Задача ПЕРИОД

Пояснение към решенията

Решение за 63 т. (`period_63p.cpp`)

В часовете по математика за 5 клас се изучава как се превръща обикновена дроб в десетична. В някои случаи получената десетична дроб може да бъда безкрайно периодична.

В програмата всъщност е реализиран познатият алгоритъм за делене на две цели числа "с лист и молив". За да получим десетичното представяне на дробта `1/a`, започваме с `b=1` и повтаряме стъпките:

1. дописваме 0 отдясно на `b`, т.е. умножаваме с 10: `b=10*b`;
2. цялата част от делението `b/a` дава поредната цифра в резултата;
3. заместваме `b` с остатъка от делението `b=b%a`;
4. повтаряме отново предишните стъпки

Процесът може да продължи до безкрайност, но ние се интересуваме само от онези цифри в резултата получаващи се в точка 2, които се отнасят за позициите с номера от `m` до `n`. За целта използваме брояч `c`, за да отпечатаме само цифрите от необходимите позиции.

```
int c=1;
while(c<=n)
{
   b=10*b;
   if(c>=m) cout << b/a;
   b=b%a;
   c++;
}
```

При големи стойности на `m` и `n` тази програма ще работи бавно.

Решение за 88 **т.** (`period_88p.cpp`)

В програмата се използва ограничението, гарантиращо че дължината на периода е ограничена до 30 000. Реализиран е същият алгоритъм, както в предната програма, но с използването на два допълнителни масива `r[]` и `p[]`. В масива `r[]` се записват поредните остатъци от делението на `b` с `a`. Когато някой такъв остатък се повтори, това означава че след това ще се повтарят и поредните цифри в десетичното представяне. Така откриването на повторение в масива `r[]` служи за намиране дължината на периода. Това се извършва с връщане назад по масива `r[]` чрез:

```
bool ex=false;
for(k=c-2;k>=0;k--)
  if(r[k]==r[c-1]) {ex=true; break;}
if(ex) break;
```

При откриване на период, се преустановява по-нататъшното пресмятане на цифрите от десетичното представяне на дробта.

Дължината на периода се получава равна на стойността `c-k-1` и цифрите от периода се записват в елементите на масива `d[]` с индекси от `k` до `c-2`. В същия масив елементите с индекси от 0 до `k-1` съдържат цифрите на пред-периода.

Цифрите, които трябва да се отпечатат са `cc=n-m+1` на брой. Програмата обработва два случая: когато в отпечатваните цифри са включени цифри от пред-периода (т.е. когато `m<=k`) и когато не са включени. Наличието на период позволява да се отпечатат всички търсени цифри чрез няколкократно повтаряне на отпечатването на цифрите от периода или на част от периода.

Решение за 100 **т.** (`period_100p.cpp`)

В горното решение "зациклянѐто" на периода се установява чрез линейно търсене. За да достигнем 100 точки, ще трябва да ускорим тази част от решението.

Връщайки се към описанието на алгоритъма, което дадохме още при частичното решение, можем да забележим че във всеки момент от алгоритъма стойността на b е най-много $10a$ – след стъпка 3 се взима остатъкът при делението на b с a, а след стъпка 1, този остатъкът се умножава с 10. Така $b \leq 10^7$ винаги и можем вместо линейно търсене да запазваме за всяка стойност, която b е заело, кога е станало това в един броячен масив.

Така за линейно време по дължината на цикъла, можем да го открием и запишем. В предложеното решение записваме цялата информация за периодът на две парчета – във векторите `tail` и `cycle`. `tail` съдържа цифрите зад десетичната запетая, които са записани преди да започне самия период, а `cycle` е точно периодът. Така примерно за $\frac{1}{12} = 0.08(3)$, `tail` (на български "опашката") ще бъде 08, а `cycle` ("цикълът") 3 (технически ще видите други числа във векторите, тези които разделяме на 12, за да получим $0, 8$ и 3).

Може би е добре да се уверим, че дължината на цикъла няма как да стане твърде голяма – ще направим аргумент, сходен на принципа на Дирихле. На всяка стъпка от алгоритъма ни, ние работим с число b и сме казали, че ако два пъти срещнем една и съща стойност на b сме зациклили и можем да спрем работата. Нека видим какъв е най-лошият сценарий, когато това ще се случи възможно най-късно. На всяка стъпка b ще има нова стойност, но тези стойности са ограничение от $10a$, тоест можем да направим най-много $10a$ такива стъпки, избягвайки две еднакви стойности на b. Сега трябва да е ясно, че така алгоритъмът ни ще има сложност $O(a)$.

Последно, ще обърнем и внимание на два момента в имплементацията на решение `period_100p.cpp`. На ред 12 ползваме командата

```
fill(quick_find, quick_find + MAXA + 1, -1);
```

за да запълним броячния ни масив с -1-ци, защото ще индексираме цифрите след десетичната запетая, започвайки от 0. На ред 17 ползваме

```
cycle = vector<int>(it + tail.begin(), tail.end());
```

за да пренесем цифрите, които участват в периода във вектор `cycle`. `it` е първият момент, когато сме срещнали стойността x, с която извършваме делението в момента. `it + tail.begin` ни дава *итератор* до тази позиция в `tail`, а `tail.end()` е итератор за края на този вектор. Подавайки двата итератора на *конструктора* на `vector<int>` той знае да вземе елементите помежду им и да ги копира във вектор `cycle`. На следващия ред ще ги изтрием от вектора `tail`.

Зорница Дженкова и Иван Лупов