

Trololo (Анализ)

Оригинално задачата (тогава именувана *EllysPopcounts*) беше предвидена и предложена в TopCoder за Div1 250 (SRM 543), но на нейно място беше ползвана малко по-лесна такава.

След осмисляне на задачата можем да забележим, че тя се свежда до това да се преброят единиците в двоичния запис на числата в интервала $[A, B]$. Този брой, също наричан *popcount* (population count), е сравнително известен и съществуват различни начини да бъде намерен ефективно за едно цяло число. Някои от по-ефективните алгоритми използват едва няколко операции. Подобна задача имаше и миналата година, но там беше върху рандом числа и трябваше да се използва именно някой от тези алгоритми.

За тази задача, обаче, пет операции на число не е достатъчно добре. Полошо, една операция на число не е достатъчно добре. Тъй като интервалът може да покрива до милиард числа на тест, това означава, че ще ни трябва около минута за целия вход. Това изобщо не е достатъчно бързо.

Хмм, а възможно ли е да направим алгоритъм за броене на единиците в двоичния запис на число за под една операция на число? Всъщност не.

Ъъ, тогава как да решим задачата?

Това, което не сме ползвали до сега, е, че числата, които търсим, са в непрекъснат интервал. Ще се опитаме да измислим сравнително по-бавен алгоритъм, който, обаче, намира сумата на *popcount*-ите на цял интервал от числа.

За простота, вместо да мислим такъв алгоритъм за произволен интервал $[A, B]$, ще измислим такъв за $[1, N]$. Това е често срещан трик, тъй като за да намерим $\text{ans}([A, B])$ е достатъчно да изчислим $\text{ans}([1, B]) - \text{ans}([1, A-1])$. Тъй като числото 0 не добавя нищо към отговора, можем да разглеждаме и интервали $[0, N]$, вместо $[1, N]$.

Първо ще разгледаме малко по-простия вариант, в който N е степен на две, минус едно, тоест търсим сумата на попкаунтите на числата в интервала $[0, 2^k - 1]$. Това е по-лесно, тъй като всички числа са с еднаква дължина в двоичен запис и можем да имаме всяка една комбинация от нули и единици за всеки един бит.

Нека разгледаме първите няколко:

1. $[0, 1] = 1$
2. $[0, 3] = 4$
3. $[0, 7] = 12$
4. $[0, 15] = 32$
5. $[0, 31] = 80$

Като цяло, $\text{ans}([0, 2^k-1]) = K * 2^{k-1}$.

Това, ако се замислим, е логично. В число с K бита, всеки бит може да е или 0, или 1, като е 0 в половината от всички възможности и 1, в останалите. Половината възможности (тоест тези, в които е 1) са 2^{k-1} . Следователно общият брой единици е $K * 2^{k-1}$ (K пъти това число - по веднъж за всеки бит).

Това ни е достатъчно да решим и опростената задача за $[0, N]$, където N е произволно естествено число. Нека представим N в двоичен запис. За пример ще ползваме $N = 1337_{(10)} = 10100111001_{(2)}$.

В колко от числата най-старшият бит е 1? Логично, в тези от $000000000_{(2)}$ до $100111001_{(2)}$ или 314 на брой. Тоест толкова на брой, колкото е N без най-старшия си бит плюс едно. При всички предходни е бил 0, като всички предходни са... именно степен на две. Значи можем да прибавим и тях (ползвайки горната формула) към отговора и да не ги мислим повече.

Сега остава да видим какво се случва в числата от $1000000000_{(2)}$ до $10100111001_{(2)}$, като вече сме преброили най-старшия бит, както и всички числа с по-малък брой двоични цифри. Но след като сме го преброили, то може просто да го игнорираме. Следователно, остава към отговора да прибавим същата подзадача, само че за числата от $[000000000_{(2)}, 100111001_{(2)}]$. Което, ако забелязвате, е отново задача от същия тип и можем да решим рекурсивно. Базисните случаи са (примерно) число с една единствена двоична цифра.

Това е! Единствено трябва да внимаваме, че макар и входът да не надхвърля 32-битов `int`, то отговорът би могъл. Следователно трябва да ползваме `long long` за да сумираме броя битове.

Сложността на това решение е $O(\log(N))$ за да намерим $\text{ans}([0, N])$, тъй като в най-лошия случай трябва да разгледаме всеки отбитовите на числото. Тъй като във всеки тест ползваме тази функция точно два пъти: $\text{ans}([0, B]) - \text{ans}([0, A-1])$, то сложността на целия алгоритъм за всеки тест е $O(\log(N))$.