UnsignedBigInt

Yksinkertainen Bignum-kirjasto

Toteutan tietorakenteiden ja algoritmien harjoitustyönä kirjaston, jonka avulla voidaan suorittaa mielivaltaisen suurilla positiivisilla kokonaisluvuilla muutamia yleisimpiä aritmeettisia operaatioita, kuten yhteen-, vähennys-, kerto- ja jakolaskut. Toteutuskieli on Java. Harjoitustyössä käytetään JUnit-yksikkkötestejä. Esimerkkinä kirjaston käytöstä toteutetaan sen avulla yksinkertainen Diffie-Hellman -avaimenvaihto.

Rajapintakuvaus

Konstruktorit

UnsignedBigInt(long)

luo UnsignedBigInt-olion, jonka arvo on annettua long-tyyppistä kokonaislukua vastaava

UnsignedBigInt(String)

luo UnsignedBigInt-olion, jonka arvo on annettua kymmenkantaista merkkijonoesitystä vastaava

Metodit

add(UnsignedBigInt) : UnsignedBigInt

palauttaa tämän ja parametrina annetun luvun summan
aikavaativuus O(n), tilavaativuus O(n)

biggerThan(UnsignedBigInt) : boolean

palauttaa tosi, mikäli tämä luku on parametrina annettua lukua suurempi, muutoin epätosi
aikavaativuus O(n), tilavaativuus O(1)

equals(UnsignedBigInt) : boolean

palauttaa tosi, mikäli tämä ja parametrina annettu luku ovat yhtäsuuret, muutoin epätosi
aikavaativuus O(n), tilavaativuus O(1)

divide(UnsignedBigInt) : UnsignedBigInt

palauttaa tämän luvun jaettuna parametrina annettulla luvulla
aikavaativuus O(n^2), tilavaativuus O(n)

longValue() : long

palauttaa tämän luvun long-tyyppisen kokonaislukuesityksen

mod(UnsignedBigInt) : UnsignedBigInt

palauttaa jakojäännöksen, kun tämä luku jaetaan parametrina annetulla luvulla

aikavaativuus O(n^2), tilavaativuus O(n)

modPow(UnsignedBigInt e, UnsignedBigInt m) : UnsignedBigInt

palauttaa modulaarisen potenssiinkorotuksen (this^e mod m) tuloksen

aikavaativuus O(log e), tilavaativuus O(n)

multiply(UnsignedBigInt) : UnsignedBigInt

palauttaa tämän ja parametrina annetun luvun tulon

aikavaativuus O(n^log_2(3)), tilavaativuus O(n)

pow(UnsignedBigInt) : UnsignedBigInt

palauttaa tämän luvun korotettuna potenssiin parametrina annetulla luvulla

aikavaativuus O(log e), tilavaativuus O(n^2)

smallerThan(UnsignedBigInt) : boolean

palauttaa tosi, mikäli tämä luku on parametrina annettua lukua pienempi, muutoin epätosi

aikavaativuus O(n), tilavaativuus O(1)

subtract(UnsignedBigInt) : UnsignedBigInt

palauttaa tämän ja parametrina annetun luvun erotuksen

aikavaativuus O(n), tilavaativuus O(n)

toString() : String

palauttaa tämän luvun merkkijonoesityksen

Algoritmit

Yhteen- ja vähennyslasku ovat melko triviaaleja tapauksia, joille parhaan tunnetun ratkaisun aikavaativuus on luokkaa O(n). [1] Käytetyt algoritmit muistuttavat hyvin paljon peruskoulussa opittua allekkainlaskua. Luvut käydään läpi alkaen vähiten merkitsevästä bitistä lisäten tai vähentäen ja tarvittaessa "kirjaten muistiin" tai "lainaten". [2: Algorithm 1.1 IntegerAddition].

Pienten lukujen kertolaskuun käytetään naiivia, ns. "basecase" -algoritmia, joka muistuttaa kertolaskua allekkain. [2: Algorithm 1.2 BasecaseMultiply] Algoritmin aikavaativuus on luokkaa O(n^2), mutta se on käytännössä se on pienen vakiokertoimensa ansiosta pienille syötteille kertaluokaltaan parempia algoritmeja nopeampi. Kun kertolaskun tekijät ovat suuria (>= 512 bittiä), käytetään Karatsuban algoritmia, jonka aikavaativuus on O(n^log_2(3)) [2: Algorithm 1.3 KaratsubaMultiply, 3] .

Jakolaskuun käytetty algoritmi muistuttaa jakokulman käyttöä. [4: Integer division (unsigned with remainder)] Algoritmin aikavaativuus on luokkaa O(n^2). [1] Samaa algoritmia käytetään myös jakojäännöksen laskemiseen.

Tavallinen ja moduularinen potenssiinkorotus käyttävät hyväkseen ”exponentiation by squaring” -menetelmää, jolla saavutetaan luokkaa O(log_2(n)) oleva aikavaativuus. [5] Modulaariseen potenssiinkorotukseen käytetään Bruce Schneierin ”Applied Cryptography” -kirjassa kuvattua algoritmia. [6]

[1] Wikipedia: Computational complexity of mathematical operations (http://en.wikipedia.org/wiki/Computational_complexity_of_mathematical_operations)

[2] Richard P. Brent & Paul Zimmermann: Modern Computer Arithmetic v. 0.5.9 (http://maths-people.anu.edu.au/~brent/pd/mca-cup-0.5.9.pdf)

[3] GNU MP 5.1.2 Manual: 15.1.2 Karatsuba Multiplication (http://gmplib.org/manual/Karatsuba-Multiplication.html)

[4] Wikipedia: Division Algorithm (http://en.wikipedia.org/wiki/Division_algorithm)

[5] Wikipedia: Exponentiation by squaring (http://en.wikipedia.org/wiki/Exponentiation_by_squaring)

[6] Bruce Schneier: Applied Cryptography p. 244, 2e, ISBN 0-471-11709-9