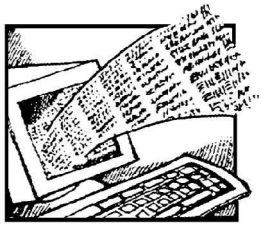

Tietojenkäsittelytehtävien ratkaisuja

Solmun edellisessä numerossa 3/2003 oli matemaattisia tietojenkäsittelytehtäviä, joista toiseen ja kolmanteen esitetään tässä alkuperäiset KöMaLin ratkaisuehdotukset. Ensimmäiseen tehtävään ei KöMaLissa ole esitetty kuin lyhyt ratkaisuperiaate, eikä lukijoilta ole tullut ratkaisuja, joten niitä voi edelleen lähettää Solmun toimitukseen.

2. Binomikertoimet voidaan järjestää tavallisesta Pascalin kolmiosta oheisen kuvan osoittamalla tavalla. Lukuunottamatta kunkin rivin uloimpia alkioita jokainen luku on summa sen suoraan ja vasemmalla yläpuolella olevista luvuista.

					$N = 6$	
1						
1	*1*					
1	*2*	*1*				
1	*3*	*3*	*1*			
1	*4*	*6*	*4*	*1*		
1	*5*	*10*	*10*	*5*	*1*	
1	*6*	*15*	*20*	*15*	*6*	*1*

Tee Excel-taulukko, joka näyttää ensimmäiset $N + 1$ riviä tällä tavalla järjestetystä Pascalin kolmiosta. N:n arvo ($1 \leqslant N \leqslant 20$) syötetään ensimmäisen rivin seitsemänteen sarakkeeseen. Taulukon tulee aina sisältää tasan $N + 1$ riviä.

Ratkaisu. Ratkaisussa otetaan huomioon Pascalin kolmion hyvin tunnettu ominaisuus: jokainen alkio on yläpuolisten naapureidensa summa. Koska taulukkomme on vinossa, summataan suoraan yläpuolella ja vasemmalla yläpuolella olevat naapurit.

Ensimmäisessä sarakkeessa on

```
=IF(ROW()<=\$G\$1+1;1;""),
```

kaikkialla on siis joko ykkösiä tai ei mitään rivin numerosta riippuen.

Muissa soluissa on

```
=IF(ROW()<=\$G\$1+1;C5+D5;""),
```

sillä vain ne rivit, joiden rivinumero on korkeintaan $N + 1$ (N:n arvo on solussa `$G$1`), tulostetaan. Esimerkkikaava on solusta `D6`, ja siinä lasketaan yhteen solujen `D5` ja `C5` arvot.

Taulukon täyttäminen on yksinkertaista, koska Excel muuttaa automaattisesti solujen rivi- ja sarakeviittaukset kaavaa kopioitaessa.

3. Funktio Kertoma(N) $= N!$ kasvaa hyvin nopeasti. Vaikka $5! = 120$, niin jo luvun $10! = 3628800$ tallentamiseen tarvitaan 4-tavuinen kokonaisluku. Tietokone ei voi tallentaa lukua $100!$ 4- tai edes 8-tavuisena kokonaislukuna. Kuitenkin tiedetään, että jokainen luonnollinen luku voidaan hajottaa alkutekijöihin. Esimerkiksi $5! = 2^3 \cdot 3 \cdot 5$ ja $10! = 2^8 \cdot 3^4 \cdot 5^2 \cdot 7$. Tee ohjelma, joka lukee luvun $N (1 \leqslant N \leqslant 10000)$ näppäimistöltä ja tulostaa sen kertoman $N!$ alkutekijähajotelman.

Ratkaisu. Ratkaisemme ongelman antamalla kaksi algoritmiä, jotka tuottavat saman tuloksen.

Ratkaisu 1.

```
Vakio max_al = 5000;

Alkulukutek = Tietue
  lkm  : Luku
  elem : Taulukko( 1 .. max_al ) : Tietue
    alkuluku, eksp: Luku
  Tietueen loppu
Tietueen loppu

Muuttujat
  fakt, akt : Alkulukutek
  i, j, n   : Luku
```

Lue luku. Jos se on 1, tekijöinti on valmis, muussa tapauksessa aloitamme laskennan.

```
Lue: N (1<=N<=10000)
Jos N=1 Niin Tulosta: 1
Muuten
```

Ensimmäinen alkuluku on 2, eksponentilla 1.

```
  fakt.lkm:=1;
  fakt.elem[1].alkuluku:=2;
  fakt.elem[1].eksp:=1;
```

Seuraavaksi laskemme alkulukutekijöinnit luvuille 3:sta N:ään.

```
  Jokaiselle i:=3 .. N
    Alkulukutekijoi(i, akt)
```

Kunkin luvun tekijöinnistä syntyvät eksponentit lisätään jo saatuihin eksponentteihin.

```
    Jokaiselle i:=1 .. akt.lkm
      fakt.elem[j].eksp:=fakt.elem[j].eksp
      + akt.elem[j].eksp;
    Jokaiselle loppu
```

