tym jest pocieszająca. Jeśli może być tak wiele różnych pomysłów rozwiązania zadania, dlaczego przynajmniej jeden z nich nie miałby wpaść do mojej głowy?

Skorzystamy z rady pochodzącej z książeczki George'a Polyi *Jak to rozwiązać*.

„*Jeśli nie możesz rozwiązać postawionego zadania, spróbuj najpierw rozwiązać jakieś zadanie pokrewne. Czy nie mógłbyś wymyślić bardziej dostępnego zadania pokrewnego?*"

Owszem, moglibyśmy. Moglibyśmy, na przykład, założyć, że dysponujemy nie po jednym tylko odważniku o masie 1, 3, 9, 27,... kg, ale mamy po dwa odważniki każdego rodzaju. Nietrudno zauważyć, że w takim przypadku każdy ciężar, którego masa mc w kg jest nieujemną liczbą całkowitą, można zrównoważyć, kładąc odważniki wyłącznie na przeciwną szalę niż ciężar. Żeby

je wyznaczyć, wystarczy znaleźć zapis liczby `mc` w systemie trójkowym.

Na przykład liczba $667 = 2 \cdot 243 + 2 \cdot 81 + 0 \cdot 27 + +2 \cdot 9 + 0 \cdot 3 + 1 \cdot 1$ ma zapis trójkowy 220201. Oznacza to, że ciężar o masie 667 kg można zrównoważyć, kładąc na przeciwnej szali: 2 odważniki 243 kg, 2 odważniki 81 kg, dwa odważniki 9 kg oraz 1 odważnik 1 kg. Ten przykład pokazuje, jak łatwo można rozwiązać wymyślone przez nas pokrewne zadanie. No dobrze, ale czy można to wykorzystać do rozwiązania naszego oryginalnego zadania? Tak, i to bardzo prosto. Popatrzmy na zapis zrównoważenia ciężaru 667 kg w postaci następującego rysunku-tabelki:

2	2	0	2	0	1
243	81	27	9	3	1
0	0	0	0	0	0

Górny wiersz tabeli wskazuje, jakie odważniki leżą na przeciwnej szali niż ciężar, a dolny – jakie razem z ciężarem (aktualnie żadne). Jest to dobre rozwiązanie pokrewnego zadania, ale nie zadania oryginalnego, ponieważ nie dysponujemy dwoma odważnikami o masie 9 kg ani 81 kg. Możemy za to kłaść odważniki także na szalce razem z ciężarem. Zamiast 2 odważników 9 kg, możemy położyć na przeciwnej szalce niż ciężar 1 odważnik 27 kg oraz 1 odważnik 9 kg na szalce razem z ciężarem. W ten sposób otrzymamy następującą sytuację równowagi:

2	2	1	0	0	1
243	81	27	9	3	1
0	0	0	1	0	0

Teraz trzeba coś zrobić z dwoma odważnikami 81 kg na szalce przeciwnej niż ciężar. Podobnie jak poprzednio, zdejmiemy je, a za to dołożymy na tej samej szali 1 odważnik o masie 243 kg oraz razem z ciężarem 1 odważnik o masie 81 kg:

3	0	1	0	0	1
243	81	27	9	3	1
0	1	0	1	0	0

W ten sposób milcząco zakładamy, że pożyczyliśmy skądś trzeci odważnik 243 kg, ale tylko na chwilę, zaraz go oddamy. Zdejmiemy z szalki trzy odważniki 243 kg i zamiast tego położymy 1 odważnik 729 kg:

1	0	0	1	0	0	1
729	243	81	27	9	3	1
0	0	1	0	1	0	0

Teraz mamy sytuację równowagi i każdy odważnik od 1 do 729 został użyty co najwyżej 1 raz. Na jednej szalce mamy ciężar 667 kg i odważniki o łącznej masie 81 kg + 9 kg = 90 kg, razem 757 kg. Na drugiej szali: 729 kg + 27 kg + 1 kg.

Ten sposób postępowania, zilustrowany przykładem, nietrudno zapisać w postaci odpowiednich procedur w Logo znajdujących szukane zrównoważenie dowolnego ciężaru. Można go także na wiele sposobów udoskonalić. Tego jednak tutaj nie będziemy robili, pozostawiając to zadanie zainteresowanym Czytelnikom. Czasem warto przeanalizować różne możliwe idee rozwiązania problemu nie po to, żeby je do końca wykorzystać, zrealizować w postaci procedur w jakimś języku programowania, ale po to, żeby lepiej, wszechstronniej zrozumieć problem i być może dzięki temu odkryć inne pomysły.

Rozwiązanie 3

W tym przypadku także skorzystamy z rady George'a Polyi.

„Jeśli nie możesz rozwiązać postawionego zadania, to ... czy nie mógłbyś wymyślić jakiegoś bardziej ogólnego zadania?"

Zdefiniujemy funkcję `BALANS`, która musi mieć dwie dane:
- dowolną potęgę `p` liczby 3 oraz
- masę ciężaru `mc` daną jako wielokrotność `p`,

a jej wynikiem jest zrównoważenie ciężaru określone w postaci listy dwóch list, tak jak tego wymagamy od funkcji `ZRW`.

Funkcja `BALANS` jest uogólnieniem `ZRW`, bo wynik `ZRW` dla dowolnej całkowitej masy ciała `mc` można otrzymać jako wynik `BALANS` z ustaloną wartością drugiego parametru 1:

```
OTO ZRW :mc
  WYNIK BALANS :mc 1
JUŻ
```

Załóżmy, że mamy daną masę ciężaru będącą wielokrotnością `mc` potęgi `p` liczby 3.

Jak w takim przypadku znaleźć odpowiednie zrównoważenie (balans)? Oczywiście w szczególnie prostym przypadku, gdy `mc` = 0, balans osiągamy, nie kładąc na obu szalach nic; wynikiem powinna być lista dwóch pustych list: `[[][]]`.

W bardziej złożonym przypadku, gdy `mc` nie jest zerem, możemy obliczyć resztę z dzielenia `mc` przez 3 i zredukować odpowiednio problem, zależnie od otrzymanej reszty.

Jeśli reszta = 0, to zadanie redukuje się do znalezienia wyniku:
`BALANS ILORAZC :MC 3 3 * :P.`

Jeśli reszta = 1, to trzeba znaleźć
`BALANS ILORAZC :MC 3 3 * :P` – jest to lista dwóch list – i do prawej z tych dwóch list dopisać na koniec `:p`.

Gdy reszta = 2, to trzeba położyć odważnik o masie `p` razem z ciężarem i znaleźć zrównoważenie odpowiednio większego o p ciężaru. Trzeba znaleźć `BALANS ( ILORAZC :MC 3 ) + 1 3 * :P` – jest to lista dwóch list i do lewej z tych dwóch list dopisać na koniec `:p`.

Pełnym rozwiązaniem zadania jest zestaw czterech procedur:

```
OTO ZRW :mc
  WYNIK BALANS :mc 1
JUŻ

OTO BALANS :mc :P
  JEŚLI :mc = 0 [WYNIK LISTA [] []]
  WYNIK WYBIERZ RESZTA :mc 3 [0 [BALANS ILORAZC
                        :mc 3 3 * :P]
                        1 [DOPW :P BALANS ILORAZC
                        :mc 3 3 * :P]
                        2 [DOLW :P BALANS ( ILORAZC
                        :mc 3 ) + 1 3 * :P]]
JUŻ
```

```
OTO DOPW :el :para
  WYNIK LISTA PIERW :para NAK :el OST :para
JUŻ

OTO DOLW :el :para
  WYNIK LISTA NAK :el PIERW :para OST :para
JUŻ
```

Podobnie, jak w rozwiązaniu 1, pierwsze dwie wytyczają ogólny plan postępowania, a dwie procedury pod kreską to pomocnicze środki realizacji.

