Liczbowy proces (rozwiązanie)

XIV OIJ, zawody I stopnia, tura ukryta
14 października 2019 – 13 stycznia 2020

Autor zadania: **Karol Pokorski**
Opracowanie: **Jakub Boguta, Jakub Wasilewski**

Olimpiada Informatyczna Juniorów

Zastanówmy się jak wygląda liczbowy proces zdefiniowany w zadaniu. Liczby, które uzyskamy za jego pomocą, będziemy nazywać *wyróżnionymi*. Z treści zadania widzimy, że pierwsze cztery wyróżnione liczby to $1, 2, 6$ i 42. Warto zauważyć, że każda kolejna liczba będzie większa od poprzedniej, czyli ciąg powstały w wyniku naszego procesu będzie ścisle rosnący.

Zastanówmy się, ile jest liczb wyróżnionych nie przekraczających $5 \cdot 10^9$ (czyli ograniczenia na wielkość zapytania w zadaniu). Aby to zrobić, napiszmy prosty program, który symuluje proces i zlicza wszystkie otrzymane liczby, aż do $5 \cdot 10^9$.

lic_symulacja.cpp

```cpp
#include "bits/stdc++.h"

using namespace std;

// Maksymalna liczba, o którą może pojawić się zapytanie.
const long long LIMIT = 5000000000LL;

// Funkcja f zdefiniowana w zadaniu - zwraca sumę cyfr liczby n.
int f(long long n) {
  // Deklarujemy zmienną na sumę.
  int suma = 0;
  // Dopóki liczba n nie jest równa zero,
  while (n != 0) {
    // dodaj do sumy ostatnią cyfrę (resztę z dzielenie przez 10)
    suma += n % 10;
    // i usuń tę cyfrę (podziel tę liczbę przez 10).
    n /= 10;
  }
  // Na końcu zwróć sumę.
  return suma;
}

// Funkcja g zdefniowana w zadaniu, zwraca n + f(n) * f(n).
long long g(long long n) {
  // Zapisujemy f(n) jako zmienną, aby nie liczyć tej funkcji dwukrotnie.
  int fn = f(n);
  // Zwracamy wartość funkcji g dla liczby n.
  return n + fn * fn;
}

int main() {
  // Deklarujemy licznik, który powie nam ile liczb jest w naszym ciągu.
  int licznik = 0;
  // Zaczynamy od n = 1.
  long long n = 1;
  // Dopóki n jest nie większe niż nasz limit:
  while (n <= LIMIT) {
    // zwiększ licznik (mamy kolejną liczbę, która jest w ciągu)
    licznik++;
    // i zamień n na g(n).
    n = g(n);
  }
  // Wypisz liczbę liczb w ciągu.
  cout << licznik << "\n";
}
```


Olimpiada finansowana jest ze środków Ministerstwa Edukacji Narodowej w ramach zadania publicznego „Organizacja i przeprowadzenie olimpiad i turniejów w latach szkolnych 2019/2020, 2020/2021, 2021/2022".

lic_symulacja.py

```python
# Maksymalna liczba, o którą może pojawić się zapytanie.
LIMIT = 5000000000

# Funkcja f zdefiniowana w zadaniu - zwraca sumę cyfr liczby n.
def f(n):
  # Deklarujemy zmienną na sumę.
  suma = 0
  # Dopóki liczba n nie jest równa zero,
  while n != 0:
    # dodaj do sumy ostatnią cyfrę (resztę z dzielenie przez 10)
    suma += n % 10
    # i usuń tę cyfrę (podziel tę liczbę przez 10).
    n //= 10

  # Na końcu zwróć sumę.
  return suma


# Funkcja g zdefniowana w zadaniu, zwraca n + f(n) * f(n).
def g(n):
  # Zapisujemy f(n) jako zmienną, aby nie liczyć tej funkcji dwukrotnie.
  fn = f(n)
  # Zwracamy wartość funkcji g dla liczby n.
  return n + fn * fn



# Deklarujemy licznik, który powie nam ile liczb jest w naszym ciągu.
licznik = 0
# Zaczynamy od n = 1.
n = 1
# Dopóki n jest nie większe niż nasz limit:
while n <= LIMIT:
  # zwiększ licznik (mamy kolejną liczbę, która jest w ciągu)
  licznik += 1
  # i zamień n na g(n).
  n = g(n)

# Wypisz liczbę liczb w ciągu.
print(licznik)
```

Po uruchomieniu takiego programu dowiadujemy się, że takich liczb jest tylko $2\,962\,717$, a więc wystarczająco mało, aby napisaną powyżej symulację zawrzeć w naszym rozwiązaniu, a przy tym zmieścić się w czasie. Nie możemy jednak sobie pozwolić na uruchamianie tego procesu dla każdego zapytania osobno.

