Liczby pechowe (rozwiązanie)

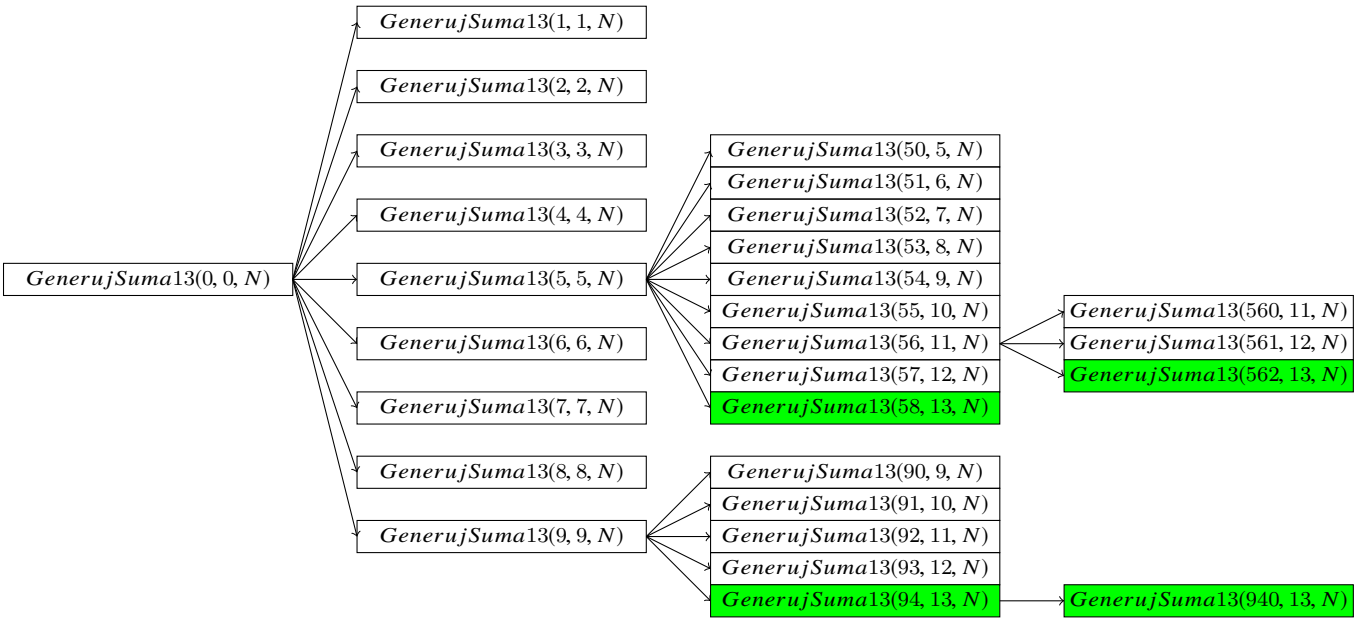
XIV OIJ, zawody I stopnia, tura ukryta
14 października 2019 – 13 stycznia 2020

Autor zadania: **Karol Pokorski**
Opracowanie: **Michał Niedziółka, Tymoteusz Wiśniewski**

Olimpiada Informatyczna Juniorów

Pierwszym możliwym pomysłem na to zadanie jest wygenerowanie wszystkich liczby nie większych od N, które mają sumę cyfr równą 13 (a potem pozostawienie tylko tych, które mają 13 jako dwie kolejne cyfry zapisu). Do generowania takich liczb użyjemy rekurencyjnej funkcji $GenerujSuma13(liczba, suma\_cyfr, N)$, która będzie budowała liczby dostawiając kolejne cyfry z prawej strony, gwarantując aby suma cyfr nie przekroczyła 13. Argumentami tej funkcji będą kolejno $liczba$, którą zbudowaliśmy do tej pory, jej $suma\_cyfr$ oraz ograniczenie górne dane przez N.

Na poniższym schemacie możemy zobaczyć przykładowe wywołanie takiej funkcji (część rozgałęzień została pominięta dla czytelności). Rozpoczynamy od liczby 0 (z sumą cyfr równą 0). Następnie przeglądamy wszystkie cyfry, które możemy dodać i wywołujemy się rekurencyjnie dla liczb z dodaną cyfrą. Jeżeli liczba ma sumę cyfr równą 13 (takie liczby zaznaczone są na zielono na poniższym schemacie), to dodajemy ją do odpowiedniej listy/tablicy.



