SLANKAUS KABLELIO DUOMENŲ TIPO BIBLIOTEKA DARBUI SU DIDELIAIS SKAIČIAIS

Linas Pašviestis, Marijus Bernotas
Šiaulių universitetas, Technologijos fakultetas

Įvadas

Pagrindinės problemos, su kuriomis susiduria šiandien plačiai vartojamas IEEE 754 standartą atitinkantis [1] slankaus kablelio skaičiaus duomenų tipas, yra riboto reikšmių diapazono skaičiavimų rezultatų paklaidos. Minėti trūkumai labai riboja platesnes šio skaičiaus duomenų tipo panaudojimo galimybes. Todėl yra poreikis realizuoti slankaus kablelio duomenų tipą, kuris leistų atlikti aritmetinius veiksmus su dideliais, t. y. išplėsto diapazono, slankaus kablelio skaičiais ir visiškai (arba dalinai) pašalintų atsirandančias paklaidas.

Yra nemažai bibliotekų, įgalinančių atlikti matematinius veiksmus su dideliais skaičiais. Tačiau ne visos jos pritaikytos darbui su slankaus kablelio skaičiais, kai kurios jų yra riboto reikšmių diapazono [6], nepilnai optimizuotais algoritmais, o tai lemia lėtą sistemos darbą. Tai labai aktuali problema kriptografijos, mokslinių skaičiavimų, finansų sektoriuose, kuriuose reikia efektyviai atlikti aritmetinius veiksmus su dideliais skaičiais.

Tyrimo tikslas – sukurti slankaus kablelio duomenų tipo biblioteką, aritmetinių operacijų algoritmus, pritaikytus darbui su dideliais skaičiais.

Uždaviniai: sumodeliuoti slankaus kablelio skaičiaus duomenų tipo struktūrą; realizuoti sumos, atimties, daugybos, dalybos ir kėlimo sveikuoju laipsniu algoritmus; patikrinti sukurtus algoritmus.

Tyrimo metodai: srities modeliavimas, funkcinė analizė, testavimas, loginis apibendrinimas.

Matematiniai algoritmai

Sudėtis – vienas pagrindinių aritmetinių veiksmų. Pats paprasčiausias sumavimo metodas, kai nuo galo imama po vieną skaitmenį ir atliekama sumavimo operacija su pernešimu. Efektyvesnis būdas yra skaidyti skaičių į lygias dalis, pvz., po *n* skaitmenų nuo skaičiaus galo. Suskaidyti skaitmenys talpinami į tinkamo dydžio blokus (sveikojo skaičiaus duomenų tipo kintamuosius), tada nuo galo imama po vieną bloką ir atliekama jų sumavimo operacija su pernešimu.

Slankaus kablelio skaičių sumavimo principas išlieka toks pat kaip ir sveikojo duomenų tipo, tik reikia paskaičiuoti blokų postūmį, atsižvelgiant į eksponenčių reikšmes, ir sumavimą atlikti pagal formulę (1):

$$\sum_{i=1}^{n} C_i = \sum_{i=1}^{n} \left(A_i + B_i \cdot m\right),\ n \in N. \tag{1}$$

Čia: A_i ir B_i – tarpusavyje sulygiuoti blokai,

m – bloko reikšmės postūmio daugiklis (nuo 10^0 iki 10^{n-1}, kur n skaitmenų kiekis bloke),

C_i – dviejų blokų sumos rezultatas.

Kadangi atimtis yra priešingas veiksmas sudėčiai, tai šios aritmetinės operacijos principas sveikiesiems ir slankaus kablelio skaičiams išlieka toks pat, kaip ir sumavimo. Slankaus kablelio skaičiams paskaičiavus blokų postūmį, atimtis atliekama pagal formulę (2):

$$\sum_{i=1}^{n} C_i = \sum_{i=1}^{n} \left(A_i + B_i \cdot \left(-m\right)\right),\ n \in N. \tag{2}$$

Daugyba – binarinė aritmetinė operacija. Natūrinių skaičių aibėje daugybos veiksmas reiškia kelių vienodų dėmenų sumą (3).

$$a \times n = \sum_{i=1}^{n} a = \sum_{i=1}^{n} \underbrace{a + a + \ldots + a}_{n},\ n \in N. \tag{3}$$

Paprasčiausias daugybos metodas – tai narių grupių daugyba ir gautų grupių sumavimas. Tarpusavyje sudaugintų skaičių porų rezultatai yra saugomi laikinuose atminties masyvuose, o vėliau susumuojami. Šis metodas reikalauja n^2 daugybos operacijų, $3 \cdot n^2$ sudėties operacijų. Efektyvesnio daugybos metodo *comba* [2] principas yra panašus į aukščiau minėto, tačiau, kad nereiktų duomenų saugoti laikinuose masyvuose, duomenys yra tiesiogiai saugomi rezultatų masyve. Nauji duomenys yra sumuojami tiesiai į rezultato masyvą.

Paprasčiausias dalybos metodas vadinamas *long division* [3]. Tai standartinę procedūrą turintis algoritmas, tinkamas paprastų ir slankaus kablelio skaičių dalybai. Šis metodas suskaldo dalybos problematiką į daug lengvesnių žingsnių, kuriais galima atlikti dalybos aritmetinę operaciją su dideliais skaičiais.