W tym rozwiązaniu funkcja `DOPW` musi mieć dwie dane: jakiś nowy element `EL` oraz listę dwóch list `PARA`. Jej wynikiem jest dana para list z nowym elementem dopisanym na koniec drugiej listy.

Funkcja `DOLW` dopisuje nowy element na koniec pierwszej listy w danej parze list.

O znaczeniu grubej kreski – abstrakcja danych

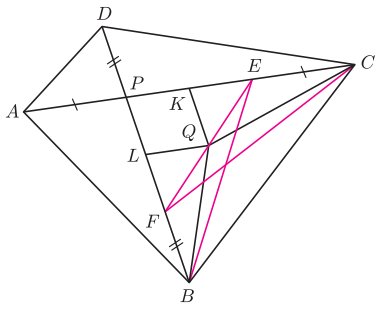
O funkcjach `DOPW` oraz `DOLW` warto chwilę pomyśleć w oderwaniu od aktualnego zadania. Funkcje dopisujące nowy element do lewego lub prawego elementu danej pary list mogą być użyteczne w różnych sytuacjach. Mogłyby istnieć już w Logo jako funkcje pierwotne obok innych funkcji, takich jak `NAP` oraz `NAK`. Jest wiele dialektów Logo, więc może istnieje taki, w którym `DOPW` oraz `DOLW` są funkcjami pierwotnymi i nie trzeba ich definiować. Wyobraźmy sobie, że istnieje. W takim Logo rozwiązanie naszego zadania byłoby jeszcze prostsze i składałoby się tylko z dwóch krótkich definicji funkcji `ZRW` oraz `BALANS`. Za tym „wyobraźmy sobie" kryje się ważna technika radzenia sobie ze złożonością. Kiedy mam do rozwiązania złożony problem i wiem, że będę potrzebował pewnych narzędzi, których nie ma w konkretnym Logo, jakim dysponuję, mogę sobie wyobrazić, że je mam. Tworzę wtedy swoje rozwiązanie w **wirtualnym Logo**, w którym mam wszystkie potrzebne narzędzia. To bardzo upraszcza rozwiązanie problemu. Następnie, żeby móc uruchomić moje rozwiązanie, poprawiam Logo, dodając nowe środki „procedury pod kreską". Ich zdefiniowanie jest już osobnym i zwykle prostym zadaniem. Technika, która polega na tym, że najpierw używam środków, którymi faktycznie nie dysponuję, jak bym je miał, i nie martwię się, jak je zdefiniuję, a następnie osobno definiuję potrzebne środki – nie zaprzątając sobie głowy, do czego są mi aktualnie potrzebne – nazywa się **abstrakcją danych**. Ważne cechy abstrakcji danych to:

1. Separowalność używania i definiowania. Używam czegoś, nie martwiąc się, jak to zdefiniuję. Definiuję, nie zaprzątając sobie głowy, do czego aktualnie chcę tego używać.
2. Tworzenie produktów wielorakiego użytku (*reusage*). Analizując różne problemy, wyodrębniamy środki (funkcje, procedury, typy struktur danych), które mogą mieć zastosowanie w różnorodnych sytuacjach; zwykle nadajemy im jakąś sensowną nazwę i próbujemy zdefiniować jako abstrakcyjne obiekty do wielorakiego użytku – w oderwaniu od aktualnej potrzeby.

Złożoność ujarzmiona

Każda epoka ma swoje wyzwania. Wielkim wyzwaniem na miarę XXI wieku jest ujarzmianie złożoności. Opanowanie złożoności wymaga czasem cierpliwego wielokrotnego podchodzenia do problemu z różnych stron, jak do dzikiego mustanga. Ale jej okiełznanie może dać równie wielką satysfakcję. Problem, który mógł się wydawać złożony i wymykał się spod kontroli, daje się nagle zamknąć w kilku prostych definicjach. To jest właśnie złożoność ujarzmiona.

Rozwiązanie zadania M 1105.
Przez punkt Q poprowadźmy prostą równoległą do prostej KL i przecinającą przekątne AC i BD odpowiednio w punktach E i F.



Wówczas punkty K i L są odpowiednio środkami odcinków PE i PF, skąd wniosek, że $AP = CE$ oraz $DP = BF$.

Odległość punktu Q od prostej BC jest średnią arytmetyczną odległości punktów E i F od prostej BC. Stąd uzyskujemy $[BCQ] = \frac{1}{2}([BCE] + [BCF]) = \frac{1}{2}([ABP] + [CDP])$.

Komety, trygonometria i komputer

Grzegorz SITARSKI

Kiedy chodziłem do gimnazjum, na lekcjach matematyki uczyliśmy się dowodzić tzw. tożsamości trygonometrycznych, np.

$$\cos 2\alpha + \sin 2\alpha + 1 = 2\cos\alpha(\cos\alpha + \sin\alpha).$$

Wychodząc z lewej strony powyższej równości, należało za pomocą przekształceń algebraicznych doprowadzić do postaci z prawej strony (albo odwrotnie). Trzeba było, oczywiście, znać kilka tożsamości podstawowych, w tym przypadku:

$$\sin^2\alpha + \cos^2\alpha = 1,$$

$$\sin 2\alpha = 2\sin\alpha\cos\alpha,$$

$$\cos 2\alpha = \cos^2\alpha - \sin^2\alpha.$$

Mnóstwo takich ćwiczeń mieliśmy zadawane do domu, a wiązało się to głównie z doprowadzaniem wyrażeń matematycznych do tzw. postaci logarytmicznej, o czym teraz mówić nie będziemy.

Nie wiem, czy dziś też uczą w szkole takich rzeczy, bo przecież wystarczy wziąć kalkulator z funkcjami trygonometrycznymi i pokazać, że dla dowolnego kąta lewa strona tożsamości jest równa prawej (albo nie jest). Sądziłem dawniej, że był to tylko rodzaj gimnastyki umysłowej w praktyce do niczego nieprzydatnej. Okazuje się jednak, że w badaniach naukowych dotyczących ruchu komet zetknąłem się z problemem dowodzenia tożsamości trygonometrycznych i moja praktyka w tej dziedzinie, nabyta w szkole, bardzo mi się w czasach przedkomputerowych przydała.