pec_suma13.py

```
# Funkcja generująca liczby o sumie cyfr 13, mniejszych od N.
# Aktualnie rozpatrywana liczba to ‘liczba‘, o sumie cyfr ‘suma_cyfr‘.
# Liczby zapisywane są w liście ‘liczby_suma13‘.
def GenerujSuma13(liczba, suma_cyfr, N, liczby_suma13):
  # Jeżeli liczba jest większa niż N, to wychodzimy z funkcji.
  if liczba > N: return
  # Jeżeli suma cyfr wynosi 13, to dopisujemy liczbę do listy.
  if suma_cyfr == 13: liczby_suma13.append(liczba)
  # Próbujemy dopisać nową cyfrę do już utworzonej liczby.
  # Możemy ograniczyć się do cyfr z przedziału [0, 13-suma_cyfr].
  for nowa_cyfra in range(min(10, 14-suma_cyfr)):
    # Dodatkowy warunek, abyśmy nie próbowali stworzyć liczb typu 0000000.
    if liczba == 0 and nowa_cyfra == 0: continue
    # Dodajemy do liczby cyfrę na koniec
    nowa_liczba = 10 * liczba + nowa_cyfra
    # i zwiększamy sumę cyfr,
    nowa_suma_cyfr = suma_cyfr + nowa_cyfra
    # po czym wywołujemy funkcję rekurencyjnie.
    GenerujSuma13(nowa_liczba, nowa_suma_cyfr, N, liczby_suma13)
```


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".

pec_suma13.cpp

```cpp
// Deklarujemy wektor, który wypełnimy liczbami o sumie cyfr 13.
vector <long long> liczby_suma13;

// Funkcja generująca liczby o sumie cyfr 13, mniejszych od N.
// Aktualnie rozpatrywana liczba to ‘liczba‘, o sumie cyfr ‘suma_cyfr‘.
// Liczby zapisywane są w tablicy ‘liczby_suma13‘.
void GenerujSuma13(long long liczba, int suma_cyfr, long long N) {
  // Jeżeli liczba jest większa niż N, to wychodzimy z funkcji.
  if (liczba > N) return;
  // Jeżeli suma cyfr wynosi 13, to dopisujemy liczbę do listy.
  if (suma_cyfr == 13) liczby_suma13.push_back(liczba);
  // Próbujemy dopisać nową cyfrę do już utworzonej liczby.
  // Możemy ograniczyć się do cyfr z przedziału [0, 13-suma_cyfr].
  for (int nowa_cyfra=0; nowa_cyfra<=min(9, 13-suma_cyfr); nowa_cyfra++) {
    // Dodatkowy warunek, abyśmy nie próbowali stworzyć liczb typu 0000000.
    if (liczba == 0 and nowa_cyfra == 0) continue;
    // Dodajemy do liczby cyfrę na koniec
    long long nowa_liczba = 10 * liczba + nowa_cyfra;
    // i zwiększamy sumę cyfr,
    int nowa_suma_cyfr = suma_cyfr + nowa_cyfra;
    // po czym wywołujemy funkcję rekurencyjnie.
    GenerujSuma13(nowa_liczba, nowa_suma_cyfr, N);
  }
}
```

Niestety, tych liczb jest dość sporo, a nie wszystkie z nich zawierają 13 w zapisie dziesiętnym. Zmieńmy zatem trochę nasz pomysł – co się stanie, jeśli z liczby pechowej usuniemy dowolne wystąpienie fragmentu 13 i zastąpimy ten fragment dwoma zerami (00)? Otrzymamy oczywiście liczbę o sumie cyfr równej 9. Możemy zatem, zamiast wypisywać wszystkie liczby o sumie cyfr równej 13, znaleźć wszystkie liczby o sumie cyfr równej 9 (których jest znacznie mniej), a następnie zamienić każde wystąpienie dwóch zer obok siebie (także wśród zer wiodących) na liczbę 13, uzyskując liczbę pechową. Dla przykładu, z liczby 1230010002 możemy uzyskać następujące pechowe liczby: $123001\underline{13}2$, $123001\underline{13}02$, $123\underline{13}10002$, $\underline{13}1230010002$, $\underline{13}01230010002$, itd.. Ta obserwacja pozwala nam zaimplementować już rozwiązanie mieszczące się w limicie czasowym oraz pamięciowym.

pec.py

```python
# Funkcja generująca liczby o sumie cyfr 9, mniejszych od N.
# Aktualnie rozpatrywana liczba to ‘liczba‘, o sumie cyfr ‘suma_cyfr‘.
# Liczby zapisywane są w liście ‘liczby_suma9‘.
def GenerujSuma9(liczba, suma_cyfr, N, liczby_suma9):
  # Jeżeli liczba jest większa niż N, to wychodzimy z funkcji.
  if liczba > N: return
  # Jeżeli suma cyfr wynosi 9, to dopisujemy liczbę do listy.
  if suma_cyfr == 9: liczby_suma9.append(liczba)
  # Próbujemy dopisać nową cyfrę do już utworzonej liczby.
  # Możemy ograniczyć się do cyfr z przedziału [0, 9-suma_cyfr].
  for nowa_cyfra in range(10-suma_cyfr):
    # Dodatkowy warunek, abyśmy nie próbowali stworzyć liczb typu 0000000.
    if liczba == 0 and nowa_cyfra == 0: continue
    # Dodajemy do liczby cyfrę na koniec
    nowa_liczba = 10 * liczba + nowa_cyfra
    # i zwiększamy sumę cyfr,
    nowa_suma_cyfr = suma_cyfr + nowa_cyfra
    # po czym wywołujemy funkcję rekurencyjnie.
    GenerujSuma9(nowa_liczba, nowa_suma_cyfr, N, liczby_suma9)


def main():
```


