Hammingova postupnosť

$$H = 2^i \cdot 3^j \cdot 5^k, \text{ where } i, j, k \geq 0$$

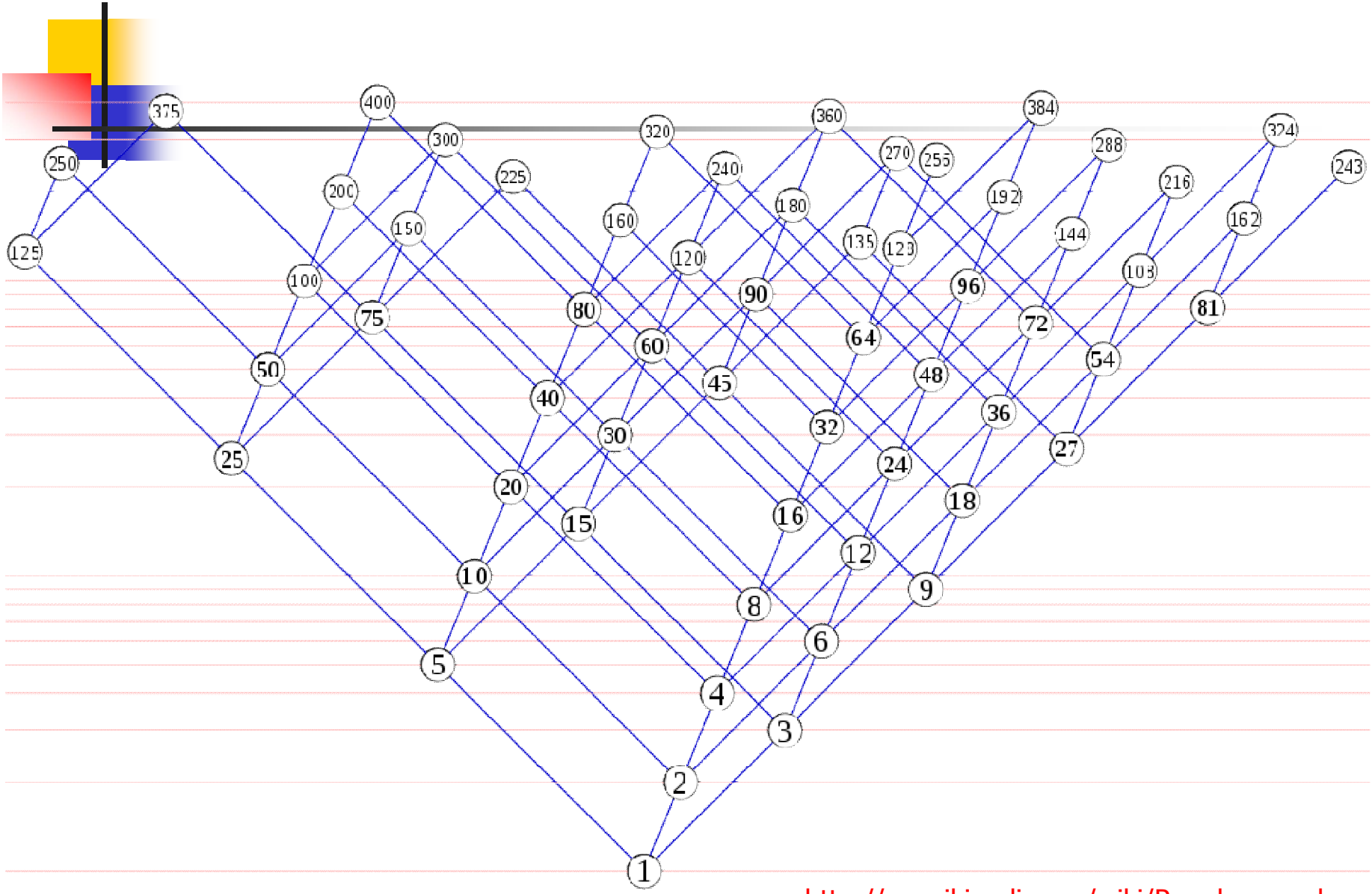
- http://rosettacode.org/wiki/Hamming_numbers

