Rešitev 3. domače naloge (Fakulteta)

Rešitev za 60%

Rešitev, ki prinese 60% možnih točk, je povsem premočrtna: izračunamo potenco p^k, nato pa v zanki računamo vrednosti $1!$, $2!$, $3!$ ..., dokler rezultat ni deljiv s p^k. Vrednost $n!$ seveda izračunamo na podlagi že izračunane vrednosti $(n-1)!$. Pozorni moramo biti le na podatkovne tipe: ker delamo s števili do 10^{18}, uporabimo tip `long`.

```
import java.util.Scanner;

public class Fakulteta {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        long prastevilo = sc.nextLong();   // p
        long eksponent = sc.nextLong();    // k

        // izračunamo potenco prastevilo^eksponent
        long pot = potenca(prastevilo, eksponent);

        // poiščemo najmanjšo fakulteto, ki je deljiva s potenco
        long fakulteta = 1;
        long n = 1;
        while (fakulteta % pot != 0) {
            n++;
            fakulteta *= n;
        }
        System.out.println(n);
    }

    public static long potenca(long osnova, long eksponent) {
        long rezultat = 1;
        for (long i = 1;  i <= eksponent;  i++) {
            rezultat *= osnova;
        }
        return rezultat;
    }
}
```

Rešitev za 98%

Pri naraščajočih n vrednost $n!$ kaj hitro postane prevelika, da bi jo lahko zapisali v podatkovnem tipu `long`. Podobno se zgodi s potenco p^k. Java sicer premore »raztegljiv« celoštevilski podatkovni tip (razred `BigInteger`), vendar pa je računanje z njim okorno in počasno.

Če malce pomislimo, ugotovimo, da nam pravzaprav ni treba računati niti potence niti fakultete. Da bomo bolj konkretni, vzemimo $p = 3$. (Zlato pravilo umetnosti programiranja:

najprej delamo s konkretnimi podatki, šele zatem s splošnimi.) Brez dejanskega računanja fakultete lahko ugotovimo, da npr. vrednost 20! vsebuje 8 faktorjev 3 in je potemtakem deljiva s 3^8. Preprosto: 20! zapišemo kot $1 \cdot 2 \cdot 3 \cdot \ldots \cdot 20$. Števila 3, 6, 12 in 15 vsebujejo po en faktor 3 (saj so vsa po enkrat deljiva s 3), števili 9 in 18 vsebujeta po dva faktorja 3 (obe sta po dvakrat deljivi s 3), ostala števila pa ne vsebujejo nobenega faktorja 3. Če posplošimo: število faktorjev p v vrednosti $n!$ izračunamo kot vsoto $\sum_{i=1}^{n} f(i, p)$, kjer $f(i, p)$ predstavlja število faktorjev p v številu i. (Ker je p praštevilo, se lahko v vsoti omejimo na vrednosti i, ki so deljive s p; ostale vrednosti i k vsoti namreč ne prispevajo ničesar.)

Program ima v podobno zgradbo kot prvotni: v zanki povečujemo n, dokler vrednost $n!$ ne postane deljiva s p^k oziroma — z drugimi besedami — dokler število faktorjev p v vrednosti $n!$ ne doseže ali preseže k. Kot smo že ugotovili, lahko vrednost n povečujemo s korakom p.

```java
import java.util.Scanner;

public class Fakulteta {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        long prastevilo = sc.nextLong();    // p
        long eksponent = sc.nextLong();     // k

        long n = 0;
        long stFaktorjev = 0;   // število faktorjev p v n!

        while (stFaktorjev < eksponent) {
            n += prastevilo;
            stFaktorjev += steviloFaktorjev(n, prastevilo);
        }
        System.out.println(n);
    }

    // Vrne število faktorjev faktor v številu stevilo.
    private static long steviloFaktorjev(long stevilo, long faktor) {
        long rezultat = 0;
        while (stevilo % faktor == 0) {
            rezultat++;
            stevilo /= faktor;
        }
        return rezultat;
    }
}
```

Rešitev za 100%

V prejšnji rešitvi smo do števila faktorjev p v številu $n!$ prišli s pomočjo zanke. Če je n zelo velik, je takšna rešitev prepočasna. Program bi lahko bil bistveno učinkovitejši, če bi odkrili formulo, ki na podlagi števil n in p takoj vrne vrednost $f(n!, p)$ — število faktorjev p v vrednosti $n!$.

