Profile-guided optimization

Originálny plán pre MathLib.Factorial funkciu bolo z nej urobiť rekurzívnu funkciu. Neskôr sme prišli na to, že tento prístup je neefektívny (v istých prípadoch trval výpočet tak dlho, že naše testy zlyhali kvôli časovej náročnosti), tak sme funkciu prerobili na jednoduchý while loop, ktorý okamžite zvýšil efektivitu a znížil časovú náročnosť.

BEFORE:

```
public static BigInteger Factorial(int n)
{
    // int exponent = num == 0 ? 0 : (int)Math.Floor((Math.Log10(Math.Abs(num))));
    if (n == 1)
    {
        return BigInteger.One;
    }
    return (BigInteger)n * Factorial(n-1);
}
```

AFTER:

```
public static BigInteger Factorial(int n)
{
    if(n == 0)
    {
        throw new ArithmeticException();
    }
    else if (n > 3248)
    {
        throw new OverflowException();
    }

    if(n == 1)
    {
        return BigInteger.One;
    }

    BigInteger result = 1;
    int count = n;
    while(count != 1)
    {
        result *= count;
        count--;
    }

    return result;
}
```