Ułamki egipskie

#ułamek_właściwy
#ułamek_prosty
#algorytm_zachłanny

W starożytnym Egipcie stosowano specyficzny sposób zapisu ułamków. Otóż dowolny *ułamek właściwy* (to znaczy taki, którego licznik jest mniejszy od mianownika) rozkładano na sumę *ułamków prostych*, czyli ułamków o licznikach równych 1. Dodatkowo mianowniki tych ułamków musiały być różne. Na przykład ułamek 7/15 możemy zapisać tak (zaraz to sprawdzimy):

$$\frac{7}{15} = \frac{1}{3} + \frac{1}{8} + \frac{1}{120}.$$

W jaki sposób wyznaczyć mianowniki kolejnych ułamków? Najłatwiej zrobić to *metodą zachłanną* (ang. **greedy method**, wymawiaj: *gridy meθod*, z długim *i*), którą cechuje swoista krótkowzroczność: na danym kroku algorytmu staramy się wybrać jak najlepszą możliwość, nie patrząc perspektywicznie na dalsze kroki. W tym przypadku wykonanie kroku polega na znalezieniu jak największego ułamka prostego, który mieści się w rozkładanym ułamku.

Oznaczmy nasz ułamek przez p/q, przykładowo $p = 7$ oraz $q = 15$. Szukamy jak najmniejszego naturalnego m takiego, że zachodzi:

$$\frac{p}{q} \geqslant \frac{1}{m}.$$

Jeśli m będzie jak najmniejsze, wtedy ułamek $1/m$ będzie jak największy, prawda? Wszystkie liczby w powyższej nierówności są dodatnie, zatem możemy zapisać ją tak:

$$m \geqslant \frac{q}{p}.$$

Szukamy minimalnego m naturalnego, a zatem widać, że będzie to sufit z ułamka po prawej stronie:

$$m = \left\lceil \frac{q}{p} \right\rceil.$$

Umiemy już obliczać sufit przy pomocy zwykłego dzielenia całkowitego (w którym domyślnie występuje funkcja podłoga – można przypomnieć sobie zadanie *Wydawanie reszty*):

$$m = \left\lfloor \frac{q+p-1}{p} \right\rfloor.$$

Dla wybranych przez nas wartości ($p = 7$, $q = 15$) mamy:

$$m = \left\lceil \frac{15}{7} \right\rceil = \left\lfloor \frac{15+7-1}{7} \right\rfloor = 3.$$

Teraz ułamek $1/m = 1/3$ należy odjąć od wyjściowego ułamka $p/q = 7/15$ i powtórzyć ten sam krok z wynikiem:

$$\frac{p}{q} - \frac{1}{m} = \frac{pm - q}{qm} = \frac{3 \cdot 7 - 1}{15 \cdot 3} = \frac{6}{45}.$$

Czyli nowe p wynosi 6, a nowe q ma wartość 45. I tu pojawia się drobny problem: wynik jest ułamkiem skracalnym. Dobrze jest taki ułamek skrócić przed przystąpieniem do obliczania kolejnego m, co pozwoli nam na łatwe sformułowanie warunku zakończenia algorytmu. Jeśli bowiem w pewnym momencie licznik p osiągnie wartość 1, to znaczy, że jest to już ostatni ułamek w rozkładzie. Po skróceniu (podzieleniu licznika i mianownika przez największy wspólny dzielnik $\mathbf{gcd}(p, q)$) mamy:

$$\frac{p}{q} = \frac{2}{15}.$$

Implementację funkcji $\mathbf{gcd}(p, q)$ znajdziemy w podrozdziale o *algorytmie Euklidesa* .

Obliczamy teraz m:

$$m = \left\lceil \frac{15}{2} \right\rceil = \left\lfloor \frac{15 + 2 - 1}{2} \right\rfloor = 8,$$

Zatem drugi ułamek ma mianownik 8. Odejmujemy $1/m$ od naszego ułamka:

$$\frac{2}{15} - \frac{1}{8} = \frac{2 \cdot 8 - 15}{15 \cdot 8} = \frac{1}{120},$$

i to jest ostatni ułamek w rozkładzie. Ostatecznie mamy:

$$\frac{7}{15} = \frac{1}{3} + \frac{1}{8} + \frac{1}{120}.$$

Rozkład na ułamki egipskie nie jest jednoznaczny, na przykład łatwo sprawdzić, że poniższy rozkład też jest możliwy:

$$\frac{7}{15} = \frac{1}{4} + \frac{1}{5} + \frac{1}{60}.$$

Algorytmy zachłanne nie zawsze dają optymalne rozwiązania, ale w naszym przypadku żądamy tylko poprawności rozkładu, a ta jest zapewniona.*

Pozostaje nam jeszcze napisać procedurę `egyptian()`, która realizuje opisany powyżej algorytm i wypisze znalezione mianowniki ułamków prostych. Napiszemy ją w formie rekurencyjnej, aby Czytelnik mógł sobie utrwalić ten sposób zapisu algorytmu. Warunkiem zakończenia rekurencji jest wspomniany już warunek $p = 1$. Sama procedura powinna zaczynać się od obliczenia $\mathbf{gcd}(p, q)$ i ewentualnego skrócenia ułamka. Oto proponowany kod:

```
void egyptian(int p, int q)
{
  int d = gcd(p, q);
  p /= d;
  q /= d;
  if(p == 1)
    cout << q << '\n';
```

*Niestety nawet Kocurro nie podejmuje się tak na szybko udowodnić, dlaczego tak jest, może kiedy wróci z urlopu nad polskim morzem.

```
  else
  {
    int m = (q + p - 1)/p;
    cout << m << ' ';
    egyptian(m * p - q, q * m);
  }
}
```

Nazwę procedury wymawiaj: *idżipszn* (akcent na drugie *i*).

Na przykład dla argumentów 7 i 15 na ekranie pojawi się rezultat:

```
3 8 120
```

Proszę zwrócić uwagę na użycie zmiennej pomocniczej *d* przy skracaniu ułamka. Gdybyśmy na przykład zapisali ten fragment kodu w poniższej formie, otrzymalibyśmy nieprawidłowy wynik (dlaczego?):

```
p /= gcd(p, q);
q /= gcd(p, q);
```