Do nadaljnjega predpostavimo, da velja $p = 3$. Brez težav ugotovimo, da je $f(3!, 3) = 1$ in $f(6!, 3) = 2f(3!, 3) = 2$ (zmnožek števil od 1 do 6 vsebuje skupno dva faktorja 3). Vrednost $f(9!, 3)$ pa ne znaša $3f(3!, 3) = 3$, ampak 4, ker ima število 9 v nasprotju s številoma 3 in 6 dva faktorja 3, ne zgolj enega. Gremo naprej. Vrednost $f(18!, 3)$ znaša $2f(9!, 3) = 8$ (zmnožek $1 \cdot \ldots \cdot 9$ vsebuje štiri faktorje, zmnožek $10 \cdot \ldots \cdot 18$ pa prav tako). Vrednost $f(27!, 3)$ ni enaka $3f(9!, 3) = 12$, ampak 13, ker ima število 27 v nasprotju s številoma 9 in 18 tri faktorje 3, ne zgolj dva. Nadaljujmo. Vrednost $f(54!, 3)$ znaša $2f(27!, 3) = 26$ (zmnožka $1 \cdot \ldots \cdot 27$ in $28 \cdot \ldots \cdot 54$ oba vsebujeta po 13 faktorjev), vrednost $f(81!, 3)$ pa $3f(27!, 3) + 1 = 40$ (število 81 v nasprotju s številoma 27 in 54 vsebuje štiri faktorje 3, ne zgolj tri). Torej:

$$
\begin{aligned}
&f(3^0!, 3) = 0 \\
&f(3^1!, 3) = 1 \\
&f(3^2!, 3) = 3f(3^1!, 3) + 1 = 3 + 1 = 4 \\
&f(3^3!, 3) = 3f(3^2!, 3) + 1 = 3(3 + 1) + 1 = 3^2 + 3 + 1 = 13 \\
&f(3^4!, 3) = 3f(3^3!, 3) + 1 = 3(3^2 + 3 + 1) + 1 = 3^3 + 3^2 + 3 + 1 = 40 \\
&f(3^5!, 3) = 3f(3^4!, 3) + 1 = 3(3^3 + 3^2 + 3 + 1) + 1 = 3^4 + 3^3 + 3^2 + 3 + 1 = 121 \\
&\qquad\qquad \ldots
\end{aligned}
$$

Vrednostim $f(3^r!, 3)$ (torej 0, 1, 4, 13, 40, 121, ...) bomo rekli *mejniki*.

Kako bi izračunali, denimo, vrednost $f(100!, 3)$? Zmnožek $1 \cdot \ldots \cdot 81$ vsebuje $f(3^4!, 3) = 40$ faktorjev 3. Preostane nam še zmnožek $82 \cdot \ldots \cdot 100$, ki je po številu faktorjev enakovreden zmnožku $1 \cdot \ldots \cdot 19$. Zmnožek $1 \cdot \ldots \cdot 9$ vsebuje $f(3^2!, 3) = 4$ faktorje 3, zmnožek $10 \cdot \ldots \cdot 19$ pa je po številu faktorjev enakovreden zmnožku $1 \cdot \ldots \cdot 10$. Zmnožek $1 \cdot \ldots \cdot 9$ vsebuje $f(3^2!, 3) = 4$ faktorje 3, zmnožek 10 pa je enakovreden zmnožku 1, ta pa vsebuje $f(3^0!, 3) = 0$ faktorjev 3. Torej:

$$f(100!, 3) = f(3^4!, 3) + f(3^2!, 3) + f(3^2!, 3) + f(3^0!, 3) = 40 + 4 + 4 + 0 = 48$$

