

Cached Fibonacci

Descriere

Una dintre cele mai simple probleme pentru recursivitate este calcularea termenului N din șirul lui Fibonacci ($f_1 = 1, f_2 = 1$). Andrei dorește să facă o implementare recursivă pentru acest algoritm, dar observă că pentru valori mari ale lui N , algoritmul nu se termină rapid.

El este fan al programării funcționale, așa că nu dorește să schimbe modul recursiv de a calcula termenii din șir și alege o soluție ingenerească pentru a rezolva problema timpului. Astfel, pentru fiecare termen anterior deja calculat, îl va reține într-un cache pentru a nu mai fi calculat în viitor.

Din păcate, el nu este familiar cu sintaxa de C++ modern, lucrând doar în limbaje pur funcționale până acum. Rezolvați bug-urile din codul lui Andrei, fără a schimba modul recursiv de calculare a termenului fibonacci.

Download sursă [aici](#).

Formatul datelor de intrare

Pe prima linie un număr de query-uri T . Pe următoarele T linii, câte un număr natural p pentru care se dorește calcularea recursivă a termenului p din șirul lui Fibonacci.

Formatul datelor de ieșire

Răspunsul pentru fiecare query.

Note

Cu excepția celor deja prezente în schelet, nu aveți voie să folosiți următoarele șiruri de caractere în codul vostru: `return`, `exit`, `goto`, `while`, `for`, `do`, `fork`, `//`, `/*`, `/**`, `#define`.