Ruch komety opisywany jest na orbicie okołosłonecznej, a obserwacje wykonywane są na Ziemi. Obliczenia ruchu komety prowadzone są zatem w innym układzie odniesienia niż obserwacje i dla porównania wyników obliczeń z obserwacjami trzeba dokonać przejścia z jednego układu do drugiego. W układzie orbitalnym podstawową płaszczyzną (x, y) jest płaszczyzna heliocentrycznej orbity komety, a podstawowym kierunkiem (osi x) jest kierunek na peryhelium orbity widziany ze Słońca. W układzie wykonywania obserwacji podstawową płaszczyzną jest płaszczyzna równika ziemskiego, a oś x jest skierowana do punktu równonocy wiosennej, tj. wzdłuż prostej przecięcia się płaszczyzny równika z płaszczyzną ekliptyki. Aby przeliczyć współrzędne jakiegoś wektora z układu orbitalnego do równikowego, trzeba pomnożyć je przez tzw. kosinusy kierunkowe, czyli kosinusy kątów między osiami obydwu układów. W tym celu dokonujemy czterech obrotów pierwszego układu na przemian wokół osi z i x o kąty ω, i, Ω oraz ε. Nie wnikając w znaczenie tych kątów, otrzymujemy ostatecznie wyrażenia trygonometryczne na kosinusy kierunkowe. Dla przykładu, kosinusy kierunkowe C_1, C_2, C_3 między osią x nowego układu a osiami x, y, z starego układu mają postać:

$$\begin{aligned} C_1 &= \cos\omega\cos\Omega - \sin\omega\sin\Omega\cos i, \\ C_2 &= \cos\varepsilon(\sin\omega\cos\Omega\cos i + \cos\omega\sin\Omega) - \sin\varepsilon\sin\omega\sin i, \\ C_3 &= \sin\varepsilon(\sin\omega\cos\Omega\cos i + \cos\omega\sin\Omega) + \cos\varepsilon\sin\omega\sin i. \end{aligned} \tag{1}$$

Wyrażenia (1) są dość skomplikowane, a nie mam żadnej pewności, czy przy ich wyprowadzaniu nie popełniłem błędu (bądź też nie przepisałem ich błędnie z podręcznika). Istnieje jednak możliwość sprawdzenia poprawności tych wyrażeń, bowiem wiemy z trygonometrii, że dla kosinusów kierunkowych musi zachodzić równość $C_1^2 + C_2^2 + C_3^2 = 1$. Nie jest to już tak prosta tożsamość, jak zadawane do domu w szkole, ale nie było rady, trzeba było podnieść do kwadratu prawe strony wyrażeń (1) i tak długo je przekształcać, aż wynik doprowadzi się do jedności. Jeśli się to nie udawało, trzeba było szukać błędu albo we wzorach (1), albo w procesie sprawdzania.

Dziś już nie trzeba się tak męczyć. Wystarczy napisać niewielki programik komputerowy i sprawdzić, czy dla dowolnych wartości kątów $\varepsilon, \omega, \Omega, i$ zachodzi $C_1^2 + C_2^2 + C_3^2 = 1$.

Grzegorz Sitarski, Centrum Badań Kosmicznych PAN

W miarę rozwoju badań nad kometami znowu natknąłem się na podobny problem trygonometryczny. Tym razem chodziło o składowe wektora siły niegrawitacyjnej działającej na lodowe jądro komety. Wyobraźmy sobie, że jądro komety jest kulą z lodowo-pyłowej mieszaniny, która pod wpływem promieniowania słonecznego zaczyna sublimować, co powoduje ulatnianie się gazów i pyłu, tworzących głowę i warkocz komety. Materia opuszczająca powierzchnię jądra wywołuje efekt odrzutu i w rezultacie wystąpienie wektora siły niegrawitacyjnej prostopadłego do powierzchni jądra. Wartość tej siły jest największa w punkcie podsłonecznym, czyli w tym miejscu na powierzchni jądra, gdzie na kometarnym niebie Słońce świeci w zenicie. Weźmy teraz pod uwagę ruchomy układ orbitalny, taki że kierunkiem podstawowym w płaszczyźnie orbity będzie kierunek na Słońce. Jak każde ciało niebieskie, jądro komety wiruje, a płaszczyzna równika jądra przecina płaszczyznę orbity pod jakimś kątem. Kiedy kometa biegnie po orbicie, punkt podsłoneczny wędruje w ciągu kometarnego roku od zwrotnika do zwrotnika po powierzchni jądra. Niezależnie jednak od położenia komety na orbicie wektor maksimum siły niegrawitacyjnej powinien zawsze być skierowany od Słońca, czyli w układzie orbitalnym powinien mieć tylko jedną składową, składową radialną. I tak by było, gdyby nie tzw. bezwładność cieplna, która powoduje, że na skutek ruchu wirowego jądra kierunek maksymalnego wyrzutu materii odchyla się nieco od kierunku radialnego. Powstaje problem znalezienia składowych wektora siły niegrawitacyjnej w układzie orbitalnym, czyli kosinusów kierunkowych między wektorem a osiami układu. Jeżeli η jest tzw. kątem opóźnienia, mierzonym od kierunku na Słońce wzdłuż równika jądra komety, I kątem między płaszczyznami równika i orbity, a λ kątem określającym położenie komety na orbicie (mierzonym od linii przecięcia się płaszczyzn równika i orbity), to kosinusy kierunkowe wektora siły niegrawitacyjnej będą równe:

$$
\begin{aligned}
C_1 &= \cos\eta + (1-\cos\eta)\sin^2 I \sin^2\lambda,\\
C_2 &= \sin\eta\cos I + (1-\cos\eta)\sin^2 I \sin\lambda\cos\lambda\\
C_3 &= [\sin\eta\cos\lambda - (1-\cos\eta)\cos I\sin\lambda]\sin I.
\end{aligned}
\tag{2}
$$

Chociaż mamy tu do czynienia tylko z trzema kątami, wyprowadzenie wzorów (2) wcale nie jest proste (pamiętam, że rysowałem sobie odpowiednie łuki na powierzchni jabłka!). A jak teraz sprawdzić poprawność tych wyrażeń? Oczywiście dla kosinusów kierunkowych musi zachodzić tożsamość $C_1^2 + C_2^2 + C_3^2 = 1$, ale sprawdzać w ten sposób wyrażenia (2) za pomocą przekształceń algebraicznych?! Nawet nie próbowałem, bo co zrobić potem z $\sin^4 I$ i $\sin^4\lambda$, które się pojawią? Sprawdzenie zrobił za mnie komputer i potwierdził poprawność wyprowadzonych wzorów.

Ale na tym nie koniec. Problem ogromnie się komplikuje, jeśli jądro komety jest spłaszczone i opiszemy jego kształt elipsoidą obrotową, której spłaszczenie oznaczymy przez $s = 1 - b/a$, gdzie b i a to odpowiednio biegunowy i równikowy promień jądra. Wektor siły niegrawitacyjnej jest nadal prostopadły do powierzchni jądra, ale wprowadziwszy pomocniczą wielkość $S = s(2-s)$, otrzymamy teraz następującą postać kosinusów kierunkowych:

$$
\begin{aligned}
C_1 &= [\cos\eta + (1-S-\cos\eta)\sin^2 I\sin^2\lambda]/Z,\\
C_2 &= [\sin\eta\cos I + (1-S-\cos\eta)\sin^2 I\sin\lambda\cos\lambda]/Z,\\
C_3 &= [\sin\eta\cos\lambda - (1-S-\cos\lambda)\cos I\sin\lambda]\sin I/Z,
\end{aligned}
\tag{3}
$$

gdzie $Z = \sqrt{1 - S(2-S)\sin^2 I\sin^2\lambda}$.

Nie zazdroszczę komuś, kto chciałby wykazać na drodze przekształceń algebraicznych, że dla wyrażeń (3) suma $C_1^2 + C_2^2 + C_3^2$ jest równa 1. Radzę napisać prosty program i komputer pokaże, że dla zupełnie dowolnych wartości η, I, λ i S (czy też s), nawet jeśli nie mają one znaczenia przyrodniczego, zawsze otrzymamy $C_1^2 + C_2^2 + C_3^2 = 1$.

Samoistne powstawanie korków

Robert ŻAK

Ile razy zastanawialiście się nad przyczyną korka na trzypasmowej autostradzie, co mogło być jego przyczyną? Zwłaszcza że sytuacja na drodze często poprawiała się bez żadnego śladu czegokolwiek, co mogłoby spowodować zatrzymanie ruchu. Czy aby powstał korek, potrzebne są światła, utrudnienia ruchu związane z robotami na drodze, czy też może kolizja? Na to pytanie postaramy się odpowiedzieć w poniższym tekście.

