Dwukrotność sumy cyfr (rozwiązanie)

Autorzy zadania: **Bartosz Łukasiewicz, Jacek Tomasiewicz**
Opracowanie: **Krzysztof Bartoszek, Marcin Kurowski**
Opis rozwiązania: **Bartosz Kostka**

Olimpiada Informatyczna Juniorów

W zadaniu ten mamy zdefiniowany ciąg A, który rozpoczyna się od liczby X, tj. $A_1 = X$, a każdy kolejny element tego ciągu jest podwojoną sumą cyfr poprzedniego elementu. Chcemy znaleźć N-ty element tego ciągu.

Najprostszą, brutalną metodą rozwiązania tego zadania jest wygenerowanie całego ciągu do N-tego elementu zgodnie z definicją.

sum_brut.py

```python
# Funkcja pomocnicza do zliczania sumy cyfr liczby.
def suma_cyfr(liczba):
  wynik = 0
  while liczba != 0:
    wynik += liczba % 10
    liczba //= 10
  return wynik

def main():
  (N, X) = tuple(map(int, input().split()))
  # Ustawiamy obecny element na X.
  A_i = X
  for i in range(2, N+1):
    # Każdy kolejny element jest dwukrotnością sumy cyfr poprzedniego.
    A_i = 2*suma_cyfr(A_i)
  print(A_i)

main()
```

sum_brut.cpp

```cpp
#include <iostream>

using namespace std;

// Funkcja pomocnicza do zliczania sumy cyfr liczby.
int suma_cyfr(long long liczba) {
  int wynik = 0;
  while (liczba != 0) {
    wynik += liczba % 10;
    liczba /= 10;
  }
  return wynik;
}

int main() {
  long long N, X;
  cin >> N >> X;
  // Ustawiamy obecny element na X.
  long long A_i = X;
  for (long long i=2; i<=N; i++) {
    // Każdy kolejny element jest dwukrotnością sumy cyfr poprzedniego.
    A_i = 2*suma_cyfr(A_i);
  }
  cout << A_i << "\n";
}
```

Takie rozwiązanie pozwalało nam zdobyć 25 punktów.

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".

W zadaniu była możliwość rozwiązania jeszcze jednej prostszej wersji zadania, w której pierwszym elementem ciągu zawsze było 1 (tj. $X = 1$). Zastanówmy się zatem jak wygląda ten ciąg. Możemy użyć naszego programu powyżej, aby wypisać pierwsze kilkanaście elementów: $[1, 2, 4, 8, 16, 14, 10, 2, 4, 8, 16, 14, 10, 2, 4, 8, 16, 14, 10, 2, 4, 8, 16, 14, \ldots]$. Widzimy, że poza pierwszym elementem wszystkie elementy powtarzają się w cyklu $[2, 4, 8, 16, 14, 10]$. Ma to oczywiście sens, ponieważ jeżeli trafimy drugi raz na taki sam element w ciągu (w naszym przypadku 2), to ponieważ kolejny element zależy jedynie od jednego poprzedniego, to „zapętlimy się" i już zawsze będziemy działać w tym samym cyklu. Pozostaje nam zatem wykorzystać tą obserwację w kodzie.

sum_x1.py

```python
def main():
  (N, X) = tuple(map(int, input().split()))
  # Przypadek brzegowy gdy N=1:
  if N == 1: print(1)
  else:
    # To jest nasz cykl powtarzających się elementów dla X=1.
    cykl = [2, 4, 8, 16, 14, 10]
    # Wypisz odpowiedni element z cyklu (przesuwamy indeks o 2,
    # jako że drugi element ciągu to zerowy element cyklu).
    print(cykl[(N-2)%6])

main()
```

sum_x1.cpp

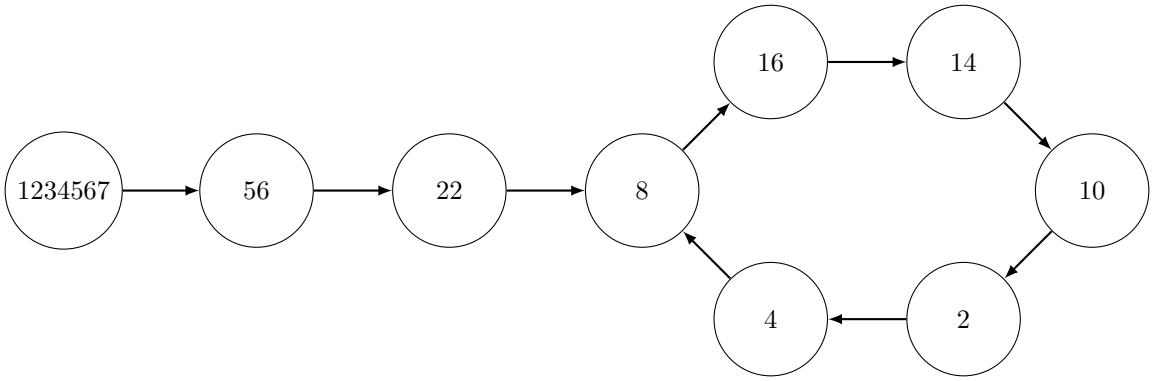
```cpp
#include <iostream>
#include <vector>

using namespace std;

int main() {
  long long N, X;
  cin >> N >> X;
  // Przypadek brzegowy gdy N=1:
  if (N == 1) cout << 1 << "\n";
  else {
    // To jest nasz cykl powtarzających się elementów dla X=1.
    vector <long long> cykl = {2, 4, 8, 16, 14, 10};
    // Wypisz odpowiedni element z cyklu (przesuwamy indeks o 2,
    // jako że drugi element ciągu to zerowy element cyklu).
    cout << cykl[(N-2)%6] << "\n";
  }
}
```

Taki kod pozwalał nam zdobyć 50 punktów.

