Задача 1. Грозни числа

Автор: Тодор Петров Петров

Първото нещо, което може да се забележи, е че сумата на цифрите в двоичното представяне на едно число е всъщност броя на единиците в него и следователно е не повече от броя на цифрите в двоичното представяне на числото N. В този смисъл при N = 10^18, то броя на цифрите в двоичното представяне е не повече от 60-70. Следователно ако отбележим с DS(X) сумата на цифрите на числото X, то за всяко X <= N, DS(X) < 70.

Ако намерим броя C на числата Y, които могат да се представят като Y = X + DS(X) и Y <= N, то тогава броят на грозните числа е N – C. Следователно задачата се свежда до това да намерим колко са числата от вида X + DS(X).

Казахме, че DS(X) < 70. Нека сега разгледаме някои свойства на събирането в двоична бройна система. Интересно е ако имаме числата X и Y и P = X + Y, то ще има ли някаква част от числото X, която ще си остане същата и след събирането с Y. Събирането в двоична бройна система е аналогично на събирането в десетична, записват се числата едно под друго като се подравняват от дясно и към числото с по-малко цифри се записват нули в началото. След това се започва от последните цифри и се събират. Очевидно сумата може да бъде 0 (0 + 0), 1 (1 + 0 или 0 + 1) и 2 (1 + 1), при 0 или 1 записваме съответно 0 или 1 и продължаваме нататък към предните цифри, в случея на 2 записваме 0 и имаме едно на ум, което използваме при следващото събиране. Гледайки този алгоритъм стигаме до следния извод:

Ако имаме двоичното число x1x2…..xk, където x1, x2, …., xk са или 0 или 1. И също така имаме, числото y1…yl0z1….zp, където y1…yl са 0 или 1, z1….,zp са или 0 или 1 и p >= k, то x1x2….xk + y1…yl0z1….zp = y1….ylT1….T(p+1), където T1...,T(p+1) са 0 или 1. С други думи началото на числото се запазва и след събирането. Нещо повече z1, z2, …, z(p – k) са само единици тъй като ако някое от числата z1, z2, …, z(p – k) е 0 – нека това е zj, то y1…yl0z1….zp + x1, x2, …., xk = y1y2..yl0z1..z(j – 1)T1…T(p-j-1). Ако вземем събирането X + DS(X) и DS(X) = x1…xk, а X = y1…yl0z1….zp. то можем да го представим като P|S1 + S2 = P|S3, като DS(P) + DS(S1) = S2 и според горните описания S1 + S2 = S3 и също така броя на цифрите на S2 и S1 е един и същ т.е. към по-малкото число се добавят нули в началото. От тук извеждаме следния алгоритъм:

1) Намираме колко са различните S1.

   Нека Max(DS(X)) е максималното DS(X) за X <= N. Тогава S2 може да бъде число между 1 и Max(DS(X)) и броя на цифрите на S2 е броя на цифрите на Max(DS(X)) следователно S1 е от вида 1....1|S2, като 1...1|S2 <= N. (Като имаме единици в началото само ако S2 започва с 1).

Пример:

N = 13 (в двоична бройна система 1101) => Max(DS(X)) = 4 -> когато всички цифри са единици (дори в случея не може да бъде 1111 т.к. 1111 > 1101, но за по-лесно смятане взимаме Max(DS(X)) да е броя на цифрите на N в двоична бройна система).

Следователно S2 може да бъде 001, 010, 011, 100 (т.к. Max(DS(X)) = 4(десетична бр. система) = 100 (в двоична бр. Система)).

От тук намираме S1:

001; 010; 011; 100; 1100 (единствено 100 започва с 1 => можем да поставяме единици в началото докато имаме S1 <= N)

2) Като втора стъпка трябва да намерим всички суми на S1 със S2.

   От примера по-горе: 001 + 001 = 010; 001 + 010 = 011; 001 + 011 = 100; 001 + 100 = 101; 001 + 1100 = 1101 и т.н.

3) Това е междинна стъпка на която трябва да се направи нещо много важно, но ще го обясним по-късно тъй като на този етап не е очевидно.

4) S1 + S2 = S3, P|S1 + S2 = P|S3 и DS(P) + DS(S1) = S2 => DS(P) = S2 – DS(S1). Трябва да определим колко различни P|S3 могат да се получат, което е C, а както казахме по рано отговорът на задачата е N – C. Тук трябва да забележим, че ако P1 има X единици и P2 има Y единици и X != Y, то P1 != P2. т.е. P'|S1' + S2' = P'|S3' ще е различно от P''|S1'' + S2'' = P''|S3'', ако S3'!= S3'' т.е. S1' + S2' != S1'' + S2'' или P'!= P''. Тоест трябва да намерим първо възможните S3, което са намерените в точка 2) и за всяко от тях да се види какъв възможен брой цифри може да има префиксът P. Това следва от DS(P) = S2 – DS(S1). И свеждаме нещата до намирането на броя начини да се избере такова дадено P, ако знаем броя на цифрите му и наставката S3, която ще добавим към него така, че P|S3 <= N.

   Как да направим споменатата сметка?

   Нека представим N като N = Pn|Sn, и броя на цифрите на Sn е равен на броя на цифрите на S3. Ако S3 <= Sn трябва да намерим броя P <= Pn, в противен случай търсим броя P < Pn. И в същото време броя единици в P да е дадено число DC.

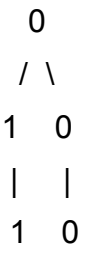
   Нека P = x1….xk, а Pn = y1….yk, като x1, …, x2 са 0 или 1 и y1, ….., yk са 0 или 1. Броя възможности се намира по следния начин:
   - Ако y1 = 1, то x1 = 1 => Result(x1…xk) = Result(x2…xk)

   - Ако y1 = 0, то Result(x1….xk) = Result(x2….xk) + начините да избереш DC единици от к числа (което е комбинация). Алгоритъм продължава рекурсивно за Result(x2…xk).

   Като се сумират резултатите за всички възможни P и S3 се получава C.

Пропуснахме един важен момент, което е точка 3. И този момент е, че при описания алгоритъм ако имаме S3’ и S3’’ и S3’’ = x1..xl|S3’ то може да имаме препокриване на резултатите (това е случеят, когато едното S3 е наставка на другото). Т.е. когато намерим бройката за всички възможни P’ комбинирани със S3’ и всички възможни P’’ комбинирани със S3’’, то някои от P’ и P’’ може да съвпаднат. За да избегнем това ако имаме такава ситуация трябва да търсим само тези P, чиято сума на цифрите дава DS(S3’) – DS(S1’) и се комбинират със S3", но не търсим тези P, чиято сума дава DS(S3’) – DS(S1’) – DS(x1…xl). За да се проверяваме дали за някоя наставка вече имаме наставка, която я включва най-бързото и ефективно решение е да се пази структурата trie. Тя пази едно дърво с всички наставки. Като всеки връх е 0 или 1 и от него може да се отива във връх 0 само ако има наставка която продължва с 0 и аналогично за 1.

Т.е. ако имаме наставкатите 011 и 000 имаме следния trie:



Ако се следват описаните идеи, решението ще работи доста по-бързо, дори по-бързо от поставените ограничения.