Анализ на homework

Подзадача 1

Тук ограниченията са достатъчно малки за да направим bruteforce решение на задачата. Програмата просто трябва да умножи всички числа в интервала [2;N] и след това да преброи нулите в края на полученото произведение.

Сложност O(N*Q) - 5 точки.

Подзадача 2

За тази подзадача вече ще трябват малко повече разсъждения. Понеже тук N е от порядъка на 10^5, не можем да умножим всички числа от 2 до N и да броим нулите – резултатът би надхвърлил максималната стойност, побираща се в стандартните числови типове.
Да разгледаме проблема от математическа гледна точка. За да има едно число нула в края на записа си, то това число трябва да се дели на 10 без остатък. А за да се дели едно число на 10 без остатък, то то трябва да се дели едновременно на 2 и на 5 без остатък. За да намерим броя на нулите в края на записа на N!, ще трябва да разложим всяко число от 2 до N на прости множители и да преброим срещанията на 2 и на 5. След това отговорът на задачата ще е по-малкото число от тези две: броят на двойките и броят на петиците. Следва да се спомене, че броят на двойките в разлагането на N! е винаги по-голям от броя на петиците. Това може лесно да се докаже, но тук ще го покажем само с примери: в разлагането на 5! има 3 двойки и една петица, в разлагането на 11! има 7 двойки и две петици, а в разлагането на 25! има 21 двойки и 6 шестици. Т.е. от тук нататък задачата се свежда до това да се намери броя на петиците в разлагането на прости множители на числото N!.
За тази подзадача имаме достатъчно малки ограничения, за да намерим броя на петиците в разлагането на всяко число измежду 2 и N включително. Това може да стане като итерираме през всяко от тези числа и всеки път, в който число може да се раздели на 5, ние го делим и увеличаваме отговора с 1.

Сложност $O(N*\log_5 N*Q)$ - 10 точки

Подзадача 3

Тук главната разлика в ограниченията е по-големият брой заявки, което ни навежда на мисълта, че ще трябва да отговаряме на заявките по-бързо отколкото в миналата подзадача. Тук важно наблюдение е, че едно естествено число има постоянен брой петици в своето разлагане, т.е.

можем отначало да намерим броя на петиците във всяко от числата от 2 до 10^6 и след това да изкараме броя на тези петици от 2 до N. За да извършим това възможно най-ефективно, ще използваме префиксен масив. Ако за всяко число i запишем броят на петиците му в един масив pref[i], тогава задачата се свежда до това да намерим сбора на числата в масива pref от позиция 2 до позиция N. Това е базова задача за префиксен масив и използвайки го ще получим сложност O(1) за отговаряне на всяка заявка. Тъй като обаче, в числата по-малки или равни на 2 няма петици в разлагането им, то pref[2]=0, и програмата просто ще трябва да изведе pref[i].

Сложност $O(N*\log_5 N + Q)$ - 15 точки.

Подзадача 4

В тази подзадача $N <= 10^{18}$, следователно не можем да си позволим сложност включваща $N*\log_5 N$. Тук ще трябва да направим оптимизация в намирането на броя петици в числата по-малки от N. Важно нещо за разглеждане е кои числа съдържат 5 в записа си. Всяко число, което се дели на 5, съдържа поне една петица в разлагането си. Но ние знаем, че числата делящи се на 5 са всяко пето число започвайки от 5. Следователно за да намерим броя на числата, които са по-малки от N и се делят на 5, то ние просто трябва да разделим N на 5. Така е и за всяка друга степен на петицата: за да намерим броя на числата, които са по-малки от N и се делят на 25, то ние трябва само да разделим N на 25 и т.н. В случая, при деление на N на число от типа 5^p ($p>=1$), ние трябва да добавяме резултата към отговора.