Exemple

Exemplul 1

Intrare

```
3
2
3
4
```

Ieșire

```
1
2
3
```

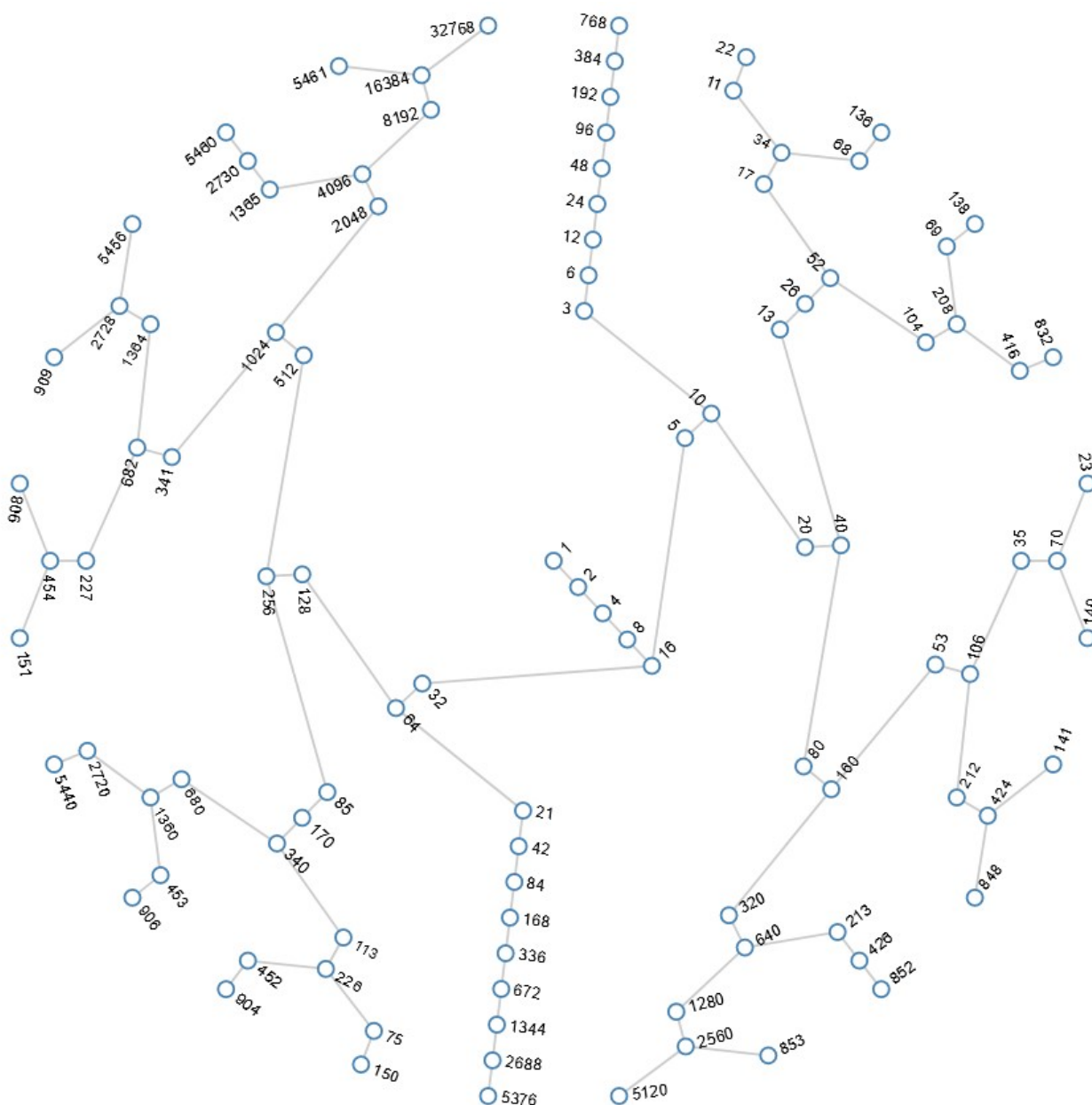

Collatz Halves

Descriere

Un șir Collatz este definit astfel: se alege un număr de început, $x = a_0$. Dacă a_n este par, atunci $a_{n+1} = \frac{a_n}{2}$, altfel $a_{n+1} = 3a_n + 1$. Exemplu de șir Collatz:

30, 15, 46, 23, 70, 35, 106, 53, 160, 80, 40, 20, 10, 5, 16, 8, 4, 2, 1, 4, 2, 1, ...

Fie un arbore generat din parcurgerea șirurilor Collatz. Rădăcina este 1. Dacă două numere sunt legate, înseamnă că există un șir Collatz în care acele două numere sunt consecutive. Arborele este aranjat astfel încât, aplicând un pas de recurență, ne apropiem de rădăcină. Ceea ce este interesant e că arborele este simetric, într-o măsură, și arată astfel:



1. Aplicăm operația de recurență a șirului până ajungem la o poartă.
2. Reținem operațiile făcute și ordinea lor (înjumătățire sau triplare).
3. Setăm numărul curent ca fiind cel de la poarta din cealaltă jumătate (5 sau 32).
4. Pornind de la numărul curent, aplicăm operațiile inverse, de la ultima la prima (dublare și împărțire la 3).
5. Numărul final este geamănul.
6. Dacă oricare din operațiile de la pasul 4 **sunt imposibile, ne oprim**. Numărul inițial este propriul lui geamăn.
7. Dacă, aplicând operația de recurență, nu se ajunge la o poartă, numărul este propriul lui geamăn.

Operațiile inverse sunt: $a_n = 2a_{n+1}$ și $a_n = (a_{n+1} - 1)/3$. Prima operație se poate aplica pentru orice număr. În schimb, a doua se poate aplica doar pentru numere de forma $3k + 1$. Dacă suntem nevoiți să aplicăm a doua operație pentru numere de forma $3k$ sau $3k + 2$, spunem că **operația este imposibilă**. Din acel punct, arborele și-a pierdut simetria.

Dacă un număr este la mijloc, acesta este propriul lui geamăn. Altfel, geamănul este numărul care se obține prin aplicarea acelorași tipuri de operații care duc la numărul inițial, dar aplicate în cealaltă jumătate a arborelui.

Cerință

Se citesc n numere naturale de la tastatură. Pentru fiecare, să se afișeze pe ecran geamănul Collatz. Aveți la dispoziție o implementare care are câteva probleme.

Download sursă [aici](#).

Examples

Input

```
5
69 416 113 80 5
```

Output

```
453 2720 17 512 32
```

Explicație

Pornind de la numărul dat, ne ducem spre rădăcină, până la un număr căruia îi știm geamănul. Numărul 69 e prezent în desenul de mai sus, în jumătatea din dreapta a arborelui. Dacă parcurgem arborele spre rădăcină, până la 5, obținem șirul:

69, 208, 104, 52, 26, 13, 40, 20, 10, 5

Operațiile care se fac sunt:

$3x + 1, x/2, x/2, x/2, x/2, 3x + 1, x/2, x/2, x/2$

Geamănul lui 5 e 32. Pornind din 32, aplicăm operațiile inverse, de la dreapta la stânga:

$2x, 2x, 2x, (x - 1)/3, 2x, 2x, 2x, 2x, (x - 1)/3$

Aplicând operațiile, obținem șirul:

32, 64, 128, 256, 85, 170, 340, 680, 1360, 453

Așadar, geamănul lui 69 este 453. Același algoritm se aplică și pentru celelalte numere.

Input

```
3
1 4 96
```

Output

1 4 672

Explicație

Numerele 1 și 4 sunt la mijlocul arborelui, așadar sunt propriul lor geamăn.

Input

1
7

Output

7

Explicație

În încercarea de a afla numărul geamăn, una din operațiile inverse a fost imposibilă. Numărul 7 este propriul lui geamăn.

Formatul datelor de intare

Pe prima linie se găsește numărul n . Pe următoarea linie se găsesc n numere pentru care trebuie să se afle geamănul Collatz.

Formatul datelor de ieșire

Se vor scrie pe ecran n numere naturale separate prin spațiu, reprezentând gemenii Collatz ai celor n numere.