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".

```
  # Wczytujemy ograniczenie górne.
  N = int(input())
  # Deklarujemy listę, którą wypełnimy liczbami o sumie cyfr 9.
  liczby_suma9 = []
  # Wywołujemy rekurencyjną funkcję, która szuka liczb o sumie cyfr 9.
  GenerujSuma9(0, 0, N, liczby_suma9)

  # Deklarujemy listę na liczby pechowe.
  liczby_pechowe = []
  # Dla każdej liczby o sumie cyfr 9,
  for liczba in liczby_suma9:
    # wstawiamy "13" w każde możliwe miejsce,
    div = 1
    # o ile mieścimy się w limicie,
    while liczba + 13 * div <= N:
      # oraz te miejsce jest wypełnione zerami.
      if liczba // div % 100 == 0:
        liczby_pechowe.append(liczba + 13 * div)
      div *= 10

  # Jako że liczby utworzone wyżej mogą się powtarzać, to usuwamy duplikaty.
  liczby_pechowe = list(set(liczby_pechowe))

  # Finalnie wypisujemy liczby pechowe.
  print(len(liczby_pechowe))

main()
```

pec.cpp

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

using namespace std;

// Deklarujemy wektor, który wypełnimy liczbami o sumie cyfr 9.
vector <long long> liczby_suma9;

// Funkcja generująca liczby o sumie cyfr 9, mniejszych od N.
// Aktualnie rozpatrywana liczba to ‘liczba‘, o sumie cyfr ‘suma_cyfr‘.
// Liczby zapisywane są w tablicy ‘liczby_suma9‘.
void GenerujSuma9(long long liczba, int suma_cyfr, long long N) {
  // Jeżeli liczba jest większa niż N, to wychodzimy z funkcji.
  if (liczba > N) return;
  // Jeżeli suma cyfr wynosi 9, to dopisujemy liczbę do listy.
  if (suma_cyfr == 9) liczby_suma9.push_back(liczba);
  // Próbujemy dopisać nową cyfrę do już utworzonej liczby.
  // Możemy ograniczyć się do cyfr z przedziału [0, 9-suma_cyfr].
  for (int nowa_cyfra=0; nowa_cyfra<=9-suma_cyfr; nowa_cyfra++) {
    // Dodatkowy warunek, abyśmy nie próbowali stworzyć liczb typu 0000000.
    if (liczba == 0 and nowa_cyfra == 0) continue;
    // Dodajemy do liczby cyfrę na koniec
    long long nowa_liczba = 10 * liczba + nowa_cyfra;
    // i zwiększamy sumę cyfr,
    int nowa_suma_cyfr = suma_cyfr + nowa_cyfra;
    // po czym wywołujemy funkcję rekurencyjnie.
    GenerujSuma9(nowa_liczba, nowa_suma_cyfr, N);
  }
}

int main() {
```


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”.

```
31    // Wczytujemy ograniczenie górne.
32    long long N;
33    cin >> N;
34    // Wywołujemy rekurencyjną funkcję, która szuka liczb o sumie cyfr 9.
35    GenerujSuma9(0, 0, N);
36
37    // Deklarujemy wektor na liczby pechowe.
38    vector <long long> liczby_pechowe;
39    // Dla każdej liczby o sumie cyfr 9,
40    for (int i=0; i<(int)liczby_suma9.size(); i++) {
41      long long liczba = liczby_suma9[i];
42      // wstawiamy "13" w każde możliwe miejsce,
43      long long div = 1;
44      // o ile mieścimy się w limicie,
45      while (liczba + 13 * div <= N) {
46        // oraz te miejsce jest wypełnione zerami.
47        if (liczba / div % 100 == 0) {
48          liczby_pechowe.push_back(liczba + 13 * div);
49        }
50        div *= 10;
51      }
52    }
53
54    // Jako że liczby utworzone wyżej mogą się powtarzać, to usuwamy duplikaty.
55    sort(liczby_pechowe.begin(), liczby_pechowe.end());
56    liczby_pechowe.erase(
57      unique(liczby_pechowe.begin(), liczby_pechowe.end()),
58      liczby_pechowe.end());
59
60    // Finalnie wypisujemy liczby pechowe.
61    cout << liczby_pechowe.size() << "\n";
62  }
```


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".