uhuge.cm

```
/*
    =================================================================

    Copyright (c) 2012-2016 Seppo Laakko
    http://sourceforge.net/projects/cmajor/

    Distributed under the GNU General Public License, version 3 (GPLv3).
    (See accompanying LICENSE.txt or http://www.gnu.org/licenses/gpl.html
        )

    =================================================================
    */

// Copyright (c) 1994
// Hewlett-Packard Company
// Copyright (c) 1996
// Silicon Graphics Computer Systems, Inc.
// Copyright (c) 2009 Alexander Stepanov and Paul McJones

// 128-bit unsigned integer arithmetics...

using System.Collections;

namespace System
{
    public class uhuge
    {
        public nothrow uhuge(): h(0u), l(0u)
        {
        }
        public nothrow uhuge(ulong l_): h(0u), l(l_)
        {
        }
        public nothrow uhuge(ulong h_, ulong l_): h(h_), l(l_)
        {
        }
        public nothrow uhuge& operator++()
        {
            ++l;
            if (l == 0u)
            {
                ++h;
            }
            return *this;
        }
        public nothrow uhuge& operator--()
        {
            --l;
            if (l == 0xFFFFFFFFFFFFFFFFu)
            {
                --h;
```

```
            }
            return *this;
        }
        public ulong h;
        public ulong l;
    }

    public nothrow inline uhuge operator+(uhuge left, uhuge right)
    {
        uhuge result;
        result.h = left.h + right.h;
        result.l = left.l + right.l;
        if (result.l < left.l)
        {
            result.h = result.h + 1u;
        }
        return result;
    }

    public nothrow inline uhuge operator-(uhuge left, uhuge right)
    {
        uhuge result;
        result.h = left.h - right.h;
        result.l = left.l - right.l;
        if (result.l > left.l)
        {
            result.h = result.h - 1u;
        }
        return result;
    }

    public nothrow uhuge operator<<(uhuge left, uhuge right)
    {
        uhuge zero = 0u;
        uhuge sixtyfour = 64u;
        while (right >= sixtyfour)
        {
            left.h = left.l;
            left.l = 0u;
            right = right - sixtyfour;
        }
        if (right > zero)
        {
            ulong s = right.l;
            ulong lefthl = left.h << s;
            ulong leftlh = left.l >> (64u - s);
            left.h = lefthl + leftlh;
            left.l = left.l << s;
        }
        return left;
    }

    public nothrow uhuge operator>>(uhuge left, uhuge right)
```

```
    {
        uhuge zero = 0u;
        uhuge sixtyfour = 64u;
        while (right >= sixtyfour)
        {
            left.l = left.h;
            left.h = 0u;
            right = right - sixtyfour;
        }
        if (right > zero)
        {
            ulong s = right.l;
            ulong lefthl = left.h << (64u - s);
            ulong leftlh = left.l >> s;
            left.h = left.h >> s;
            left.l = lefthl + leftlh;
        }
        return left;
    }

    public nothrow inline uhuge operator&(uhuge left, uhuge right)
    {
        uhuge result;
        result.h = left.h & right.h;
        result.l = left.l & right.l;
        return result;
    }

    public nothrow inline uhuge operator|(uhuge left, uhuge right)
    {
        uhuge result;
        result.h = left.h | right.h;
        result.l = left.l | right.l;
        return result;
    }

    public nothrow inline uhuge operator^(uhuge left, uhuge right)
    {
        uhuge result;
        result.h = left.h ^ right.h;
        result.l = left.l ^ right.l;
        return result;
    }

    public nothrow inline uhuge operator~(uhuge x)
    {
        uhuge result;
        result.h = ~x.h;
        result.l = ~x.l;
        return result;
    }
```

```
//  The Art of Computer Programming, Volume 2, 4.3.1, Algorithm M (
    multiplication of nonnegative integers), p. 268

    public Pair<uhuge, uhuge> mul(uhuge left, uhuge right)
    {
        ulong[4] u;
        u[3] = left.h >> 32u;
        u[2] = left.h & 0xFFFFFFFFu;
        u[1] = left.l >> 32u;
        u[0] = left.l & 0xFFFFFFFFu;
        ulong[4] v;
        v[3] = right.h >> 32u;
        v[2] = right.h & 0xFFFFFFFFu;
        v[1] = right.l >> 32u;
        v[0] = right.l & 0xFFFFFFFFu;
        ulong[8] w;
        for (int j = 0; j < 4; ++j)
        {
            if (v[j] != 0u)
            {
                uint k = 0u;
                for (int i = 0; i < 4; ++i)
                {
                    ulong t = u[i] * v[j] + w[i + j] + k;
                    w[i + j] = t & 0xFFFFFFFFu;
                    k = cast<uint>(t >> 32u);
                }
                w[j + 4] = k;
            }
        }
        uhuge h((w[7] << 32u) + w[6], (w[5] << 32u) + w[4]);
        uhuge l((w[3] << 32u) + w[2], (w[1] << 32u) + w[0]);
        return MakePair(h, l);
    }

    public uhuge operator*(uhuge left, uhuge right)
    {
        Pair<uhuge, uhuge> hl = mul(left, right);
        return hl.second;
    }

//  The Art of Computer Programming, Volume 2, 4.3.1 exercise 16 (short
    division), p. 625...