Załóżmy, że możemy zapisać wszystkie liczby wyróżnione w tablicy/liście. Załóżmy także, że mamy wszystkie zapytania zapisane w drugiej tablicy/liście, dodatkowo posortowane.

liczby wyróżnione

1	2	6	42	78	303	⋯

1	2	2	30	42	731

zapytania

Teraz nasz algorytm (zwany **metodą dwóch wskaźników**) będzie wyglądał następująco. Ustawiamy dwa wskaźniki

Olimpiada finansowana jest ze środków Ministerstwa Edukacji Narodowej w ramach zadania publicznego „Organizacja i przeprowadzenie olimpiad i turniejów w latach szkolnych 2019/2020, 2020/2021, 2021/2022".

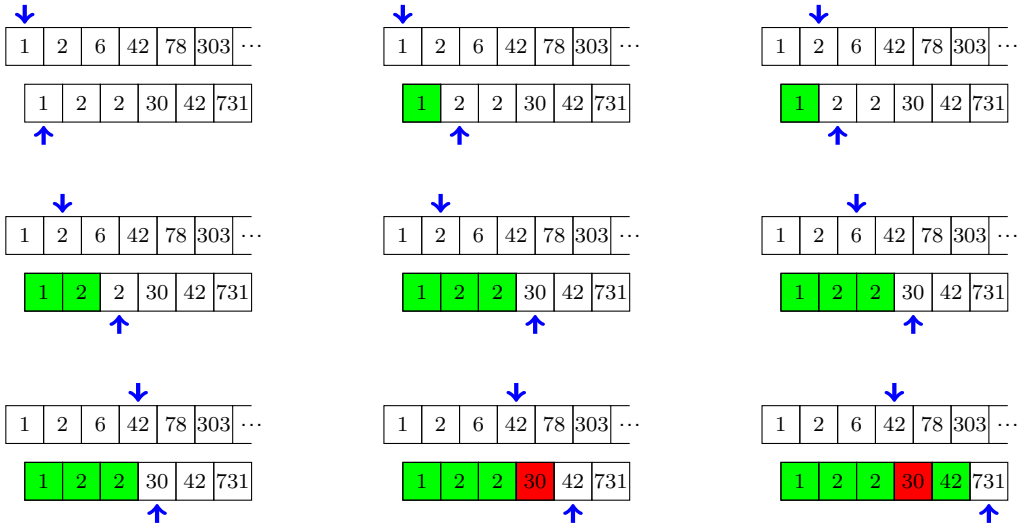
(strzałki) na początki naszych tablic/list. Sprawdzamy elementy wskazywane przez wskaźniki. Oznaczmy przez $aktualna$ liczbę na którą obecnie pokazuje wskaźnik w liście/tablicy liczb wyróżnionych, a $zapytanie$ odpowiedni element w tablicy/liście zapytań. Mamy trzy możliwości:

1. $aktualna = zapytanie$. Jeżeli tak jest, oznacza to, że liczba we właśnie sprawdzanym zapytaniu występuje na liście liczb wyróżnionych. Zaznaczamy, że odpowiedź jest twierdząca dla tego zapytania i przesuwamy wskaźnik w prawo w naszej tablicy/liście zapytań (może się zdarzyć, że mamy więcej zapytań o tę samą liczbę).

2. $aktualna > zapytanie$. Oznacza to, że dla tego zapytania odpowiedź jest negatywna – $zapytanie$ nie występuje wśród liczb wyróżnionych. Zapisujemy odpowiednią odpowiedź i przechodzimy wskaźnikiem w tablicy/liście zapytań w prawo.

3. $aktualna < zapytanie$. Oznacza to, że $aktualna$ już nie będzie nam potrzebna, bo wszystkie pozostałe zapytania są od niej większe. Musimy się przesunąć do kolejnej liczby w tablicy/liście liczb wyróżnionych. Przesuwamy odpowiedni wskaźnik w prawo.

