АНАЛИЗ НА РЕШЕНИЕТО НА ЗАДАЧА СУМА

Стратегията „от очевидното – към ефективното”

Първото, което ни хрумва, е очевидното решение директно да програмираме условието на задачата. Нещо такова:

```
for (sum=0; A <= B; A++) if (A%P==0 && A%Q==0) sum+=A;
```

Ако приемем, че в интервала [*A*, *B*] има *N* числа за сумиране, то в горния фрагмент за всяко число от интервала ще се извършат 2 деления, 4 логически операции (едната при управлението на цикъла) и едно събиране (пак при управлението на цикъла). Допълнително се извършват *N*-те събирания на числата, отговарящи на условието. При граничния случай (*A*=1, *B*=10^9) това прави 7.10^9+*N* операции, които няма как да се извършат в състезателни условия за разумно време. Следователно – налага се подобряване на решението.

Най-напред се опитваме да проверяваме дали поредното естествено число се дели на най-малкото общо кратно (*LCM*) на *P* и *Q*:

```
for (sum=0; A <= B; A++) if (A%LCM==0) sum+=A;
```

Така директно елиминираме 1 делене и 2 логически операции в сравнение с първоначалния вариант или подобрението е малко над 40% (ако отчетем и *N*-те събирания в *sum*, подобрението е по-малко).

След известно вглеждане установяваме, че в цикъла изобщо няма нужда да се разглеждат всичките числа в интервала [*A*, *B*], а само *N*-те от тях, които се делят на *LCM*, като започваме от *F* – първото по-голямо или равно на *A*, което е кратно на *LCM* и завършим с *L* – най-голямото кратно на *LCM*, което не е по-голямо от *B*:

```
for (sum=0; F <= L; F+=LCM) sum+=F;
```

С което решението се свежда до *N* логически операции и 2*N* събирания. В най-лошия случай (*A*=1, *B*=10^9, *P*=*Q*=*LCM*(*P*,*Q*) = 3) *N* е 333 333 333 или решението изисква около 10^9 операции. Подобрението спрямо началния вариант е над 7 пъти (!), но все още броят операции е твърде голям за състезателна задача.

Решителното подобрение идва с прилагане на метода на Гаус за събиране на поредица числа, всяко от които е по-голямо от предходното с константна стойност (в нашия случай с *LCM*).

Учителят задал на малкия Гаус да събере числата от 1 до 100. Гаус много бързо получил отговора 5050, защото съобразил, че сумата на първото и последното число (1+100) е еднаква със сумата на второто и предпоследното (2+99), е еднаква със сумата на третото и предпредпоследното (3+98) и т.н. От *N* числа (при случката с Гаус *N*=100) могат да се образуват *N*/2 двойки, даващи еднакви суми. Или резултатът е

$$сума = \frac{(първо\_число + последно\_число) \cdot брой\_числа}{2}$$

За избраните от нас означения решението на задачата е $\frac{(F+L)\cdot N}{2}$, за което трябват само 2 умножения и 1 събиране при произволно *N* !!!

Така стигаме до следния алгоритъм за решаване на задачата:

- Определяне на най-малкото общо кратно (*LCM*) на *P* и *Q*:

```
if (P<Q) swap(P,Q);
for (LCM = P; LCM % Q; LCM+=P);
```

- Определяне на *F* и *L*:

```
if (A%LCM==0) F=A;
else F=A+LCM-A%LCM;
L = B-B%LCM;
```

- Определяне на *N*:

```
N = (L-F)/LCM +1;
```

- Определяне на *sum*. Предварително трябва да съобразим, че е възможно в търсения интервал [*A*, *B*] да няма числа кратни на *LCM*. В този случай $L < F$, което е индикатор за $sum = 0$.

```
if (L<F) sum = 0;
else sum = (F+L)*N/2;
```

Поглед отвъд ограниченията

Както се вижда от горния фрагмент - програмирали сме директно формулата на Гаус. За конкретната редакция на условието с така посочените ограничения това върши работа. Какво би се случило с решението ни, обаче, при $B \leq 10\,000\,000\,000$? Можем да очакваме, дори без да сме правили прецизни пресмятания, че резултатът ще стане много голямо число и вероятно ще има проблем с представянето му. Първо се опитваме да използваме за резултата типа `unsigned long long`, защото допуска два пъти по-голямо максимално положително цяло от `long long`. Но даже и резултатът да се побира в `unsigned long long` , програмата ни може да не пресмята вярно заради уловка във формулата на Гаус, която се проявява при големи стойности на (*F*+*L*) и *N*. Възможно е произведението $(F+L)\cdot N$ да е по-голямо от максималната стойност за `unsigned long long` и в този случай се получава препълване (или на програмистки жаргон „превъртане") на типа (към задачата са приложени допълнителни тестови примери, предлагащи точно такава ситуация). За да намалим риска от препълване най-напред ще опитаме да извършим деленето на 2. Това може да се извърши без загуба на точност от целочисленото делене, защото при събиране на цели числа числителят във формулата на Гаус винаги е четно число! ($N\cdot(F+L)=2\cdot sum$, а *sum* е сбор от цели, следователно е цяло) След като произведението $(F+L)\cdot N$ е четно, то поне единият от множителите е четен – намираме кой е, разделяме го на 2 и го умножаваме с другия множител. Ето и подобрения вариант за изчисляване на сумата:

```
if (L<F) sum = 0;
else if (N%2) sum = (F+L)/2*N;
else sum =  N/2*(F+L);
```

Ако ограниченията са още по-големи и резултатът става толкова голям, че не можем да го пресметнем вярно дори с посочените по-горе трикове, се налага да си напишем аритметика за умножаване на дълги числа.

Автор: Евгений Василев