Fakultät und Binomialkoeffizient Ac 2013-2016

Die Fakultät (natürliche Zahl):

Die Fakultät „n Fakultät“ ist so definiert: $n! = 1 \cdot 2 \cdot 3 \cdot \ldots \cdot (n-1) \cdot n \quad , \text{ wobei } \quad 0! = 1$

Die rekursive Definition ist: $\textit{Falls } n = 0, \textit{ dann } n! = 1; \textit{ sonst } n! = n \cdot (n-1)!$

Die Ergebnisse für n! wachsen enorm stark, wesentlich stärker als die der Exponentialfunktionen.
So übertrifft bereits 70! = 1,197857166... $\cdot 10^{100}$ das Fassungsvermögen gewöhnlicher Taschenrechner.

JAVA-Methoden(iterativ):

```
public static long nFak(int n) {
    long fak = 1;
    for ( int i = 2; i <= n; i++ )
       fak = fak*i;
    return fak;
}

public static BigInteger nFakBig(int n) {
    // n! = 1*2*3* ... *n
    BigInteger fakBig = BigInteger.ONE;
    for ( int i = 2; i <= n; i++ )
        fakBig = fakBig.multiply(BigInteger.valueOf(i));
    return fakBig;
}
```

Die rekursive Methode ist nachteilig wegen eines möglichen „Stack-Overflows“ !

```
public static long nFakRek(int n) {
    long fak = n;
    if ( n == 0 )
        return 1;
    else nFakRek = n*nFakRek(n-1);
}
```

Anzahl der Ziffern einer Fakultät

Will man die Anzahl der Ziffern einer Fakultät herausfinden, so gibt es u.a. folgende 2 Möglichkeiten, die auf die Logarithmierung zurückgehen:

1) Aus obiger Definition ergibt sich durch Logarithmieren der beiden Seiten:

$$\lg(n!) = \lg(1 \cdot 2 \cdot 3 \cdot \ldots \cdot (n-1) \cdot n) = \lg(1) + \lg(2) + \lg(3) + \ldots + \lg(n-1) + \lg(n)$$

Wegen lg(1) = 0 folgt dann: $$\lg(n!) = \sum_{i=2}^{n} \lg(i)$$

Aus dem Ergebnis für lg(n!) kann man sofort auf die Anzahl der Stellen von n! schließen, denn es gilt:

(*) $$\text{Anzahl der Stellen von n!} = (\text{int})[1 + \sum_{i=2}^{n} \lg(i)]$$ Anm.: int bedeutet "Vorkommaanteil".

2) Stirling-Formel: $$\ln(n!) \approx n \cdot (\ln(n) - 1) + \frac{\ln(2\pi n)}{2} + \frac{1}{12n}$$ (modifizierte Stirling-Formel)

Auch hier wird Formel (*) angewandt, nur dass statt der Summe der Term der Stirling-Formel eingesetzt werden muss !

Während Formel (1) exakt gilt, jedoch für große n aufwendig zu berechnen ist, gilt Formel (2) nur näherungsweise, ist aber auch für große n schnell zu berechnen.

Die folgende Tabelle gibt einen Einblick in die Brauchbarkeit der beiden Formeln unter Anwendung von Formel (*) :

n	**Anzahl der Ziffern n!**	**Summenformel 1)**	**mod. Stirling-Formel 2)**
1	1	1	1
10	7	7	7
100	158	158	158
1 000	2568	2568	2568
10 000	35660	35660	35660
100 000	456574	456574	456574
1 000 000	5565709	5565709	5565709
10 000 000	65657060	65657060	65657060

Beide Formeln sind also sehr brauchbar !

Eine Formel für die Anzahl der Endnullen von n! ist: $$\text{Anzahl der Endnullen von n!} = \sum_{k=1}^{(\text{int})\log_5 n} (\text{int})\frac{n}{5^k}$$

Beispiel: n=125: Anzahl Endnullen von 125! = [125/5] + [125/25] + [125/125] = 25 + 5 + 1 = **31** .

Der Binomialkoeffizient:

Der Binomialkoeffizient „n über k“ ist für n ≥ k so definiert:

$$\boxed{\binom{n}{k} = \frac{n!}{k!\cdot(n-k)!}} \quad \text{mit} \quad \binom{n}{n-k} = \binom{n}{k} \quad \text{und} \quad \binom{n}{0} = \binom{n}{n} = 1 \ .$$

Durch Kürzen von n! gegen (n-k)! lässt sich die Formel noch vereinfachen:

$$\binom{n}{k} = \frac{n!}{k!\cdot(n-k)!} = \frac{n(n-1)(n-2)\cdot\ldots\cdot(n-(k-2))\cdot(n-(k-1))}{1\cdot 2\cdot\ldots\cdot(k-2)\cdot(k-1)\cdot k} =$$

$$\frac{(n-k+1)\cdot(n-k+2)\cdot\ldots\cdot(n-k+(k-1))\cdot(n-k+k)}{1\cdot 2\cdot\ldots\cdot(k-2)\cdot(k-1)\cdot k} = \prod_{i=1}^{k}\frac{n-k+i}{i}$$

Der Vorteil ist, dass hier alle Zwischenergebnisse natürliche Zahlen sind und die Rechnung schnell ist.

