АНАЛИЗ НА РЕШЕНИЕТО НА ЗАДАЧА БОНБОНИ

Решение $O(N^2K)$

За всеки вид бонбони можем да преизчислим колко бонбона има от съответния вид от началото на редицата до всеки един индекс. Имайки този масив (partial sums), може лесно да отговорим колко бонбона от всеки вид има в подредицата от i до j. Ако за някой вид имаме масив *sum*, където *sum*[p] е равно на общия брой бонбони от съответния вид, които се съдържат в торбите от 1 до p, тогава *sum*[j] – *sum*[i – 1] ни дава броя на всички бонбони от съответния тип за подредицата от торбички от i до j. Знаейки това, можем да направим пълно изчерпване по броя на различните i и j. Със сложност K може да намерим броя на бонбоните от всеки вид в торбичките от текущата подредица. Този алгоритъм има сложност $O(N^2K)$, което не би довело до голям успех при по-големите тестове.

Ускоряване на решението

Първо, нека направим още няколко наблюдения. За да знаем дали всички условия за подредицата са изпълнени, ние не се нуждаем от самия сбор на бонбоните от всеки вид в нея, а само на тяхната четност. Още при прочитане на данните можем да вземем всички числа, описващи броя на бонбоните от някой вид, по модул 2. Тогава трябва да намерим сумата на числата от i до j по модул 2. Всички числа от съответната редица са 0 и 1. Търсената сума на числата от i до j е равна на XOR (изключващо или) на числата от i до j. До тук, обаче, не печелим нищо. Можем да направим същия масив *sum* за всеки вид, в който *sum*[i] = *sum*[i – 1] ^ a[i]. В a[i] съхраняваме информация за четността на бонбоните (1 – нечетен брой и 0 – четен) от съответния вип в i-тата торбичка. Тогава *sum*[j] ^ *sum*[i – 1] ни дава четността на бонбоните от съответния вид в подредицата от i до j. Имайки това, отново можем да решим задачата с $O(N^2K)$ по същия начин, но използвайки новия масив *sum*.

Нека забележим, че трябва да използваме двумерен масив *sums*[K][N] – за всеки вид имаме *sum*, който е масив с N елемента. Нека разглеждаме *sums* като разменим двата индекса – тогава за всеки индекс на редицата имаме K елемента, които ни показват четността на сборовете от различните видове бонбони в интервала от 1 до текущия индекс на редицата. Вместо да поддържаме по един масив *sum* за всеки вид, можем да обединим първите 32 масива в един. Реално *sum* ни представлява булев масив, чийто стойности могат да бъдат кодирани в един бит на 32-битово число. По този начин, вместо да пазим K на брой масива *sum*, можем да пазим K/32 цели числа, върху които можем да прилагаме операцията XOR и да съхраняваме същата информация, но кодирана в 32-битови числа. Така за всеки индекс i имаме масив от 32-битови числа, които съхраняват информация за четността на общия брой бонбони от всеки вид, намиращи се от първата до i-тата торбичка. Първото 32-битово число съхранява четността на първите 32 вида, като първия бит носи информация за първия вид бонбони, втория бит – за втория вид бит бонбони и т.н., второто 32-битово число носи информация за следващите 32 вида бонбони и т.н. При това положение за всеки $idx1$ – индекс на редицата и всяко $idx2$, от 1 до K/32, имаме *sum2*[$idx1$][$idx2$] = *sum*2[$idx1$–1]^a[$idx1$][$idx2$]. Тук a[$idx1$][$idx2$] съхранява кодирани четностите на видовете, за които отговаря $idx2$ в торбичката с номер $idx1$. Сега нека се върнем на наивното решение. Избираме всички двойки i и j, които да са съответно начало и край на подредицата. Обхождаме всички K/32 числа, които ни носят информация за четността. Ако в *sum*2[p][q] пазим q-тото 32-битово число, което носи информация за

четността на някои 32 вида, в подредицата от 1-вата до *p*-тата торбичка, то *sum*2[*j*][*t*] ^ *sum*2[*i* – 1][*t*] ни дава информация за четността на общия брой бонбони от някои 32 вида в подредицата от *i* до *j*. Менейки *t* от 1 до *K*/32 може да намерим последователно кодирани четностите на общия брой бонбони от всеки вид. Когато всички кодирани числа, които получаваме, бъдат съставени или само от 0 или само от 1 в двоична бройна система, тогава подредицата удовлетворява всички условия. При това решение проверката дали някоя подредица е валидна става със сложност *K*/32, което при зададените ограничения е съществено по-малко.

Забележка: K/32 не винаги е цяло число, но лесно може да се съобрази, че при това положение последното число, в което кодираме, няма да участва с всичките си битове.

Автор: Ивайло Чернев