Big Integer and String Processing

Section 5.3, 6.3

Dr. Mayfield and Dr. Lam

Department of Computer Science
James Madison University

Dec 04, 2015

acm International Collegiate Programming Contest

IBM event sponsor

C++ integers

How many values can you represent with b bits?

- Signed: -2^{b-1} to $+2^{b-1}-1$
- Unsigned: 0 to 2^b-1

Type	Min	Max
`char`	-128	127
`unsigned char`	0	255
`short`	-32,768	32,767
`unsigned short`	0	65,535
`int`	-2,147,483,648	2,147,483,647
`unsigned int`	0	4,294,967,295
`long long`	-9,223,372,036,854,775,808	9,223,372,036,854,775,807
`unsigned long long`	0	18,446,744,073,709,551,615

java.math.BigInteger

https://docs.oracle.com/javase/7/docs/api/java/math/BigInteger.html

See also java.math.BigDecimal

When do you need `BigInteger`?

- Numbers with 20 or more digits (e.g., if ever $> 10^{20}$)
- Factorials over 20! (2,432,902,008,176,640,000 is 19 digits)

`BigInteger` also convenient for:

- Number base conversion
- Greatest common divisors
- Modular arithmetic
- Large prime numbers

Getting started

Constructors

- `BigInteger(byte[] val)` (two's-complement)
- `BigInteger(String val)` (string in base 10)
- ...
- `BigInteger.valueOf(long val)` (64-bit integer)

Static constants

- `BigInteger.ONE`
- `BigInteger.TEN`
- `BigInteger.ZERO`

Implementation

- BI objects are immutable
- Sign is stored as an `int`
- Magnitude stored as `int[]`

Math/logic operations

Arithmetic UVa 10523

- `add(BigInteger val)`
- `subtract(BigInteger val)`
- `multiply(BigInteger val)`
- `divide(BigInteger val)`
- `pow(int exponent)`

Comparison

- `compareTo(BigInteger val)`
- `equals(Object x)`
- `max(BigInteger val)`
- `min(BigInteger val)`

Sign

- `abs()`
- `negate()`
- `signum()`

Conversion

- `doubleValue()`
- `floatValue()`
- `intValue()`
- `longValue()`
- `toString()`
- `toByteArray()`

Binary operations

Bitwise

- `and(BigInteger val)`
- `andNot(BigInteger val)`
- `not()`
- `or(BigInteger val)`
- `xor(BigInteger val)`
- `shiftLeft(int n)`
- `shiftRight(int n)`

Size

- `bitCount()`
- `bitLength()`

One at a time

- `clearBit(int n)`
- `flipBit(int n)`
- `getLowestSetBit()`
- `testBit(int n)`
- `setBit(int n)`

See also java.util.BitSet

Bonus features

Number base conversion — UVa 00343

- `BigInteger(String val, int radix)`
- `toString(int radix)`

Greatest common divisor — UVa 10814

- `gcd(BigInteger val)`

Modular arithmetic — UVa 11879

- `divideAndRemainder(BigInteger val)`
- `mod(BigInteger m) // non-negative`
- `modInverse(BigInteger m)`
- `modPow(BigInteger exponent, BigInteger m)` UVa 11287
- `remainder(BigInteger val) // this % val`

Large prime numbers

Probabilistic test

UVa 11287

- `isProbablePrime(int certainty)`
- returns `true`: very likely to prime
- returns `false`: definitely composite

Trade-off: time vs accuracy

- $\mathrm{P(prime)} = 1 - 1/2^{certainty}$
- 10 is usually good enough ($P > 0.999$)

Other methods

- $\mathrm{P(prime)} = 1 - 1/2^{100}$
- `nextProbablePrime()`
- `probablePrime(int bitLength, Random rnd)`

String processing

Cipher/Encode/Encrypt/Decode/Decrypt

- UVa 10878: figure out binary to decimal

Frequency counting

- UVa 902: read char by char, build a map

Input Parsing (Non Recursive)

- UVa 11878: simple pattern recognition

Output Formatting

- UVa 488: use several loops (CS 139 lab)

String Comparison

- UVa 644: check prefixes with brute force