Naš problem je pravzaprav obraten: za podano število faktorjev k moramo določiti najmanjši n z lastnostjo $f(n!, 3) \geqslant k$. Kako to ugotovimo? Preprosto: število k razstavimo na vsoto mejnikov $m_i = f(3^r!, 3)$ in seštejemo vrednosti $n_i = 3^r$, ki pripadajo posameznim mejnikom. Oglejmo si postopek za primer $k = 48$. Največji mejnik, ki ne presega vrednosti 48, je $m_1 = f(3^4!, 3) = 40$. Temu mejniku pripada vrednost $n_1 = 3^4 = 81$. Mejnik odštejemo od začetne vrednosti $k_1 = k$ in dobimo $k_2 = k_1 - m_1 = 8$. Največji mejnik, ki ne presega te vrednosti, je $m_2 = f(3^2!, 3) = 4$. Temu mejniku ustreza vrednosti $n_2 = 3^2 = 9$. Mejnik odštejemo od vrednosti k_2 in dobimo $k_3 = k_2 - f(3^2!, 3) = 8 - 4 = 4$. Največji mejnik, ki ne presega vrednosti k_3, je $m_3 = f(3^2!, 3) = 4$. Mejnik odštejemo od vrednosti k_3 in dobimo $k_4 = k_3 - 4 = 0$. Število 48 smo tako razstavili na vsoto $40 + 4 + 4$. Ustrezna vrednost n znaša $n_1 + n_2 + n_3 = 81 + 9 + 9 = 99$. Najmanjši n, pri katerem je vrednost $n!$ deljiva s 3^{48}, je tako enak 99.

Postopek zlahka posplošimo na poljubno praštevilo p. Mejnike izračunamo po sledečih formulah:

$$
\begin{aligned}
f(p^0!, p) &= 0\\
f(p^1!, p) &= 1\\
f(p^2!, p) &= pf(p^1!, p) + 1 = p + 1 = 4\\
f(p^3!, p) &= pf(p^2!, p) + 1 = p(p+1) + 1 = p^2 + p + 1\\
f(p^4!, p) &= pf(p^3!, p) + 1 = p(p^2+p+1) + 1 = p^3 + p^2 + p + 1\\
f(p^5!, p) &= pf(p^4!, p) + 1 = p(p^3+p^2+p+1) + 1 = p^4 + p^3 + p^2 + p + 1\\
&\dots
\end{aligned}
$$

Na primer, pri $p = 7$ imamo mejnike $f(p^0!, p) = 0$, $f(p^1!, p) = 1$, $f(p^2!, p) = 7 + 1 = 8$, $f(p^3!, p) = 7 \cdot 8 + 1 = 57$, $f(p^4!, p) = 7 \cdot 57 + 1 = 400$ itd. Če iščemo najmanjši n, pri katerem je $n!$ deljiv z, denimo, 7^{1000}, potem število $k = 1000$ razstavimo na vsoto $400 + 400 + 57 + 57 + 57 + 8 + 8 + 8 + 1 + 1 + 1 + 1 + 1$. Tej vsoti ustreza vrednost $n = 7^4 + 7^4 + 7^3 + 7^3 + 7^3 + 7^2 + 7^2 + 7^2 + 7^1 + 7^1 + 7^1 + 7^1 + 7^1 = 6013$.

Naš program lahko sedaj napišemo takole:

```
import java.util.Scanner;

public class Fakulteta {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        long prastevilo = sc.nextLong();
        long eksponent = sc.nextLong();

        long preostaloStFaktorjev = eksponent;
        long n = 0;

        while (preostaloStFaktorjev > 0) {
            long potencaZaMejnik = prastevilo;
            long mejnik = 1;

            while (prastevilo * mejnik + 1 <= preostaloStFaktorjev) {
                mejnik = prastevilo * mejnik + 1;
                potencaZaMejnik *= prastevilo;
            }

            preostaloStFaktorjev -= mejnik;
            n += potencaZaMejnik;

        }
        System.out.println(n);
    }
}
```

V zunanji zanki število k (`eksponent`) razstavimo na vsoto mejnikov. V vsakem obhodu notranje zanke poiščemo največji mejnik, ki ni večji od trenutne vrednosti k (`preostaloStFaktorjev`). Notranja zanka poleg mejnika m (`mejnik`) izračuna tudi število p^r (`potencaZaMejnik`), pri katerem je $f(p^r!, p) = m$. Vrednosti p^r se seštejejo v iskano število n.