Historia badań nad ruchem samochodów sięga lat pięćdziesiątych XX wieku, kiedy to proponowano modele zarówno kinematyczne, jak i oparte na dynamice jednowymiarowego przepływu lepkiej, lecz nieściśliwej cieczy. Modele takie miały swoje zalety, ale nie były zbyt efektywne. Np. drugie podejście nie tylko wymagało rozwiązania równania Naviera–Stokesa (które, jak wiadomo, jest ściśle rozwiązywalne tylko w pewnych szczególnych przypadkach), ale także wprowadzało wiele wolnych parametrów. Przełom nastąpił na początku lat 90. za sprawą wzrastającej mocy obliczeniowej komputerów. Zaproponowano wtedy tzw. modele automatów komórkowych (ang. *cellular automata*). Modele te był następnie udoskonalane i rozszerzane, przybierając coraz bardziej zaawansowaną postać.

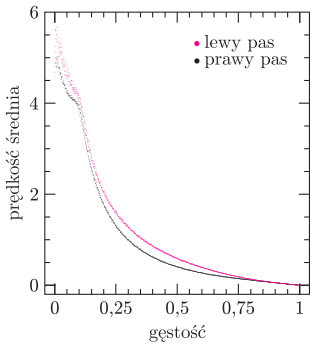
Taki właśnie dyskretny model zaproponujemy do naszych rozważań. Jego dyskusję rozpoczniemy od omówienia przypadku szczególnie prostego, mianowicie modelu jednopasmowej drogi, na której ruch odbywa się bez żadnych przeszkód. Następnie rozszerzymy nasze rozważania na drogę dwupasmową, aby zarysować ideę wprowadzania dodatkowych pasów ruchu. W końcu porównamy otrzymane rezultaty w ramach modelu trzypasmowej autostrady z wynikami rzeczywistych pomiarów.

Zakładamy, że droga podzielona jest na komórki równej długości, przy czym samochody poruszają się w pętli. Każda komórka może znajdować się w $v_{\max}+2$ stanach, tzn. może się w niej znajdować samochód o dyskretnej szybkości od zera do szybkości maksymalnej $v_{\max}$ albo komórka może być pusta. Zaznaczmy jednak, że $v_{\max}$ oznacza maksymalną szybkość dopuszczalną na drodze, natomiast szybkość maksymalna indywidualnych samochodów może być mniejsza. Zakładamy, iż samochody poruszają się zgodnie z obowiązującymi przepisami. Na długość komórki składa się długość samochodu oraz bezpieczny dystans przed i za samochodem (pomijamy różnice w długości samochodów). Przyjmujemy także, iż na drodze nie występują przeszkody w postaci uszkodzonych samochodów, skrzyżowań, świateł itp. Samochody nie mogą również zderzać się.

Zajmijmy się ruchem pojedynczego samochodu w modelu jednopasmowym. Samochód w każdym kroku przesuwa się o tyle komórek do przodu, ile wynosi jego prędkość. Jeżeli liczba pustych komórek do znajdującego się przed nim samochodu jest mniejsza od jego szybkości, samochód zwalnia do szybkości równej liczbie komórek dzielących oba samochody. Kierowca może jednak zahamować zbyt silnie i zmniejszyć szybkość jeszcze bardziej. Takie zachowanie kierowcy opisujemy w naszym modelu za pomocą prawdopodobieństwa i możemy traktować je jako pewną miarę jego umiejętności. Samochód przyspiesza, jeżeli ma wystarczająco dużo miejsca przed sobą i jego prędkość jest mniejsza od jego prędkości maksymalnej. Manewr przyspieszania opisujemy tym samym rozkładem prawdopodobieństwa, co w przypadku zwalniania.

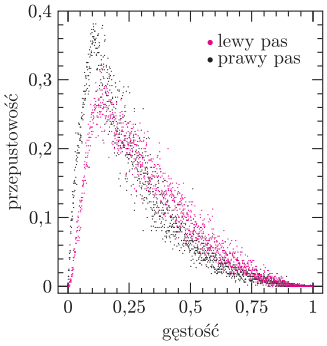
Rozszerzenie na więcej pasów jest dość naturalne. W naszej symulacji zaproponowaliśmy następujące rozwiązanie. Samochód zmienia pas na lewy, jeżeli znajduje się przed nim samochód poruszający się wolniej, a liczba wolnych komórek przed i za samochodem na pasie docelowym jest co najmniej równa maksymalnej prędkości dopuszczonej w symulacji. W przeciwnym razie pojazd jest zmuszony zmniejszyć prędkość. Samochody powracają na prawy pas zawsze, gdy tylko mają taką możliwość.

Przedyskutujemy teraz wyniki naszych symulacji w przypadku dwupasmowym. Na rysunku 1 widzimy zależność średniej szybkości samochodów od gęstości na drodze.



Rys. 1. Prędkość średnia, mierzona liczbą komórek pokonywanych w jednym kroku, odpowiadająca gęstości, mierzonej liczbą samochodów przypadających na komórkę.

Widzimy, co jest dość oczywiste, że ruch na lewym pasie odbywa się zawsze nieco szybciej niż na prawym. Ponadto wzrost liczby samochodów powoduje spadek średniej szybkości ruchu. Ciekawa jest też zależność natężenia ruchu (liczby samochodów przejeżdżających przez komórkę) od gęstości (rys. 2).

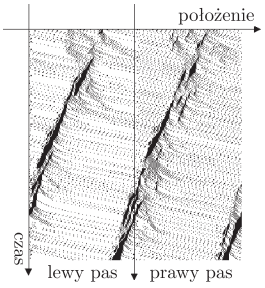


Rys. 2. Natężenie ruchu, mierzone stosunkiem liczby samochodów przejeżdżających przez daną komórkę do liczby kroków, odpowiadające gęstości, mierzonej liczbą samochodów przypadających na komórkę.

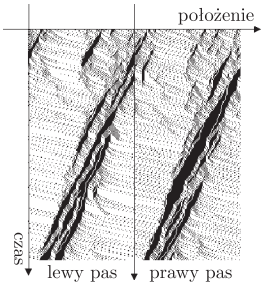
Robert Żak, student, Wydział Fizyki, Uniwersytet Warszawski

Na początku natężenie wzrasta w przybliżeniu liniowo ze wzrostem gęstości (coraz więcej samochodów mogących się poruszać z prędkościami bliskimi maksymalnej), a następnie zaczyna maleć (coraz więcej samochodów uwięzionych w korkach). Zauważmy jeszcze, że charakter obu wykresów zmienia się dla tej samej wartości gęstości, około 0,06 samochodu na komórkę. Na obu wykresach pokazano wyniki wielu symulacji (różne rozmieszczenie i szybkości początkowe samochodów na drodze przy starcie programu).

Zajmiemy się teraz szczegółową analizą tego, co dzieje się na drodze. Na rysunkach 3 i 4 przedstawione są diagramy czasoprzestrzenne dla gęstości samochodów 0,15 i 0,25 samochodu na komórkę.