Rekursiv gilt: $\boxed{\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k} = \frac{n-(k-1)}{k}\cdot\binom{n}{k-1},\ k>0\ ;\ \binom{n}{0} = 1}$

JAVA-Methode(iterativ):

```
public static long nUeberK(int n,int k) {
   if (n<k) return 0;
   if (n<2*k) k = n-k;
   if (k==1) return n;
   if (k==0) return 1;
   long bin = n-k+1;
   for (int i=2; i<=k; i++)
      bin = bin*(n-k+i)/i;
   return bin;
}

public static BigInteger nUeberKBig(int n, int k) {
   if (n<k) return BigInteger.ZERO;
   if (n<2*k) k = n-k;
   if (k==1) return new BigInteger(String.valueOf(n));
   if (k==0) return BigInteger.ONE;
   BigInteger binBig = BigInteger.valueOf(n-k+1);
   for (int i=2; i<=k; i++)
      binBig = binBig.multiply(BigInteger.valueOf(n-k+i)).divide(BigInteger.valueOf(i));
   return binBig;
}
```

JAVA-Methode(rekursiv):

```
public static long nUeberKRek(int n,int k) {
   if (n<k) return 0;
   if (n<2k) k = n-k;
   if (k=1) return n;
   if (k=0) return 1;
      else return nUeberKRek(n-1,k-1)+nUeberKRek(n-1,k);
}
```

Approximative Berechnungen (STIRLING-Formel):

Nicht immer sind natürliche Zahlen zur Berechnung von Fakultät oder Binomialkoeffizient geeignet.
Selbst der JAVA-Datentyp BigInteger tut sich bei großen Zahlen (15 oder mehr Stellen) schwer.

Als Ausweg können Approximationen mithilfe der Stirling-Formel dienen:

Stirling-Reihenentwicklung für n! :

$$\ln(n!) = n\cdot(\ln(n)-1) + \frac{\ln(2\pi n)}{2} + \frac{1}{12n} - \frac{1}{360n^3} + \frac{1}{1260n^5} - \frac{1}{1680n^7} + \ldots + \frac{(-1)^{k-1}\cdot B_{2k}}{(2k-1)\cdot 2k\cdot n^{2k-1}}$$

B_{2k} ist hierbei die so genannte "**2k-te Bernoullizahl**" (k = 1 gilt für 1/(12n)) .

B_k ist rekursiv definiert wie folgt: $B_0 = 1$; $\boxed{B_k = -\frac{1}{k+1}\cdot\sum_{j=0}^{k-1}\binom{k+1}{j}\cdot B_j} = -\frac{1}{k+1}\cdot\sum_{j=0}^{k-1}\left(\prod_{i=1}^{j}\frac{k+1-j+i}{i}\cdot B_j\right)$

Achtung: $B_{2k+1} = 0$ ab k = 1 !!
Berechnung einiger Bernoullizahlen durch rekursive Java-Methode mit "Bruchklasse":
$B_0 = 1$ $B_1 = -1/2$ $B_2 = 1/6$ $B_3 = 0$ $B_4 = -1/30$ $B_5 = 0$ $B_6 = 1/42$ $B_7 = 0$ $B_8 = -1/30$ $B_9 = 0$
$B_{10} = 5/66$ $B_{11} = 0$ $B_{12} = -691/2730$ $B_{13} = 0$ $B_{14} = 7/6$ $B_{15} = 0$ $B_{16} = -3617/510$ $B_{17} = 0$
$B_{18} = 43867/798$ $B_{19} = 0$ $B_{20} = -174611/330$ $B_{21} = 0$ $B_{22} = 854513/138$

Nun wieder zurück zur Stirling-Reihe:
Bricht man die Reihe nach dem zweiten Summanden ab, so ist der absolute Fehler kleiner als 1/(12n).
Für n >1000 genügt der erste Summand, um den relativen Fehler kleiner als 1% zu halten !
Für sehr große n wird der relative Fehler noch kleiner.