Note

- $n \leq 100$
- Numerele din șir sunt între 1 și 100

Gauss is coming

Descriere

Vlad este elev de liceu în clasa a 10-a. Pasionat atât de informatică, cât și de matematică, el decide să realizeze un program care să rezolve una dintre problemele lui preferate: rezolvarea unui sistem de ecuații liniare, oricât de mare. Astfel, ajunge să citească despre metoda lui Gauss de rezolvare a sistemelor de ecuații, care funcționează astfel: un sistem de forma

$$\begin{aligned}a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n &= b_1 \\a_{21}x_1 + a_{22}x_2 + \dots + a_{2n}x_n &= b_2 \\&\vdots \\a_{n1}x_1 + a_{n2}x_2 + \dots + a_{nn}x_n &= b_n\end{aligned}$$

Este adus la un sistem superior triunghiular numai prin scalări și adunări/scăderi de ecuații, astfel:

$$\begin{aligned}a'_{11}x_1 + a'_{12}x_2 + \dots + a'_{1n}x_n &= b'_1 \\0 + a'_{22}x_2 + \dots + a'_{2n}x_n &= b'_2 \\&\vdots \\0 + 0 + \dots + a'_{nn}x_n &= b'_n\end{aligned}$$

Acest sistem este foarte simplu de rezolvat, deoarece necunoscutele pot fi determinate independent, începând cu ultima (x_n) și terminând cu prima necunoscută (*substituție înapoi*).

Cu toate că este o metodă simplă, ușor de înțeles, Vlad își dă seama că uneori el poate avea probleme datorită împărțirilor la 0 ce pot apărea pe parcursul programului. De aceea, el aplică, în plus, următoarea strategie: atunci când va realiza eliminarea termenilor doriți, el va interschimba câte 2 ecuații, astfel:

- la primul pas, va interschimba ecuația 1 cu o ecuație care are coeficientul lui x_1 mai mare în modul.
- la cel de-al doilea pas, după ce s-a realizat eliminarea tuturor coeficienților lui x_1 (în afară de prima ecuație), se aduce în locul ecuației 2 pe cea care are coeficientul lui x_2 mai mare decât cel curent. La fiecare pas intermediar, se interschimbă ecuația cu o ecuație care se află sub ecuația curentă.
- în mod similar, procedeul continuă până când se ajunge la ultima ecuație.

Vlad a realizat implementarea algoritmului împreună cu strategia sa (numită **strategie de pivotare**), însă are probleme în testarea implementării. Mai exact, implementarea sa nu returnează rezultatul dorit. Ajutați-l să identifice problemele existente.

Download sursă [aici](#).

Examples

Fie următorul sistem de ecuații:

$$\begin{aligned}x_1 + x_2 &= 2 \\2x_1 + 3x_2 &= 5\end{aligned}$$

Astfel, coeficienții ecuațiilor și termenii liberi sunt memorați astfel:

$$A = \begin{pmatrix} 1 & 1 \\ 2 & 3 \end{pmatrix}$$

$$b = \begin{pmatrix} 2 \\ 5 \end{pmatrix}$$

Pentru primul pas, realizăm operația de pivotare: comparăm pivotul (*elementul diagonal*, 1 în acest caz) cu elementele de sub el. Observăm că $1 < 2$, de aceea sistemul devine:

$$A = \begin{pmatrix} 2 & 3 \\ 1 & 1 \end{pmatrix}$$

$$b = \begin{pmatrix} 5 \\ 2 \end{pmatrix}$$

Facem eliminarea efectivă (realizăm $L_2 = L_2 - 0.5L_1$; cu L_i am notat liniile din matricea sistemului):

$$A = \begin{pmatrix} 2 & 3 \\ 0 & -0.5 \end{pmatrix}$$

$$b = \begin{pmatrix} 5 \\ -0.5 \end{pmatrix}$$

Sistemul superior triunghiular este:

$$2x_1 + 3x_2 = 5$$

$$0x_1 - 0.5x_2 = -0.5$$

Acum rezolvăm sistemul prin substituție înapoi, astfel: determinăm x_2 din a doua ecuație și apoi îl introducem în prima ecuație.

$$x_2 = 1$$

$$2x_1 + 3 = 5$$

$$2x_1 = 2$$

$$x_1 = 1$$

Astfel, soluția sistemului este:

$$x = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

Note

- Se garantează că toate sistemele rezolvate au soluție unică.
- Ordinul sistemului $n \leq 100$.
- Nu aveți voie să modificați liniile marcate cu DO NOT MODIFY.

Square Prime

Descriere

Se citesc n numere naturale de la tastatură. Pentru fiecare, să se determine dacă poate fi scris sub forma $a^2 + p$, unde a este un număr natural nenul și p este un număr prim.

Liviu este inginer la Google, dar nu prea știe matematică, așa ca l-a rugat pe ChatGPT să scrie codul pentru problema aceasta. Totuși, ceva nu e în regulă...

Download sursă [aici](#).

Note

- $n \leq 1000$
- $1 \leq x_i \leq 1000$ (numerele din șir)

Exemple

Exemplul 1

Intrare

```
10
1 2 3 4 5 6 7 8 9 10
```

Ieșire

```
No
No
Yes
Yes
No
Yes
Yes
Yes
Yes
No
```