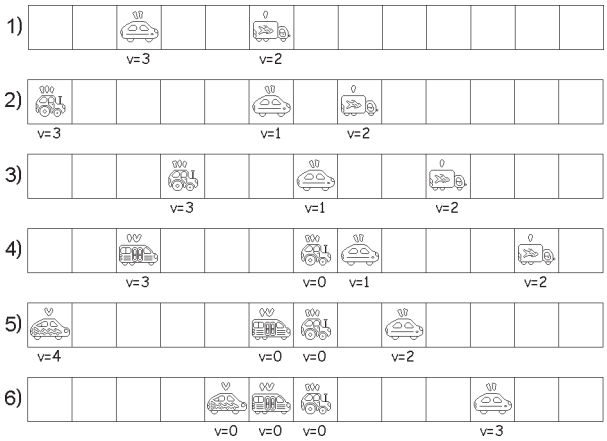
Rys. 3



Rys. 4

Każda kropka obrazuje samochód. Czarne obszary odpowiadają samochodom stojącym jeden za drugim w korku. Widzimy, że kiedy samochody poruszają się w prawo, korek porusza się w lewo. Widzimy także, że sytuacja na obu pasach jest dość symetryczna, tzn. powstaniu korka na prawym pasie towarzyszy pojawienie się zatoru na pasie lewym.

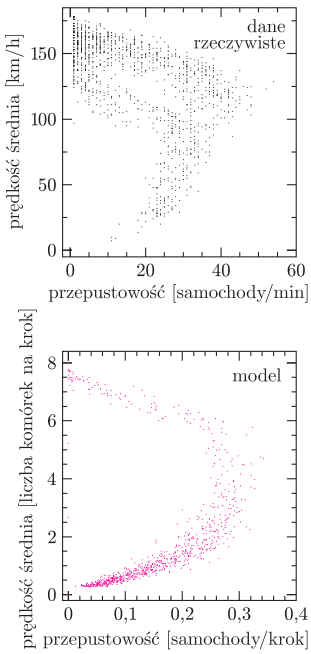
Spróbujemy wyjaśnić mechanizm powstawania korka. W tym celu powrócimy do modelu jednopasmowego, aby uprościć nieco rozważania. Przyjrzyjmy się rysunkowi 5.



Rys. 5. Mechanizm powstawania korka. Opis w tekście.

Podzieliliśmy go na 6 części i teraz kolejno je omówimy. Samochód II ($v = 3$) porusza się z szybkością większą od liczby wolnych komórek. W myśl naszych zasad musi zwolnić. Przypuśćmy jednak, że kierowca hamuje zbyt silnie i zwalnia o dwie jednostki szybkości. W następnym kroku samochód II nie może przyspieszyć, bo nie ma wystarczająco dużo komórek przed nim. W kroku trzecim samochód III gwałtownie hamuje i w wyniku hamowania zatrzymuje się w kroku czwartym. Pojazd IV zmuszony jest także się zatrzymać. W tym samym czasie samochód II przyspiesza. Następnie samochód V także staje, podczas gdy samochód II dalej zwiększa swoją prędkość. Możemy sobie wyobrazić, że kolejne samochody będą zatrzymywać się na końcu korka, z kolei samochody znajdujące się na początku będą przyspieszać i odłączać się od korka w miarę możliwości. Widzimy, że korek nie tylko może powstać samoistnie, ale także rozumiemy przyczynę ruchu zagęszczenia w kierunku przeciwnym do kierunku ruchu samochodów.

Warto na koniec odnieść się do rzeczywistości. Ponieważ wykonanie badań wymaga odpowiedniej aparatury i dostępu do autostrad, których nie mamy zbyt wiele, poprosiliśmy o pomoc Petera Wagnera z German Aerospace Centre i Landschaftsverband Nordrhein-Westfalen w Niemczech, któremu w tym miejscu pragniemy podziękować. Otrzymane wyniki pomiarów opracowaliśmy i dostosowaliśmy do skonfrontowania z naszymi symulacjami. Porównamy zależność średniej szybkości od natężenia ruchu dla trzypasmowej autostrady. Przyjmujemy, że komórka ma długość 6 m, a jeden krok odpowiada jednej sekundzie, czyli np. pokonywanie jednej komórki na krok odpowiada szybkości 22 km/h. Na rysunku 6 przedstawiono oba wykresy.



Rys. 6. Związek między natężeniem ruchu a średnią prędkością dla rzeczywistych danych (u góry) i rozpatrywanego modelu (u dołu).

Prezentujemy wykresy dla lewego pasa (dla środkowego i prawego wykresy są podobne, tyle że ruch odbywa się wolniej). Widzimy pewne podobieństwo między wykresami, chociaż w przypadku rzeczywistym rozrzut prędkości jest dużo większy. Górna część wykresów odpowiada sytuacji, kiedy nie ma korków, dolna załamaniu się ruchu i powstaniu na drodze zagęszczeń.

Choć nasz model jest bardzo uproszczony, to pozwala dostrzec i opisać pewne cechy ruchu pojazdów na autostradzie. Wyjaśnia także spontaniczne pojawianie się korków, co było zasadniczym jego celem. Można go jednak rozwijać, wprowadzając np. szerszą skalę szybkości. Wymaga to jednak zwiększenia liczby komórek składających się na autostradę, tak aby pozostała ona wystarczająco długa, a to z kolei prowadzi do konieczności użycia komputerów o większej mocy obliczeniowej. Mamy nadzieję, że tym artykułem przyczynimy się nie tyle do zmniejszenia ilości korków na drogach, ile do pogłębienia wiedzy na temat ich natury.

Klub 44

Redaguje Jerzy B. BROJAN

Czołówka ligi zadaniowej
Klub 44 F
po uwzględnieniu ocen rozwiązań zadań
392 ($WT = 2{,}13$) i **393** ($WT = 3{,}33$)
z numeru 2/2005

Jerzy Witkowski	– Radlin	33,16
Marian Łupieżowiec	– Gliwice	26,54
Konrad Kapcia	– Częstochowa	20,83
Mateusz Łącki	– Kraków	17,70
Tomasz Tkocz	– Rybnik	10,58

Rozwiązania zadań z fizyki z numeru 4/2005

Przypominamy treść zadań:

396. Cylinder, do którego stałym strumieniem wlewa się woda, pełni funkcję zegara wodnego – na bocznej ściance naklejono pionowo skalę minutową. Czy jakieś inne przezroczyste naczynie o kształcie bryły obrotowej z pionową osią symetrii, po naklejeniu na nie tej samej skali, może pełnić rolę zegara? Skala powinna być umieszczona w jednej płaszczyźnie z osią symetrii. Grubość ścianki należy pominąć. Jeśli jest to możliwe, to narysować przekrój naczynia, na podstawie odpowiednich obliczeń numerycznych.

397. Rozważmy dwuwymiarowy model „atomu" składającego się z n „elektronów" odpychających się wzajemnie siłą Coulomba ($F = 1/r^2$), na które dodatkowo działa przyciągająca siła centralna ze strony nieruchomego „jądra"; siła ta powinna umożliwiać utrzymanie „elektronów" w stanie równowagi (np. niech ta siła będzie postaci $F = r^\beta$, gdzie $\beta > 0$). Zadanie polegało na numerycznym zbadaniu tych stanów równowagi, w zależności od liczby n (równej kilka-kilkanaście, maksymalnie około 50) oraz parametrów opisujących siłę centralną (w powyższym przykładzie β).

Rozwiązanie zadania **397** zamieściliśmy na naszej stronie internetowej.