    public Pair<uhuge, uint> divmod(uhuge left, uint right)
    {
        uhuge digitBits = 32u;
        ulong b = 0x100000000u;
        uint r = 0u;
        uhuge q = 0u;
        int j = 3;
        while (j >= 0)
        {
```

```
            uhuge shift = uhuge(cast<uint>(j) * 32u);
            uhuge m = left >> shift;
            uint u = cast<uint>(m.l);
            ulong d = r * b + u;
            uint w = cast<uint>(d / right);
            q = (q << digitBits) | uhuge(w);
            r = cast<uint>(d % right);
            --j;
        }
        return MakePair(q, r);
    }

//  The Art of Computer Programming, Volume 2, 4.3.1, Algorithm D (
    division of nonnegative integers), p. 272

    public Pair<uhuge, uhuge> divmod(uhuge left, uhuge right)
    {
        ulong b = 0x100000000u;
        if (right < b)  // see, if we manage using short division
            algorithm...
        {
            Pair<uhuge, uint> p = divmod(left, cast<uint>(right.l));
            return MakePair(p.first, uhuge(p.second));
        }
        ulong m = 0xFFFFFFFFu;
        ulong[8] u;
        u[3] = left.h >> 32u;
        u[2] = left.h & m;
        u[1] = left.l >> 32u;
        u[0] = left.l & m;
        ulong[4] v;
        v[3] = right.h >> 32u;
        v[2] = right.h & m;
        v[1] = right.l >> 32u;
        v[0] = right.l & m;
        ulong[4] q;
//      D1. [Normalize.]
        int n = 4;
        while (v[n - 1] == 0u)
        {
            --n;
        }
        ulong vh = v[n - 1];
        int d = 0;
        while (vh < 0x80000000u)
        {
            ++d;
            vh = vh << 1u;
        }
        ulong t = 0u;
        ulong k = 0u;
        for (int j = 0; j < n + 4; ++j)
        {
```

```
            t = (u[j] << cast<uint>(d)) + k;
            u[j] = t & m;
            k = t >> 32u;
        }
        k = 0u;
        for (int j = 0; j < 4; ++j)
        {
            t = (v[j] << cast<uint>(d)) + k;
            v[j] = t & m;
            k = t >> 32u;
        }
        vh = v[n - 1];
        ulong vmh = 0u;
        if (n > 1)
        {
            vmh = v[n - 2];
        }
//      D2. [Initialize j.]
        for (int j = 3; j >= 0; --j)
        {
//          D3. [Calculate qhat.]
            t = (u[j + n] << 32u) + u[j + n - 1];
            ulong qhat = t / vh;
            ulong rhat = t - vh * qhat;
            if (n > 1)
            {
                while (qhat == b || qhat * vmh > (rhat << 32u) + u[j + n
                    - 2])
                {
                    --qhat;
                    rhat = rhat + vh;
                    if (rhat >= b) break;
                }
            }
//          D4. [Multiply and subtract.]
            k = 0u;
            for (int i = 0; i < n; ++i)
            {
                int ipj = i + j;
                uhuge tc = u[ipj] + uhuge(0xFFFFFFFF00000000u) - k - qhat
                     * v[i];
                t = tc.l;
                u[ipj] = t & m;
                k = m - (t >> 32u);
            }
//          D5. [Test remainder.]
            if (u[j + n] != k)
            {
//              D6. [Add back.]
                --qhat;
                k = 0u;
                for (int i = 0; i < n; ++i)
                {
```

```
                    int ipj = i + j;
                    t = u[ipj] + v[i] + k;
                    u[ipj] = t & m;
                    k = t >> 32u;
                }
            }
            q[j] = qhat;
//      D7. [Loop on j.]
        }
//      D8. [Unnormalize.]
        ulong mask = (ulong(1u) << cast<ulong>(d)) - ulong(1u);
        int j = 3;
        while (j >= n)
        {
            u[j] = 0u;
            --j;
        }
        k = 0u;
        while (j >=  0)
        {
            t = (k << 32u) + u[j];
            u[j] = t >> cast<uint>(d);
            k = t & mask;
            --j;
        }
        uhuge quotient((q[3] << 32u) + q[2], (q[1] << 32u) + q[0]);
        uhuge remainder((u[3] << 32u) + u[2], (u[1] << 32u) + u[0]);
        return MakePair(quotient, remainder);
    }

    public uhuge operator/(uhuge left, uhuge right)
    {
        Pair<uhuge, uhuge> qr = divmod(left, right);
        return qr.first;
    }

    public uhuge operator%(uhuge left, uhuge right)
    {
        Pair<uhuge, uhuge> qr = divmod(left, right);
        return qr.second;
    }

    public nothrow inline bool operator==(uhuge left, uhuge right)
    {
        return left.h == right.h && left.l == right.l;
    }

    public nothrow inline bool operator<(uhuge left, uhuge right)
    {
        if (left.h < right.h)
        {
            return true;
        }
```

```
        else if (left.h > right.h)
        {
            return false;
        }
        else
        {
            return left.l < right.l;
        }
    }
}
```