Jokainen uusi alkuluku lisätään listaan. (Tämä tapahtuu vain, jos uusi luku on alkuluku, joten kullakin askeleella alkulukujen määrä voi kasvaa korkeintaan yhdellä.)

```
    Jos fakt.lkm < akt.lkm Niin
      fakt.lkm:=fakt.lkm + 1;
      fakt.elem[fakt.lkm]:=akt.elem[akt.lkm]
    Jos loppu
  Jokaiselle loppu
```

Lopuksi tulostamme kaikki kertoman tekijöinnin elementit.

```
  Jokaiselle i:=1 .. fakt.lkm
    Tulosta: fakt.elem[i].alkuluku,
    fakt.elem[i].eksp;
Jos loppu

Proseduuri Alkulukutekijoi(i : luku; akt :
alkulukutek)
```

Käytämme funktiota `Alkuluku` päättämään, onko luku alkuluku vai ei.

```
  Funktio Alkuluku(j : luku) : looginen
```

Etsimme jakajia luvusta 2 luvun neliöjuureen.

```
    Toista niin kauan kun (k*k<j) Ja
    ((j mod k)>0)
      k:=k+1
    Toista loppu
```

Jos i jakaantuu, se ei ole alkuluku, muussa tapauksessa se on.

```
    Alkuluku:=(k*k>j)
  Funktio loppu
```

2 on ensimmäinen alkuluku, eksponentilla 1.

```
  j:=2;
  akt.lkm:=1;
  akt.elem[j].alkuluku:=2;
  akt.elem[j].eksp:=0;
```

Jatkamme niin kauan kun alkuluku on vähemmän tai yhtäsuuri kuin i.

```
  Toista niin kauan kun j<=i
```

Jos j jakaa i:n, eksponenttia kasvatetaan.

```
    Jos i mod j = 0 Niin
      akt.elem[akt.lkm].eksp:=
      akt.elem[akt.lkm].eksp + 1;
      i:=i div j;
    Muuten
      j:=j+1;
```

Jos jako ei mene tasan, etsimme seuraavan (i:tä pienemmän) alkulukujakajan.

```
Toista niin kauan kun (j<=i) Ja
ei Alkuluku(j)
  j:=j+1;
Toista loppu
```

Jos löydetään alkuluku, saamme uuden alkuluvun j, eksponentilla 0. (Eksponentti kasvatetaan seuraavan kierroksen alussa yhteen).

```
Jos j<=i Niin
  akt.lkm:=akt.lkm + 1;
  akt.elem[akt.lkm].alkuluku:=j;
  akt.elem[akt.lkm].eksp:=0;
Jos loppu
Jos loppu
Toista lopppu
```

Viimeisellä nollaeksponentilla ei ole merkitystä.

```
Jos akt.elem[akt.lkm].eksp = 0 Niin
akt.lkm:=akt.lkm - 1;
Proseduuri loppu
```

Ratkaisu 2. Käytämme Legendren kaavaa seuraavassa muodossa. Jos p on mielivaltainen alkuluku ja N positiivinen kokonaisluku, niin p:n eksponentti $N!$:n alkulukutekijöinnissä saadaan kaavalla

$$\left\lfloor \frac{N}{p} \right\rfloor + \left\lfloor \frac{N}{p^2} \right\rfloor + \left\lfloor \frac{N}{p^3} \right\rfloor + \dots$$

Summa on äärellinen, koska kokonaislukuosat ovat nolla kun $N < p^i$.

Luetaan luku.

```
Lue: N
```

Jos luku on 1, tekijöinti on valmis.

```
Jos N=1 Niin Tulosta: 1
```

Pienin alkuluku on kaksi.

```
p:=2;
```

Jos 2 ei ole suurempi kuin N, 2 eksponentteineen tulostetaan.

```
Jos p<=N Niin Tulosta: p, Eksponentti(p,N);
```

Seuraava alkuluku on 3.

```
p:=3;
```

Niin kauan kun alkuluku ei ole suurempi kuin luku, alkuluku ja sen eksponentti tulostetaan.

```
Toista niin kauan kun p<=N
  Tulosta: p, Eksponentti(p,N);
  Seuraava_alkuluku(p);
Toista loppu
```

Alkuluvun eksponentin laskeminen Legendren kaavalla.

```
Funktio Eksponentti(p,N : luku) : luku
```

Eksponentti on laskettava summa, aluksi nolla. `pp`:ssä on p:n potenssit, aluksi p.

```
  Eksponentti:=0;
  pp:=p;
```

Jatkamme niin kauan kuin p:n potenssi on vähemmän tai yhtäsuuri kuin N.

```
  Toista niin kauan kun pp<=N
```

Eksponenttia kasvatetaan ja seuraava potenssi lasketaan.

```
    Eksponentti:=Eksponentti +
    Pyoristys(n/pp);
    pp:=pp * p;
  Toista loppu
Funktio loppu
```

Lähde: *KöMaL*, Volume 1, Number 1, December 2002, 66–69.
Käännös ja ladonta: ***Timo Metsälä***