396. Objętość bardzo cienkiej warstwy wody w naczyniu jest równa

$$\mathrm{d}V = \pi r^2 \mathrm{d}h,$$

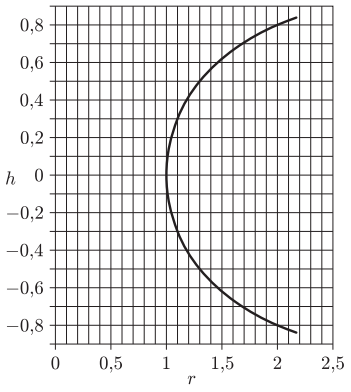
gdzie r jest promieniem przekroju poziomego naczynia na wysokości h. Niech $\mathrm{d}l$ będzie odcinkiem podziałki odpowiadającym wysokości $\mathrm{d}h$ – w takim razie $\mathrm{d}h = \mathrm{d}l \cos\alpha$ (gdzie α – kąt odchylenia tego odcinka od pionu), a ponieważ zgodnie z treścią zadania $\mathrm{d}V/\mathrm{d}l = \mathrm{const}$, więc kształt naczynia jest określony przez warunek

$$r^2 \cos\alpha = \mathrm{const} = c.$$

Ponieważ $\mathrm{d}r/\mathrm{d}h = \operatorname{tg}\alpha$, więc po prostych przekształceniach znajdujemy postać funkcji $h(r)$

$$h(r) = \int^r \frac{\mathrm{d}r'}{\sqrt{r'^4/c^2 - 1}}.$$

Stałą c możemy położyć równą 1, gdyż odpowiada to przeskalowaniu funkcji $h(r)$ (jednakowej zmianie skali obu wielkości h i r). Otrzymaliśmy tzw. całkę eliptyczną. Wykres otrzymany na podstawie obliczeń numerycznych (dla dolnej granicy równej 1) jest podany na rysunku.



Rozwiązanie zadania F 649.
W układzie środka masy minimalna energia do zajścia reakcji to $E_1 = 2m_e c^2 + Mc^2$ (wszystkie cząstki po zajściu reakcji spoczywają). Niezmiennik czteropędu

$$\mu^2 c^2 = \frac{E_1^2}{c^2} - \vec{p}_1^{\,2}$$

wynosi więc

$$\mu^2 c^2 = \frac{E_1^2}{c^2} = M^2 c^2 + 4m_e^2 c^2 + 4Mm_e c^2,$$

ale z drugiej strony

$$\mu^2 c^2 = \frac{\left(Mc^2 + E_f\right)}{c^2} - \frac{E_f^2}{c^2} = M^2 c^2 + 2E_f M,$$

gdzie E_f to energia fotonu, a $\frac{E_f}{c}$ to jego pęd. Wobec tego

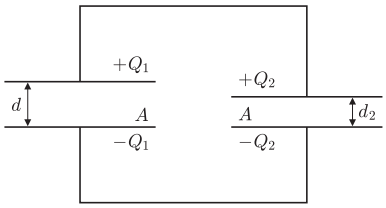
$$E_f = 2m_e c^2 + 2\frac{m_e}{M} m_e c^2.$$

Jest to właśnie minimalna energia fotonu dopuszczająca zajście procesu.

Rozwiązanie zadania M 1106.
Wszystkich niepustych podzbiorów danego zbioru dziesięcioelementowego jest $2^{10} - 1 = 1023$. Każdy z tych podzbiorów zawiera co najwyżej 10 liczb, z których żadna nie przekracza 100. A zatem suma liczb w każdym podzbiorze nie przekracza $10 \cdot 100 = 1000$. Stąd, na mocy zasady szufladkowej Dirichleta, uzyskujemy tezę.

Rozwiązanie zadania F 650.
Zasada zachowania ładunku ma postać $Q_1 + Q_2 = 2Q$, a równość napięć na kondensatorach

$$\frac{Q_1 d}{A\epsilon_0} = \frac{Q_2 d_2}{A\epsilon_0},$$



zatem po rozwiązaniu

$$Q_1 = \frac{2d_2}{d + d_2} Q, \qquad Q_2 = \frac{2d}{d + d_2} Q.$$

Energia całkowita układu to

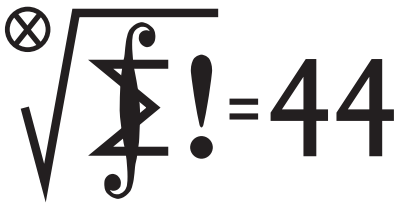
$$E = \frac{Q_1^2 d}{2A\epsilon_0} + \frac{Q_2^2 d_2}{2A\epsilon_0} = \frac{Q^2 d_2}{2A\epsilon_0}\left(\frac{4}{1 + d_2/d}\right).$$

Zauważmy, że energia zgromadzona na pojedynczym kondensatorze naładowanym do ładunku Q to

$$E_1 = \frac{Q^2 d_2}{2A\,\epsilon_0} < E.$$

Rozwinąwszy E w szereg Taylora wokół $d = d_2$ i pozostawiwszy tylko wyrazy liniowe w d_2/d, widzimy, że siła przyciągania okładek jest cztery razy większa niż w przypadku pojedynczego kondensatora.

Klub 44



Redaguje Marcin E. KUCZMA

Rozwiązania zadań z matematyki z numeru 4/2005

Przypominamy treść zadań:

499. Dla ustalonej liczby naturalnej n rozważamy ciągi $(a_0, a_1, \ldots, a_n)$ spełniające warunki $a_0 = 0$ oraz $|a_k| = |1 + a_{k-1}|$ dla $k = 1, \ldots, n$. Wyznaczyć najmniejszą możliwą wartość sumy $a_1 + \ldots + a_n$.

500. Znaleźć taką liczbę naturalną n, że w każdym zbiorze n punktów kratowych na płaszczyźnie istnieje pięć punktów, których środek ciężkości jest punktem kratowym. Im mniejsza liczba n, tym lepszy wynik i wyższa ocena.

499. Oznaczmy sumę $a_1 + \ldots + a_n$ przez s_n. Zależność między a_k oraz a_{k-1} można zapisać tak:

$$a_k^2 = 1 + 2a_{k-1} + a_{k-1}^2.$$

Dodając stronami te równości dla $k = 1, \ldots, n$ oraz redukując składniki, które się pojawią po obu stronach, dochodzimy do zależności

$$a_n^2 = n + 2s_{n-1}, \qquad \text{czyli} \qquad (a_n + 1)^2 = n + 1 + 2s_n.$$

Stąd

$$s_n = \frac{(a_n + 1)^2 - (n + 1)}{2} \geqslant -\frac{n+1}{2}.$$

Zatem wartość sumy s_n (będącej liczbą całkowitą) nie jest mniejsza od $-\lceil \frac{1}{2}n \rceil$. Jest to szukane minimum, osiągane dla ciągów $(a_0, a_1, \ldots, a_n)$ o wyrazach $a_k = \frac{1}{2}((-1)^k - 1)$.

500. Każdemu punktowi kratowemu przyporządkujmy parę reszt z dzielenia jego obu współrzędnych przez 5. Jest 25 możliwych par. Mając 101 punktów kratowych, można spośród nich wybrać 5 punktów wyznaczających tę samą parę reszt. Suma pierwszych współrzędnych tej piątki punktów oraz suma ich drugich współrzędnych są liczbami podzielnymi przez 5, czyli ich środek ciężkości jest punktem kratowym. Zatem liczba $n = 101$ ma żądaną własność.

Aby znaleźć mniejszą „dobrą" liczbę n, skorzystamy ze znanego twierdzenia (trójki autorów Erdős–Ginzburg–Ziv): *W każdym* $(2n-1)$*-elementowym zbiorze liczb całkowitych istnieje* n *liczb, których suma dzieli się przez* n.

