Kod Graya

Oczywiście, kod dwójkowy nie jest dla nikogo z nas tajemnicą. Wiemy, że do zapisu liczby w tym systemie używamy tylko dwóch cyfr, wiemy, że wagi kolejnych pozycji są potęgami dwójki (1, 2, 4, 8, …) i zapewne bez większych trudności bylibyśmy przeliczyć dowolną liczbę dziesiętną na dwójkową i odwrotnie. Ale zer i jedynek można używać także w inne, mniej naturalne sposoby, a przykładem niech będzie tu kod nazywany od nazwiska wynalazcy **kodem Graya**.

Kod Graya jest kodem *niepozycyjnym* (co oznacza, że położenie cyfr nie dostarcza wprost informacji o wielkości liczby) i *bezwagowym* (co oznacza, że kolejnym pozycjom nie przypisuje się żadnych wag), a to sprawia, że wzajemny porządek różnych kombinacji bitów jest w kodzie Graya wyznaczany w bardzo swoisty sposób, który można najprościej wyrazić zdaniem: „*kolejna kombinacja bitów różni się od poprzedniej zmianą tylko jednego bitu*".

Przyjrzyjmy się temu na przykładzie, w którym operować będziemy ciągiem bitów o długości 3. Zerowa kombinacja to oczywiście taka, która składa się z samych zer, czyli:

000

Pierwszą otrzymamy zmieniając tylko jeden bit (tu ważne zastrzeżenie – zmieniamy możliwie najmłodszy bit, czyli taki, który znajduje się najbliżej prawego końca ciągu cyfr). W naszym przypadku jest to trywialne:

001

Drugą kombinację uzyskamy podobnie, z tym, że skrajny prawy bit jest teraz dla nas nietykalny – jego zmiana doprowadziłaby nas do już wykorzystanej kombinacji zero. Zamiast tego zmienimy więc drugi bit:

011

Za to przy kombinacji trzeciej możemy już posłużyć się bitem najmłodszym:

010

Kombinacja czwarta wymaga użycia najstarszego bitu (*czemu?*):

110

Za to piąta – najmłodszego:

111

Szóstą uzyskamy zmieniając bit środkowy:

101

A siódmą (ostatnią) – zmieniając bit najmłodszy:

100

Zauważ, że przejście od kombinacji siódmej do zerowej również spełnia wymaganie zmiany tylko jednego bitu.

Istnieje również prostsza (bo bezpośrednia – *nieiteracyjna*) metoda uzyskania dowolnego elementu kodu Gray, którą opisuje się następująco:

- jeśli *x* jest ciągiem bitów reprezentującym na *n* pozycjach pewną liczbę w naturalnym kodzie dwójkowym, to podziel ją przez dwa, a wynik umieść w *y* (pamiętamy oczywiście, że dzielenie przez dwa możemy zastąpić przesunięciem liczby o jeden bit w prawo)
- wykonaj operację *XOR* na *n* odpowiadających sobie bitach liczb *x* i *y*
- wynik da ci reprezentację liczby *x* w kodzie Graya

Sprawdźmy to na trzech bitach i dla liczby 5:

- $5_{10} = 101_2$
- $5_{10} / 2_{10} = 010_2$
- 101_2 ***xor*** $010_2 = 111_2$

co daje wynik zgodny z naszymi wcześniejszymi eksperymentami (tu dodajmy jeszcze, że rolę operatora xor pełni w języku C znak ^ (*caret*)).

Polecenie: napisz program, który wczyta ze standardowego wejścia dwie liczby całkowite: *n* oznaczające liczbę bitów i *m*, oznaczające pewną liczbę, reprezentowalną na *n* bitach, a następnie wypisze na standardowe wyjście ciąg *n* bitów odpowiadający *m*-tej kombinacji bitów w *n*-bitowym kodzie Graya

Dane wejściowe: dwie liczby całkowite

- liczba bitów *n*
  (n: $n >= 1$ i $n <= 32000$)
- numer poszukiwanej kombinacji bitów w *n*-bitowym kodzie Graya
  (m: $m >= 0$ i $m < 2^{32000}$)

Dane wyjściowe: ciąg *n* bitów składający się na *m*-tą kombinację bitów w *n*-bitowym kodzie Graya

Przykład:

Wejście:

3 5

Wyjście:

111