Poniżej znajduje się graficzna wizualizacja pierwszych kroków tego algorytmu dla naszego przykładu:



Mamy zatem algorytm, który odpowiada na wszystkie zapytania i możemy zastanowić się jaka jest jego złożoność. Zauważmy, że w każdym kroku przesuwamy któryś ze wskaźników w prawo. Zatem sumarycznie wykonamy $O(Q + P)$ kroków, gdzie P oznacza liczbę elementów w tablicy liczb wyróżnionych.

Zauważmy, że nie musimy nawet trzymać całej listy/tablicy liczb wyróżnionych w pamięci. Zamiast tego będziemy generować liczby wyróżnione po jednej – robiliśmy to już w powyższej symulacji – i pamiętać tylko jedną, aktualnie potrzebną. Zamiast przesunięcie wskaźnika w prawo do następnej liczby w tablicy, będziemy po prostu generować w tym momencie następną liczbę i zapominać o poprzedniej.

Ostatecznie zatem algorytm będzie wczytywał wszystkie zapytania do listy/tablicy, sortował je, a następnie używał metody dwóch wskaźników (z drugim wskaźnikiem generującym kolejne liczby wyróżnione) do znalezienia odpowiedzi na podane zapytania. Zauważmy, że oprócz samego zapytania, musimy też trzymać, które to zapytanie było na wejściu. Dlatego w obu wzorcowych rozwiązaniach używamy par, które składają się z zapytania oraz jego indeksu.

Złożoność czasowa wzorcowego rozwiązania to $O(Q \log(Q) + P)$, a pamięciowa to $O(Q)$.

lic.py

```
# Maksymalna liczba, o którą może pojawić się zapytanie.
LIMIT = 5000000000


# Funkcja f zdefiniowana w zadaniu - zwraca sumę cyfr liczby n.
def f(n):
  # Deklarujemy zmienną na sumę.
```


Olimpiada finansowana jest ze środków Ministerstwa Edukacji Narodowej w ramach zadania publicznego „Organizacja i przeprowadzenie olimpiad i turniejów w latach szkolnych 2019/2020, 2020/2021, 2021/2022".

```
8     suma = 0
9     # Dopóki liczba n nie jest równa zero,
10    while n != 0:
11      # dodaj do sumy ostatnią cyfrę (resztę z dzielenie przez 10)
12      suma += n % 10
13      # i usuń tę cyfrę (podziel tę liczbę przez 10).
14      n //= 10
15
16    # Na końcu zwróć sumę.
17    return suma
18
19
20  # Funkcja g zdefniowana w zadaniu, zwraca n + f(n) * f(n).
21  def g(n):
22    # Zapisujemy f(n) jako zmienną, aby nie liczyć tej funkcji dwukrotnie.
23    fn = f(n)
24    # Zwracamy wartość funkcji g dla liczby n.
25    return n + fn * fn
26
27
28  def main():
29    # Deklarujemy i wczytujemy liczbę zapytań.
30    q = int(input())
31
32    # Deklarujemy wektor na zapytania.
33    # Zapytania będziemy trzymali jako parę określającą liczbę, o którą
34    # pytamy, a także indeks zapytania (które to było zapytanie w kolejności).
35    M = []
36    for i in range(q):
37      m = int(input())
38      M.append((m, i))
39
40    # Sortujemy wszystkie zapytania.
41    M.sort()
42
43    # Deklarujemy wektor na nasze odpowiedzi, które będziemy uzupełniać.
44    odpowiedz = [False for _ in range(q)]
45
46    # Zaczynamy od n = 1.
47    n = 1
48
49    # Deklarujemy wskaźnik na wektor M, początkowo równy zero.
50    wsk_m = 0
51
52    # Dopóki n jest nie większe niż nasz limit:
53    while n <= LIMIT:
54      # Sprawdzamy czy aktualna wartość w wektorze M jest mniejsza od n.
55      while wsk_m < q and M[wsk_m][0] < n:
56        # Jeżeli tak, to musimy przesunąć nasz wskaźnik w prawo.
57        wsk_m += 1
58
59      # Sprawdzamy teraz czy aktualna wartość w wektorze M jest równa n.
60      while wsk_m < q and M[wsk_m][0] == n:
61        # Jeżeli tak, to daną wartość możemy uzyskać w procesie,
62        # dlatego ustawiamy odpowiednią odpowiedź na true.
63        odpowiedz[M[wsk_m][1]] = True
64        # Następnie przesuwamy wskaźnik w prawo.
65        wsk_m += 1
```


