**Анализ**

*Тагове: взаимно рекурентни редици, моделиране с матрици, бързо степенуване*

Първата подзадача е за 10 точки. Тук реализираме описания процес на машината – на всяка итерация образуваме нов низ като добавяме съответните букви по правилата. За съжаление с този подход големината на низа много бързо нараства (в най-лошия случай около 2 пъти на итерация). Сложността е $O(N\*K\*2^{N}+M)$.

Втората подзадача е за 20 точки. Време е да направим първата важна стъпка към пълното решение. Лесно се забелязва, че е достатъчно просто да следим в текущия низ колко пъти се среща всяка двойка съседни букви. Например, ако имаме правилото αβ → γ и знаем, че има три срещания на αβ, то след итерацията ще трябва да добавим по три срещания на αγ и γβ. Забележете, че понеже добавяме нова буква в средата то нямаме нужда от повече информация. Така в една таблица следим по колко срещания имаме за всяка двойка съседни букви и с едно обхождане на правилата можем да намерим тези бройки за следващата итерация. Допълнително можем да спестим памет като пазим бройките само на текущата и предходната итерация. Накрая трябва просто да вземем предвид за всяка буква наредените двойки, в които участва. Така ще броим всяка вътрешна буква по два пъти за двете двойки съседни букви, от които е част. Това не е вярно само за първата и последната буква, които са същите като в началото. Тях ги броим само веднъж в наредените двойки, които са в началото и в края. Затова трябва да ги преброим изкуствено още веднъж, с което всяко срещане да се брои наистина по два пъти. Един детайл е, че трябва да разделяме на две по модул, но това може да направим с една проверка, както е във всички авторски решения. Сложността е $O(N\*L^{2})$.

Третата подзадача е за 40 точки. Трябва някак да автоматизираме процеса по смятане. Това лесно става с матрици. Нека си представим, че дефинираме редица за всяка наредена двойка, с която редица пазим бройката за дадена итерация. Описаният алгоритъм в предния абзац всъщност задава рекурентно уравнение за смятането на съответния член на всяка редица. Така тези редици са взаимно рекурентни и както е стандартно за бързото им смятане можем да използваме матрица. Всъщност за всеки ред, който съответства на наредена двойка, слагаме единици на тези колони, които съответстват на наредените двойки, които получаваме при прилагането на правилото. Тук един случай, който може да се обърка, е при правило от вида αα → α, защото тогава трябва да сложим 2 на диагоналния елемент на реда. Иначе, за наредени двойки, които не участват в дадено правило трябва да сложим единица на диагоналния елемент, за да не ги изгубим. След като построим матрицата я вдигаме бързо на степен ***N*** и умножаваме с бройките на наредените двойки в началото, за да получим крайните бройки след последната итерация. Накрая извеждаме отговора, както в предния абзац. Предвидената сложност за тази подзадача е $O(log\_{2}N\*L^{6})$.

Последната подзадача е за 30 точки. Трябва да оптимизираме подхода с матриците. Един начин е да съобразим, че е достатъчно да ползваме само въведените преходи, а не да следим бройките на всяка наредена двойка. Всъщност нещата, които се променят са свързани само с наредените двойки, които се получават от правилата. Затова всъщност в матрицата можем да пазим само тях и така размерите стават най-много $2M×2M$. Тук трябва да се внимава, защото след степенуването, умножаваме по началните бройки, а може единствено в първата итерация да се използва правило от рода на αβ → γ, защото просто αβ да не се получава вследствие от правилата. Затова най-лесно е да направим първата итерация ръчно и така да запълним бройките, като съответно после вдигаме матрицата на степен ***N***-1. Крайната сложност е $O(log\_{2}N\*M^{3})$.

Има и други начини да се оптимизира матрицата, като се направи по-внимателен анализ кои двойки трябва да включим, но идеята на задачата беше да не е прекалено пипкава и да е лесната за деня, затова и ограничението по време беше толкова щедро. Интересно е може ли допълнително да се оптимизира решението или да се направи анализ на цикличността на добавянията.

*Идея: Димо Бунов
Автор: Илиян Йорданов*