Zastanówmy się teraz jak wykorzystać tą obserwację do rozwiązania całego zadania. Powiedzieliśmy już, że jeżeli w danym ciągu jakiś element się powtórzy, to wszystkie elementy pomiędzy tymi powtarzającymi się elementami stworzą cykl, którym będziemy się poruszać już w nieskończoność.

Dla przykładu, rozważmy $X = 1234567$.

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".



Mamy tutaj cykl, który rozpoczyna się przy elemencie równym 8 i prowadzący do tego cyklu "ogon".

Zastanówmy się teraz jak długie mogą być cykl i "ogon". Zauważmy, że suma cyfr liczby nie przekraczającej 10^{18} może wynieść co najwyżej $18 \cdot 9 = 162$. To z kolei oznacza, że poza pierwszym elementem, wszystkie następne nie będą przekraczały $2 \cdot 162 = 324$. A zatem różnych liczb, które pojawią się w naszym ciągu może być co najwyżej 325 (przy czym ta liczba obejmuje zarówno liczby z cyklu, jak i "ogona"). Generujemy zatem kolejne elementy tak długo, dopóki któryś z nich się nie powtórzy, a następnie na podstawie długości cyklu wypisujemy odpowiednią liczbę.

sum.py

```python
def main():
  (N, X) = tuple(map(int, input().split()))
  # Ustawiamy obecny element na X.
  # Wstawiamy dodatkowo element zerowy (aby indeksy się zgadzały).
  A = [0, X]
  # Dodatkowo będziemy utrzymywali słownik indeks, który dla każdego
  # elementu ciągu będzie wskazywał jego pierwsze wystąpienie.
  indeks = {X: 1}
  for i in range(2, N+1):
    # Każdy kolejny element jest dwukrotnością sumy cyfr poprzedniego.
    A.append(2*suma_cyfr(A[-1]))
    # Jeżeli ostatni element ma już wartość w indeks, to znaleźliśmy cykl.
    if A[-1] in indeks.keys():
      # Ustaw początek i koniec ciągu.
      start_cyklu = indeks[A[-1]]
      koniec_cyklu = i
      # Liczymy długość cyklu.
      dlugosc_cyklu = koniec_cyklu - start_cyklu
      # Znajdujemy odpowiednią pozycję N-tego elementu w cyklu.
      indeks_na_cyklu = (N - start_cyklu) % dlugosc_cyklu
      # I wypisujemy odpowiedni element z cyklu.
      print(A[start_cyklu + indeks_na_cyklu])
      return
    # Ustawiamy indeks aktualnego elementu na i.
    indeks[A[-1]] = i
  # Jeżeli nie doszliśmy do ciągu, po prostu wypisz wynik.
  print(A[N])

main()
```

sum.cpp

```cpp
int main() {
  long long N, X;
  cin >> N >> X;
  // Ustawiamy obecny element na X.
  // Wstawiamy dodatkowo element zerowy (aby indeksy się zgadzały).
  vector <long long> A = {0, X};
```


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".

```
23    // Dodatkowo będziemy utrzymywali słownik indeks, który dla każdego
24    // elementu ciągu będzie wskazywał jego pierwsze wystąpienie.
25    map <long long, int> indeks = {{X, 1}};
26    for (int i=2; i<=N; i++) {
27      // Każdy kolejny element jest dwukrotnością sumy cyfr poprzedniego.
28      A.push_back(2*suma_cyfr(A[i-1]));
29      // Jeżeli ostatni element ma już wartość w indeks, to znaleźliśmy cykl.
30      if (indeks.find(A[i]) != indeks.end()) {
31        // Ustaw początek i koniec ciągu.
32        int start_cyklu = indeks[A[i]];
33        int koniec_cyklu = i;
34        // Liczymy długość cyklu.
35        int dlugosc_cyklu = koniec_cyklu - start_cyklu;
36        // Znajdujemy odpowiednią pozycję N-tego elementu w cyklu.
37        int indeks_na_cyklu = (N - start_cyklu) % dlugosc_cyklu;
38        // I wypisujemy odpowiedni element z cyklu.
39        cout << A[start_cyklu + indeks_na_cyklu] << "\n";
40        return 0;
41      }
42      // Ustawiamy indeks aktualnego elementu na i.
43      indeks[A[i]] = i;
44    }
45    // Jeżeli nie doszliśmy do ciągu, po prostu wypisz wynik.
46    cout << A[N] << "\n";
47  }
```


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".