pridali sme *7^m

- zisťovali, či v prvočíselnom rozklade sú len 2,3,5,7 (0.5)
- java-collection väčšinou nepadli na čase (2) (z mnohých napr. MartinKamilM)

```
public static long nty(int n) {
   TreeSet<Long> tree = new TreeSet<Long>();
   tree.add((long) 1);
   long el = 0;
   for (int i = 2; i <= n + 1; i++) {
       el = tree.first(); // vyber najmenší prvok z množiny
       tree.remove(el); // odstráň ho a pridaj el*2, …, *9
       for (int j = 2; j <= 9; j++) tree.add(el * j); }
    return el; }
```

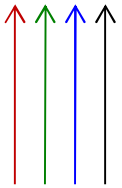
http://dai.fmph.uniba.sk/courses/JAVA/sources/PremiaMnozinova/TreeSet



http://en.wikipedia.org/wiki/Regular_number

Hammingova finta

{ 1 , 2 , 3 , 4 , 5 , 6 ,



** **

23 5 7

Množinová prémia

```
public static long nty(int k) {
 int i2 = 0, i3 = 0, i5 = 0, i7 = 0;
 long A[] = new long[k]; A[0] = 1;
 for (int i = 1; i < k; i++) {
  long min = A[i2] * 2; // hľadáme min(2*A[i2],3*A[i3],5*A[i5],7*A[i7])
  min = (min <= A[i3] * 3) ? min : A[i3] * 3;
  min = (min <= A[i5] * 5) ? min : A[i5] * 5;
  min = (min <= A[i7] * 7) ? min : A[i7] * 7;
  A[i] = min;  // zoberieme min a posunieme index zodpovedajúci mi
  if (min == A[i2] * 2) i2++;
  if (min == A[i3] * 3) i3++;
  if (min == A[i5] * 5) i5++;
  if (min == A[i7] * 7) i7++;
 }
 return A[k - 1];
```

http://dai.fmph.uniba.sk/courses/JAVA/sources/PremiaMnozinova/

Výsledky
(získané vlastnou hlavou, array+BigInteger)

- [0ms.]10-ty clen:10
- [0ms.]100-ty clen:450
- [0ms.]1000-ty clen:385875
- [0ms.]2014-ty clen:7350000
- [15ms.]10000-ty clen:63221760000
- **100.000-ty clen:-29849355667481113 12 ... long overfloat -> BigInteger**
- [94ms.]100.000-ty clen:123093144973968750000
- [1.453ms.]1.000.000-ty clen:4157409948433216829957008507500000000
- [17s.]10.000.000-ty clen:
  1037754506929393275846369194817675812165052808572460991244140544000
- [254s.]100.000.000-ty clen:
  1974029813103265481247271090534939578978639415946231933658845201548773240496359449025476351380348205566406250000000000

Doping: zväčšený heap size na GB
RunConfig/Arguments/VM -Xmx1024m

Výsledky
(upravené riešenie , PriorityQueue+BigInteger)

- [0ms.]10-ty clen:10
- [0ms.]100-ty clen:450
- [16ms.]1.000-ty clen:385875
- [15ms.]2014-ty clen:7350000
- [32ms.]10.000-ty clen:63221760000
- [640ms.]100.000-ty clen:12309314497396875000
- [14 s.]1.000.000-ty clen: 4157409948433216829957008507500000000
- [232s.]10.000.000-ty clen: 1037754506929393275846369194817675812165052808572460991244140544000
- [3.446s.]100.000.000-ty clen: 19740298131032654812472710905349395789786394159462319333658845201548773240496359449025476351380348205566406250000000000

http://rosettacode.org/wiki/Hamming_numbers#Java

Doping: zväčšený heap size na GB
RunConfig/Arguments/VM -Xmx1024m

http://dai.fmph.uniba.sk/courses/JAVA/sources/PremiaMnozinova/MnozinaPriorityQueue