**The problem:**

Every natural number has a factorization into primes, unique up to the order of factors.

Find a prime factor of a given large number n.

**Algorithm:**

1. select a smoothness bound *B*
2. define {\displaystyle M=\prod \_{{\text{primes}}~q\leq B}q^{\lfloor \log \_{q}{n}\rfloor }} (note: explicitly evaluating *M* may not be necessary)
3. randomly pick *a* coprime to *n* (note: we can actually fix *a*, e.g. if *n* is odd, then we can always select *a* = 2, random selection here is not imperative)
4. compute *g* = gcd(*aM* − 1, *n*) (note: exponentiation can be done modulo *n*)
5. if 1 < *g* < *n* then return *g*
6. if *g* = 1 then select a larger *B* and go to step 2 or return failure
7. if *g* = *n* then select a smaller *B* and go to step 2 or return failure

**Data Set:**

predefinedNumbers.Add(BigInteger.Parse("4204458777"));

predefinedNumbers.Add(BigInteger.Parse("8921796065"));

predefinedNumbers.Add(BigInteger.Parse("1241143"));

predefinedNumbers.Add(BigInteger.Parse("6057318955"));

predefinedNumbers.Add(BigInteger.Parse("1281936157"));

predefinedNumbers.Add(BigInteger.Parse("8733973167"));

predefinedNumbers.Add(BigInteger.Parse("1033554461"));

predefinedNumbers.Add(BigInteger.Parse("7986611107"));

predefinedNumbers.Add(BigInteger.Parse("7460145079"));

predefinedNumbers.Add(BigInteger.Parse("9218338053"));

predefinedNumbers.Add(BigInteger.Parse("1275948761"));

Time for classic algoritm: 31780000 ms

Time for pollar p-1 alg: 145510000 ms