Będzie nam ono potrzebne tylko w wersji dla $n = 5$; redukuje się ono wówczas do kilkuset przypadków, które oczywiście da się przebadać bezpośrednio. Ale można też uzyskać jego tezę dla $n = 5$ w sposób następujący:

Jeżeli α, β, γ, δ są liczbami niepodzielnymi przez 5, to wykreślając z sumy $\alpha + \beta + \gamma + \delta$ pewne składniki (być może wszystkie, być może żadnego), jesteśmy w stanie uzyskać każdą z wartości 0, 1, 2, 3, 4 jako resztę z dzielenia sumy pozostałych liczb przez 5; nietrudne uzasadnienie tej uwagi pozostawiamy jako ćwiczenie.

Mamy wykazać, że wśród dziewięciu liczb całkowitych istnieje pięć o sumie podzielnej przez 5. Rozważamy reszty z dzielenia tych liczb przez 5: $0 \leqslant a_1 \leqslant \ldots \leqslant a_9 \leqslant 4$. Jeżeli istnieje pięć jednakowych wartości a_i, teza jest oczywista. W pozostałym przypadku następujące różnice są dodatnie: $a_6 - a_2 = \alpha$, $a_7 - a_3 = \beta$, $a_8 - a_4 = \gamma$, $a_9 - a_5 = \delta$. W myśl poprzedniej uwagi, możemy usunąć ze sumy $\alpha + \beta + \gamma + \delta$ pewne składniki tak, by suma pozostałych przystawała (mod 5) do $-(a_1 + a_2 + a_3 + a_4 + a_5)$. Dodajemy te pozostałe składniki do sumy $a_1 + a_2 + a_3 + a_4 + a_5$ i wykonując oczywiste redukcje, otrzymujemy pięć liczb, których suma dzieli się przez 5.

Łatwo teraz wywnioskować, że rozważaną w zadaniu własność ma liczba $n = 41$: spośród danych 41 punktów kratowych wybieramy dziewięć o jednakowej (mod 5) pierwszej współrzędnej; wśród nich – na mocy wykazanego twierdzenia – znajduje się pięć punktów, których drugie współrzędne dają w sumie liczbę podzielną przez 5; ich środek ciężkości jest punktem kratowym.

Potrafimy jeszcze nieco zmniejszyć n metodami obliczeniowymi (na przykład $n = 33$ jest „dobre"). Niewymyślne algorytmy prowadzą jednak do ogromnej liczby przypadków. Jesteśmy przekonani, że zgrabne algorytmy i znacząco mniejsze „dobre" wartości n poznamy od uczestników konkursu. Innych Czytelników także zachęcamy do obliczeniowego zaatakowania problemu.

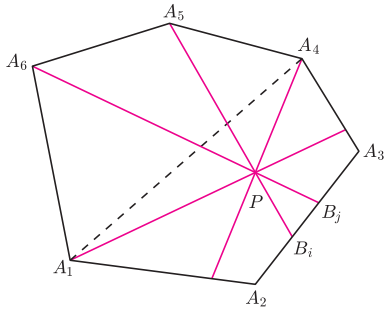
Jedno jest pewne: nie można zejść poniżej $n = 17$; bowiem wśród 16 punktów, których każda współrzędna jest równa 0, 1, 5 lub 6, nie ma pięciu punktów, których środek ciężkości byłby punktem kratowym.

Czołówka ligi zadaniowej
Klub 44 M
po uwzględnieniu ocen rozwiązań zadań **491** ($WT = 1{,}14$) i **492** ($WT = 3{,}59$) z numeru 12/2004

Józef Siwy	– Łaziska Górne	44,68
Zbigniew Sewartowski	– Wieliczka	42,66
Bartłomiej Dyda	– Wrocław	41,53
Tomasz Rawlik	– Braunschweig	39,35
Zbigniew Galias	– Kraków	38,09
Piotr Kumor	– Olsztyn	37,78
Tomasz Warszawski	– Kraków	36,24
Marcin Kasperski	– Warszawa	34,88

Pan Józef Siwy uczestniczy z długimi przerwami, jego nazwisko jest więc rzadko widoczne w ogłaszanej czołówce. Ale oto zamknął drugą rundę (ciekawostka: pierwszą ukończył przeszło 20 lat temu!). Gratulujemy! i rozpoczynamy czekanie na trzecią.

Rozwiązanie zadania M 1107.
Rozpatrzmy przekątną A_1A_{n+1} i przyjmijmy, bez straty ogólności, że punkt P leży wewnątrz wielokąta $A_1A_2\ldots A_{n+1}$ (na rysunku $n=3$).



Wówczas każda spośród $n+1$ prostych $A_{n+1}P, A_{n+2}P, \ldots, A_{2n}P, A_1P$ przecina jeden spośród n boków $A_1A_2, A_2A_3, \ldots, A_nA_{n+1}$. Stąd wynika, na mocy zasady szufladkowej Dirichleta, że któryś z tych boków zawiera co najmniej dwa punkty B_i, B_j. Zatem któryś z pozostałych n boków danego $(2n)$-kąta nie zawiera żadnego z punktów $B_1, B_2, \ldots, B_{2n}$.

Patrz w niebo

Zapadający się obłok materii międzygwiazdowej, gdy stanie się dostatecznie nieprzezroczysty, zaczyna się ogrzewać. Jest to skutek przemiany jego energii grawitacyjnej w energię kinetyczną tworzących go cząstek. Warunek, że obłok musi być nieprzezroczysty, jest tu istotny, gdyż we wcześniejszej fazie zapadania gęstniejący obłok nie ogrzewa się, ponieważ powstające przy tym promieniowanie (na razie długofalowe) może go natychmiast opuścić. W każdym razie dalszy kolaps obłoku powoduje dalsze jego ogrzewanie się, zwłaszcza w centrum, gdzie – jeżeli temperatura osiągnie wartość około 10 mln K – atomy wodoru zaczynają wchodzić w reakcje termojądrowe. Obiekt taki staje się „urzędowo” gwiazdą.

Jak widać, jeżeli zapadałby się obłok o zbyt małej masie, to gwiazdą nigdy by nie został, gdyż nie byłby w stanie uruchomić reakcji termojądrowych. Mógłby osiągnąć stadium protogwiazdy i nawet przez pewien czas świeciłby jak gwiazda, ale tylko kosztem energii grawitacyjnej. Całkiem sensowne jest więc pytanie, jaką najmniejszą masę może mieć gwiazda. Teoria budowy gwiazd podaje tu wartość 0,08 masy Słońca. Otóż niedawno wykryto w gwiazdozbiorze Pompy w odległości około 4 pc jedną z najsłabszych znanych gwiazd, której masę oceniono właśnie na 0,08. Została wykryta w programie Deep Near Infrared Survey (DENIS), ma katalogową nazwę DENIS 1048-39, temperaturę powierzchniową 2200 K i widomą jasność 16 mag. Jej typ widmowy to M9, a na tym właśnie symbolu kończy się klasyfikacja widmowa gwiazd ciągu głównego. Z racji małej odległości gwiazda bardzo szybko porusza się po niebie ($1''{,}5$ na rok), aczkolwiek pod tym względem jeszcze jej daleko do Gwiazdy Barnarda (ponad $10''$ na rok). Szczerze mówiąc, należałoby się upewnić, że obiekt DENIS 1048-39 świeci rzeczywiście dzięki reakcjom termojądrowym, ale do tego potrzebne są dalsze badania.

