АНАЛИЗ НА РЕШЕНИЕТО НА ЗАДАЧА
ДВОИЧНИ ПАЛИНДРОМИ

Първото предложено решение (файл **nbinpal_slow.cpp**) е стандартното. То генерира всички числа, които в двоична бройна система имат точно `n` цифри и проверява за всяко от тях дали е симетрично. Това решение е твърде бавно и дава само около `30%` от точките.

Второто решение (файл **nbinpal.cpp**) се основава на метода динамично оптимиране, но до идеята му може да се достигне и чрез наблюдения на резултатите, които извежда бавното решение или чрез различни комбинаторни съображения. При него се използва, че броят на двоичните палиндроми с `n` цифри е равен на броят на двоичните палиндроми с `(n-2)` цифри умножен по две. Ако означим с `dp[i]` броят на двоичните `i`-цифрени палиндроми, то:

`dp[0]=1` (по подразбиране)

`dp[1]=2 (0 и 1)`

`dp[2]=1 (11)`

`dp[3]=2 (101 и 111)`

`dp[4]=2 (1001 и 1111)`

`dp[5]=4 (10001, 10101, 11011 и 11111)`

`dp[6]=4 (100001, 101101, 110011 и 111111)`

`dp[7]=8 (1000001, 1001001, 1010101, 1011101, 1100011, 1101011, 1110111 и 1111111)`

Забелязва се, че `dp[6]=dp[4]+dp[2]+dp[0]`, тъй като 6-цифрените палиндроми са 1**1001**1, 1**1111**1 (4-цифрените, оградени с 1), 10**11**01 (2-цифрените, оградени с 10) и 100001 (0-цифрените, оградени с 100).

Аналогично: `dp[7]=dp[5]+dp[3]+dp[1]`, защото 7-цифрените палиндроми са 1**10001**1, 1**10101**1, 1**11011**1, 1**11111**1 (5-цифрените, оградени с 1), 10**101**01, 10**111**01 (3-цифрените, оградени с 10) и 100**0**001, 100**1**001 (1-цифрените, оградени с 100).

Но `dp[4]=dp[2]+dp[0]`. Следователно `dp[6]=dp[4]+dp[4]=2*dp[4]`.

Аналогично: `dp[5]=dp[3]+dp[1]`. Следователно `dp[7]=dp[5]+dp[5]=2*dp[5]`. От тук се извежда рекурентната зависимост:

`dp[1]=1`

`dp[2]=1`

`dp[i]=2*dp[i-2]`, за всяко `i>2`.

Трябва да се разглежда като частен случай `n=1`, тъй като тогава се брои и нулата. Този алгоритъм може да се реализира линейно, дори без използването на масив (файл **nbinpal_cycle.cpp**).

С още малко наблюдения или доказателство ☺ се вижда, че:

1. `dp[n]`$=2^{n/2-1}$, когато `n` е четно

2. dp[n]=$2^{(n+1)/2 - 1}$, когато n е нечетно
   Наистина:

n	**dp[n]**
2	1
4	2
6	4
8	8
10	16
12	32
14	64

n	**dp[n]**
1	2
3	2
5	4
7	8
9	16
11	32
13	64

Това наблюдение води до константно решение, ако се използват побитовите операции за повдигане 2 на степен! Отново трябва да се разглежда като частен случай n=1, тъй като тогава се брои и нулата.

Поради ограничението на n, то трябва да се работи с 64-битов тип данни. В противен случай се губят 50% от точките.

Автор: Велислава Емилова