Wie erhält man nun n! ?
Es ist nicht möglich, 10^lg(n!) für sehr große n zu berechnen, weil diese Zahl für normale Rechner zu groß ist.
Man muss stattdessen eine Umformung vornehmen, um von ln(n!) auf das gesuchte n! zu schließen .
Man bildet zunächst: **ln(n!) / ln(10) = lg(n!)** .
Jetzt hat man eine neue Dezimalzahl, deren Vorkommastelle den Exponenten der Zehnerpotenz von n! anzeigt. Der Exponent ist also expo := int(lg(n!)). Man notiert dann einfach den String 10^expo .
Die Mantisse steckt im Nachkommateil von lg(n!), und zwar **berechnet** man: mant = 10^frac(lg(n!)) .
Der gesuchte Gesamtstring setzt sich dann zusammen aus: "mant" und "*10^expo" .

Ergebnis: **n! = 10^frac(lg(n!)) * 10^int(lg(n!))**

Beispiele:

ln(56!) ≈ 56·(ln(56)-1) + ln(112π)/2 + 1/672 – 1/63221760 ≈ 172,3527971
Also ist **56! ≈ 7,109985878*10^74**

ln(500!) ≈ 500·(ln(500)-1) + ln(1000π)/2 + 1/6000 – 1/4,5*10^10 ≈ 2611,330458
Also ist **500! ≈ 1,2201368259*10^1134**

ln(12000!) ≈ 12000·(ln(12000)-1) + ln(24000π)/2 + 1/6,2208*10^14 – 1/3,1352832*10^23 ≈ 100717,5584
Also ist **12000! ≈ 1,2018584067*10^43741**

Ebenso
50000! ≈ 3,3473205095*10^213236

Stirling-Reihenentwicklung für "n über k" :

Unter Verwendung der Stirling-Formel für n!

$$\ln(n!) = n\cdot(\ln(n)-1)+\frac{\ln(2\pi n)}{2}+\frac{1}{12n}-\frac{1}{360n^3}+\frac{1}{1260n^5}-\frac{1}{1680n^7}+\ldots+\frac{(-1)^{k-1}\cdot B_k}{(2k-1)\cdot k\cdot n^{2k-1}}$$

kann man auch eine Approximationsformel für den Binomialkoeffizienten "n über l" herleiten.

Nimmt man obige Reihe bis zum Glied 1/(12n), so ergibt sich

$$\ln\binom{n}{k} = \ln\frac{n!}{k!(n-k)!} = \ln(n!)-\ln(k!)-\ln((n-k)!) =$$

$n\cdot\ln(n) - n + \ln(2\pi n)/2 + 1/(12n)$
$- k\cdot\ln(k) + k - \ln(2\pi k)/2 - 1/(12k)$
$- (n-k)\cdot\ln(n-k) + n-k - \ln(2\pi(n-k))/2 - 1/(12(n-k)) =$

$n\cdot\ln(n) - k\cdot\ln(k) - (n-k)\cdot\ln(n-k) + (\ln(2\pi n) - \ln(2\pi k) - \ln(2\pi(n-k))) / 2 + (1/n - 1/k - 1/(n-k)) / 12 =$
$n\cdot\ln(n) - k\cdot\ln(k) - (n-k)\cdot\ln(n-k) + (\ln(2\pi)+\ln(n) - \ln(2\pi)-\ln(k) - \ln(2\pi)-\ln(n-k)) / 2 +$
$(1/n - 1/k - 1/(n-k)) / 12 =$
$n\cdot\ln(n) - k\cdot\ln(k) - (n-k)\cdot\ln(n-k) + (\ln(n) - \ln(k) - \ln(2\pi) - \ln(n-k)) / 2 + (1/n - 1/k - 1/(n-k)) / 12$

Also erhalten wir die Näherungsformel:

$$\ln\binom{n}{k} \approx \mathbf{(n+0{,}5)\cdot\ln(n) - (k+0{,}5)\cdot\ln(k) - (n-k+0{,}5)\cdot\ln(n-k) - \ln(2\pi)/2 + (1/n - 1/k - 1/(n-k))/12}$$

gültig für n,k > 0 und n > k

Auch hier wird wieder die Beziehung $\lg\binom{n}{k} = \frac{\ln\binom{n}{k}}{\ln(10)}$ verwendet .

Die weitere Vorgehensweise ist analog zu der bei den Fakultäten .

Beispiele:

$\ln\binom{300}{150} \approx 300{,}5\cdot\ln(300) - 150{,}5\cdot\ln(150) - 150{,}5\cdot\ln(150) - \ln(2\pi)/2 + (1/300 - 1/150 - 1/150)/12 \approx$ 204,8656382

$\ln\binom{300}{150}/\ln(10) \approx$ 88,97201622

$\binom{300}{150} \approx$ **9,375970**63***10**^{88}

Ebenso

$\binom{15000}{7000} \approx$ **5,9918950837*10**^{4498} (mit BigInteger berechnet)