Olimpiada finansowana jest ze środków Ministerstwa Edukacji Narodowej w ramach zadania publicznego „Organizacja i przeprowadzenie olimpiad i turniejów w latach szkolnych 2019/2020, 2020/2021, 2021/2022".

```
66
67      # Zamieniamy n na g(n).
68      n = g(n)
69
70
71    # Finalnie, dla wszystkich zapytań wypisujemy odpowiednie odpowiedzi.
72    for i in range(q):
73      if odpowiedz[i]: print("TAK")
74      else: print("NIE")
75
76
77  main()
```

lic.cpp

```cpp
1  #include "bits/stdc++.h"
2
3  using namespace std;
4
5  // Maksymalna liczba, o którą może pojawić się zapytanie.
6  const long long LIMIT = 5000000000LL;
7
8  // Funkcja f zdefiniowana w zadaniu - zwraca sumę cyfr liczby n.
9  int f(long long n) {
10   // Deklarujemy zmienną na sumę.
11   int suma = 0;
12   // Dopóki liczba n nie jest równa zero,
13   while (n != 0) {
14     // dodaj do sumy ostatnią cyfrę (resztę z dzielenie przez 10)
15     suma += n % 10;
16     // i usuń tę cyfrę (podziel tę liczbę przez 10).
17     n /= 10;
18   }
19   // Na końcu zwróć sumę.
20   return suma;
21 }
22
23 // Funkcja g zdefniowana w zadaniu, zwraca n + f(n) * f(n).
24 long long g(long long n) {
25   // Zapisujemy f(n) jako zmienną, aby nie liczyć tej funkcji dwukrotnie.
26   int fn = f(n);
27   // Zwracamy wartość funkcji g dla liczby n.
28   return n + fn * fn;
29 }
30
31 int main() {
32   // Deklarujemy i wczytujemy liczbę zapytań.
33   int q;
34   cin >> q;
35
36   // Deklarujemy wektor na zapytania.
37   // Zapytania będziemy trzymali jako parę określającą liczbę, o którą
38   // pytamy, a także indeks zapytania (które to było zapytanie w kolejności).
39   vector <pair <long long, int>> M(q);
40   for (int i=0; i<q; i++) {
41     long long m;
42     cin >> m;
43     M[i] = make_pair(m, i);
44   }
45
```


Olimpiada finansowana jest ze środków Ministerstwa Edukacji Narodowej w ramach zadania publicznego „Organizacja i przeprowadzenie olimpiad i turniejów w latach szkolnych 2019/2020, 2020/2021, 2021/2022".

```cpp
  // Sortujemy wszystkie zapytania.
  sort(M.begin(), M.end());

  // Deklarujemy wektor na nasze odpowiedzi, które będziemy uzupełniać.
  // Początkowo wszystko jest ustawione na 0 (czyli fałsz).
  vector <bool> odpowiedz(q);

  // Zaczynamy od n = 1.
  long long n = 1;

  // Deklarujemy wskaźnik na wektor M, początkowo równy zero.
  int wsk_m = 0;

  // Dopóki n jest nie większe niż nasz limit:
  while (n <= LIMIT) {
    // Sprawdzamy czy aktualna wartość w wektorze M jest mniejsza od n.
    while (wsk_m < q and M[wsk_m].first < n) {
      // Jeżeli tak, to musimy przesunąć nasz wskaźnik w prawo.
      wsk_m++;
    }
    // Sprawdzamy teraz czy aktualna wartość w wektorze M jest równa n.
    while (wsk_m < q and M[wsk_m].first == n) {
      // Jeżeli tak, to daną wartość możemy uzyskać w procesie,
      // dlatego ustawiamy odpowiednią odpowiedź na true.
      odpowiedz[M[wsk_m].second] = true;
      // Następnie przesuwamy wskaźnik w prawo.
      wsk_m++;
    }
    // Zamieniamy n na g(n).
    n = g(n);
  }

  // Finalnie, dla wszystkich zapytań wypisujemy odpowiednie odpowiedzi.
  for (int i=0; i<q; i++) {
    if (odpowiedz[i]) cout << "TAK\n";
    else cout << "NIE\n";
  }
}
```


Olimpiada finansowana jest ze środków Ministerstwa Edukacji Narodowej w ramach zadania publicznego „Organizacja i przeprowadzenie olimpiad i turniejów w latach szkolnych 2019/2020, 2020/2021, 2021/2022".