Сложност $O(\log_5 N*Q)$ - 21 точки

Подзадача 5

За да пригодим предходната идея за числа от порядъка на 10^{100}, ще трябва да използваме string и да разглеждаме числата като дълги числа. Делението на дълги числа с дълги числа обаче е прекалено сложно и бавно, затова най-добре би било ако делим низовете на едноцифрено число. За целта трябва да използваме следното наблюдение: всяко следващо число е всъщност предното, разделено на 5. Затова вместо всеки път да делим N на 5^p и да прибавяме частното към отговора, можем да записваме частното в N, след като сме го добавили в отговора, и после вместо да делим на 5^{p+1}, да разделим само на 5 (формално казано, $N/(5^{p+1)}=N/(5^p)/5$).

Сложност $O(\log_{10} N*\log_5 N*Q)$ - 22 точки

Подзадача 6

Решението на тази подзадача не се отличава идейно от това на предната, но изисква добре имплементирани алгоритми за операции с низове поради големите размери на числата. Една често използвана оптимизация при събирането на числа, представени чрез низове, е да не добавяме новата цифра в началото на числото, тъй като това е прекалено бавно – може да си представим, че компютърът трябва всеки път да измести всички символи преди да добави новия в началото, което очевидно не е добра идея. Вместо това, ще добавяме новите цифри в края и единствено ще обърнем получения резултат наобратно, когато сме приключили с пресмятането (например с функцията reverse), което ще има обща сложност $\log_{10}N$ (броя цифри на числото).

Сложността отново е $O(\log_{10}N*\log_5N*Q)$, но с по-малка скрита константа, поради което това решение дава допълнителни 10 точки

Подзадача 7

За решаването на тази подзадача е необходимо използването на една хитрост, която да намали размера на числата, с които работим. До момента за да извършим някоя операция с определена цифра, ние вземаме съответния символ и го преобразуваме в число от тип int. Така всеки път обхождаме числото цифра по цифра.

Вместо това, можем да използваме факта, че съществува тип long, който позволява мигновена работа с числа до 19 цифри. Какво ще стане, ако разделим числото на отделни части и вместо да разглеждаме по една цифра всеки път, разглеждаме по 18 едновременно? Именно това е идеята, която ще използваме тук.
Ще разделим числото на части от по 18 цифри, което на практика означава да го преобразуваме от десетична бройна система, в бройна система с база 10^{18}.
За да избегнем неудобства при делението и събирането на такива числа, можем да ги представим чрез vector от long елементи, където всеки елемент показва поредната част от нашето дълго число. Важно е да се отбележи, че за да избягваме добавяне на елементи в началото на вектора, ще пазим числото в обратен ред и накрая единствено ще завъртим отговора наобратно, за да го възвърнем в нормален вид.
Допълнителна специфика има при делението на 5 - трябва да следим в началото на числото (края на вектора) да не остават елементи с размер нула. Ако има такива, то трябва да ги премахнем (например с командата pop_back()). Освен това, делението трябва да извършваме отзад напред, тъй като сме обърнали числото наобратно.

Има нещо за съобразяване и при извеждането на отговора. Да си представим, че сме завършили работата в главния цикъл и вече имаме получен краен отговор. Той все още се намира в голямата бройна система, а на нас ни е необходим изведен в десетична система. Преобразуването става лесно – просто извеждаме числата както са подредени във вектора (след като сме го обърнали, за да бъде подреден нормално). Тънкият момент е, че някои числа ще имат по-малко от 18 цифри, а ние имаме нужда всяко число след първото да има точно 18 цифри . За да оправим този проблем, преди да изведем всяко число от второто нататък, първо ще проверяваме с цикъл колко цифри има и ще добавяме нули пред него, докато го допълним до 18-цифрено.

Благодарение на тази оптимизация смъкване сложността около 18 пъти, с което намаляваме константата на решението значително.

Сложност $O(\log_{10}N*\log_5N*Q)$ - 17 точки

Автори: Преслав Тошев и Калоян Върбанов