Tomasz KWAST

Sierpień

Wieczorem w sierpniu najokazalej prezentuje się obszar Drogi Mlecznej, w którym znajduje się centrum naszej Galaktyki, zwłaszcza że dzięki wakacjom mamy na ogół możliwość oglądania nieba z miejsc o czystym powietrzu. Nisko na południu widać więc Strzelca i Skorpiona, a nad nimi Tarczę (dawniej Tarczę Sobieskiego), gwiazdozbiór niezwiązany z mitologią, a otoczony przez same „mitologiczne”. Gwiazdozbiór ten, mały i nie zawierający jasnych gwiazd, wprowadził na niebo Heweliusz na cześć króla Jana Sobieskiego. Przez lornetkę widać tam wielkie chmury gwiazdowe tworzące Drogę Mleczną, a na ich tle piękną gromadę otwartą M 11 (NGC 6705). Gromada ma jasność 6,3 mag, średnicę kątową $12'$ i zawiera 400 gwiazd. Jest to gromada o dość silnej koncentracji gwiazd ku środkowi (jak na gromadę otwartą), gdyż w jej centrum gęstość gwiazd jest 10 000 razy większa od gęstości gwiazd w pobliżu Słońca. Leży w odległości 1700 pc.

Merkury znajdzie się 23 VIII najdalej kątowo od Słońca, można go więc próbować odnaleźć na niebie przed wschodem. Wenus i Jowisz są w Pannie i planety te widać po zachodzie Słońca na zachodnim niebie. Mars jest w Baranie i widać go w drugiej połowie nocy. W Raku, a więc niedaleko, jest też Saturn, który wschodzi poźniej od Marsa, widać go więc krócej. Nów Księżyca wypada 5 VIII, a pełnia 19 VIII. Księżyc zakryje Wenus 8 VIII, będzie to jednak widoczne tylko w Kanadzie i na Alasce. Następnie 10 VIII zakryje Jowisza, ale to z kolei będzie można zaobserwować z Oceanu Indyjskiego. Wreszcie 14 VIII Księżyc zakryje Antaresa, co zobaczą mieszkańcy Półwyspu Arabskiego, południowej Azji i Indonezji. My zobaczymy tylko zbliżenia Księżyca do tych obiektów. W pierwszej połowie sierpnia spodziewamy się większej liczby meteorów z roju Perseidów. Maksimum roju przypada na 12 VIII, a oczekiwane natężenie – niemal jeden błysk na minutę. Nie zdziwmy się jednak, jeżeli wypadnie to zupełnie inaczej.

T. K.

Ewolucja geometryczna, czyli jak powstało oko

W jaki sposób mogło wyewoluować oko? Przypadki różnych systemów biologicznych, których funkcjonowanie jest możliwe tylko wtedy, gdy wszystkie ich składowe idealnie współgrają, wydają się niemożliwe do powstania drogą stopniowych ulepszeń. Bo jaki może być pożytek z posiadania 5% oka?

Zacznijmy od najprostszego narządu wzroku, jakim jest pojedyncza światłoczuła komórka, umieszczona gdzieś w ciele organizmu. Czy taki narząd może się na coś przydać? Naturalnie, że tak: już samo rozróżnienie, czy jest jasno, czy ciemno, pozwala wodnym skorupiakom ukrywać się przed wrogami. W nocy skorupiaki żerują przy powierzchni wody zjadając plankton, w dzień natomiast opadają w głębiny, gdzie nie mogą ich dostrzec ryby. Jeśli ciało właściciela takiego jednokomórkowego oka jest nieprzezroczyste, pozwala także wykryć kierunek, z którego płynie światło. Organizm może się obracać, notując zmiany w oświetleniu swojego oka i ustawiać w pożądanym kierunku. Robią tak, na przykład, czerwie, by odpełznąć z oświetlonej przestrzeni.

Zwiększenie liczby reagujących na światło komórek umożliwia dostrzeżenie słabszego światła, ale też stwarza nowe możliwości. Tym razem można już wykryć kierunek, z którego nadchodzi światło, bez konieczności poruszania ciałem, gdyż światłoczułe komórki, umieszczone w różnych punktach krzywizny ciała, są osłonięte przed światłem padającym z różnych kierunków. Skupienie grupy światłoczułych komórek w tarczkę może początkowo po prostu zwiększyć skuteczność wykrywania światła. Jednak już niewielka zmiana krzywizny tej tarczki daje nowe możliwości. Co ciekawe, bez względu na to, czy tarczka zacznie się robić wypukła, czy wklęsła, powstanie zwarty organ zdolny do wykrywania kierunku, z którego pada światło. Taki jest początek ewolucji dwóch głównych typów oczu: uwypuklanie się światłoczułej płaszczyzny dało początek powstaniu oczu złożonych, jakie mają np. owady, wgłębianie się – oczu, jakie mamy choćby my sami.

Oczy różnych typów ewoluowały niezależnie przynajmniej 40 razy, tworząc wiele różnych struktur pełniących tę samą funkcję. Dalej zajmiemy się tylko jednym z nich – modelem „wklęsłej tarczki".

Pozwala ona na rozpoznanie kierunku, z którego pada światło, lecz nadal nie umożliwia rozpoznawania kształtów oświetlonych obiektów. Dzieje się tak dlatego, że każdy punkt widzianego obrazu odbija promienie we wszystkich kierunkach i promienie te padają na każdą z komórek wklęsłej tarczki. Jednak im bardziej wklęsła będzie tarczka, tym mniej wpadnie do niej przypadkowych promieni, aż wreszcie, w skrajnym przypadku, gdy promienie będą wpadały tylko przez mały otworek, powstaje *camera obscura*, która tworzy całkiem wyraźny obraz na dnie oka. Jednak tylko do pewnego stopnia – im mniejszy otwór, tym obraz wyraźniejszy, lecz także ciemniejszy. Oczy tego typu ma wiele organizmów, na przykład glony, łodziki, niektóre ślimaki czy małże.

Dalsze usprawnienie powstającego mechanizmu polega na jego ochronie mechanicznej. U łodzika cenne światłoczułe komórki pozostają wystawione na bezpośrednie działanie środowiska. Osłony muszą być oczywiście przezroczyste. W istocie oczy szczelinowe małży czy ślimaków są w mniejszym lub większym stopniu wypełnione przezroczystą substancją. Jeśli dojdzie do całkowitego wypełnienia gałki ocznej taką substancją, otwiera się możliwość powstania kolejnego usprawnienia. Problem ostrości obrazu można rozwiązać za pomocą soczewki, czyli np. bryłki przezroczystej substancji o tak zaokrąglonych brzegach, iż wszystkie promienie pochodzące z danego punktu przestrzeni skupiają się w jednym punkcie obrazu.

Używane w aparatach fotograficznych soczewki mają bardzo precyzyjnie dobrany kształt, aby zdjęcie było możliwie ostre i pozbawione innych wad. Jednak już zastosowanie tak prymitywnej soczewki, jaką jest torebka foliowa wypełniona wodą, pozwala na uzyskanie całkiem czytelnego zdjęcia.

Formowanie się soczewki ocznej zaczęło się zapewne od powstania zgęstnienia substancji wypełniającej wnętrze oka, które w kolejnych etapach przyjmowało coraz doskonalszy kształt, pozwalający widzieć coraz wyraźniej.

Dodatkową wskazówką trafności przedstawionego rozumowania okazała się opublikowana w 1989 roku symulacja komputerowa ewolucji oka. Uwzględniała ona prawa optyki, a rezultat – niemal dokładne powtórzenie opisanego wyżej scenariusza – przedstawia rysunek.

Ewolucja oka – wybrane etapy opisanych w tekście ciągłych zmian (na podstawie: Richard Dawkins „Wspinaczka na szczyt nieprawdopodobieństwa").

Paweł PORĘBA

Współpraca: Anna LORENC, Jarek BRYK

Korespondencję do Kącika Biologicznego prosimy przysyłać do Jarka Bryka pod adresem: bryko@poczta.onet.pl