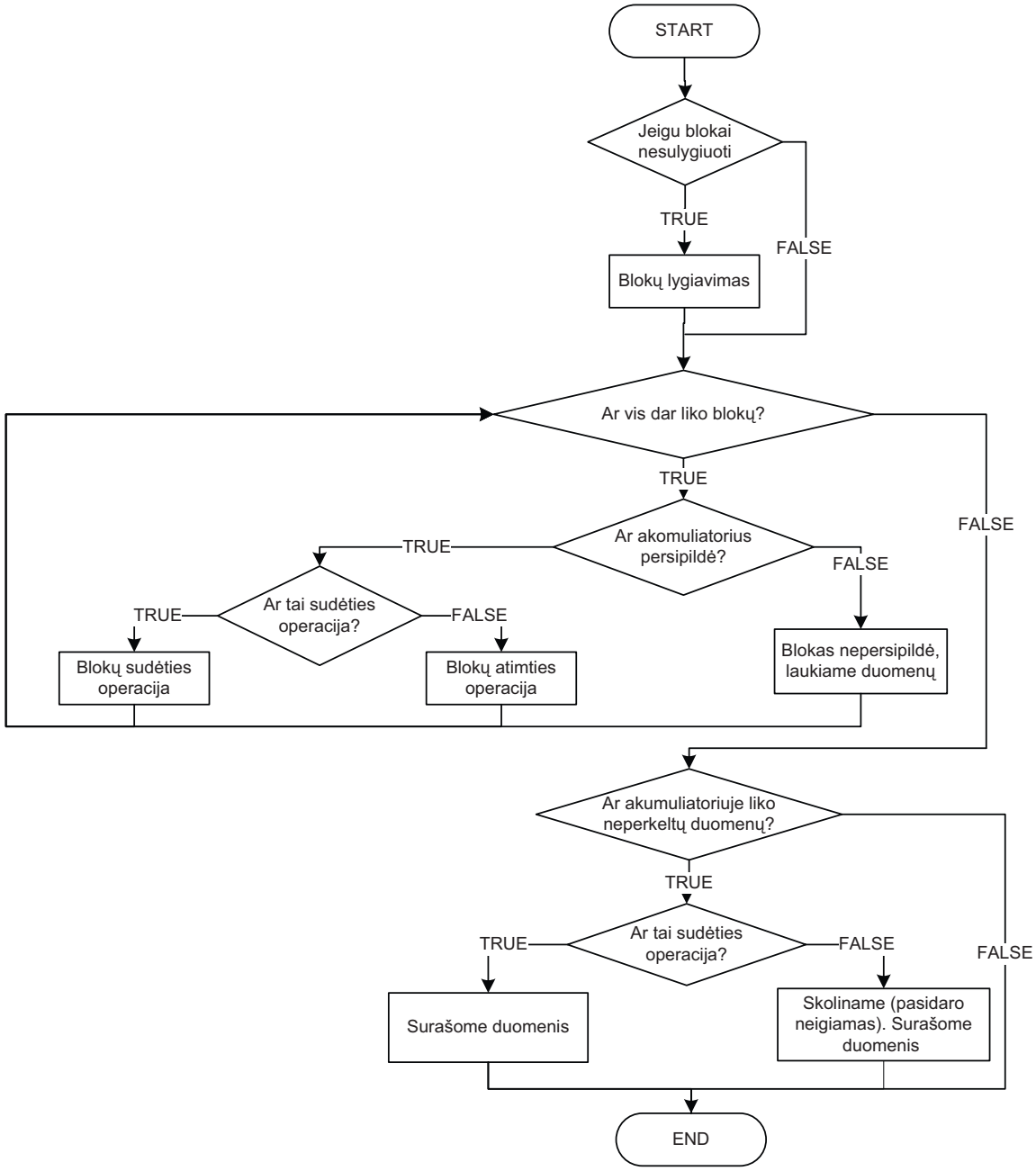
Kėlimas sveikuoju laipsniu [4] – binarinė aritmetinė operacija, reiškianti kartotinę daugybą ar jos apibendrinimus. Realiojo arba kompleksinio skaičiaus kėlimas natūriniu laipsniu *n* reiškia tiek kartų paimto šio skaičiaus sandaugą (4).

$$a^n = \prod_{i=1}^{n} a = \underbrace{a \times a \times \ldots \times a}_{n},\ n \in N. \tag{4}$$

Efektyvesnis kėlimo sveikuoju laipsniu metodas – manipuliuojantis kėlimo daugikliu $C = C \times x$ ir kėlimo kvadratu $C = C \times C$ formulėmis.

Sumatoriaus blokinė diagrama

Diagramoje (1 pav.) pateiktas sumatorius, kurio pagrindinė funkcija – sumuoti blokus. Šis algoritmas vartojamas kaip sumavimo, atimties, daugybos, dalybos algoritmų pagrindas. Prieš atliekant sumavimą, skaičių blokai sulygiuojami atsižvelgiant į abiejų operandų eksponenčių reikšmes. Paskaičiuojamas bloko reikšmės postūmio daugiklis ir tada atliekamas blokų sumavimas (1) arba atimtis (2).



1 pav. Blokinė diagrama, blokų sumatorius

Slankaus kablelio duomenų tipo struktūra

1 lentelėje aprašyta vidinė slankaus kablelio skaičiaus duomenų tipo struktūra, 2 lentelėje – būsenos struktūros elementas.

1 lentelė. ***Vidinė slankaus kablelio skaičiaus duomenų tipo struktūra***

Būsena	Eksponentė	Skaitmenų kiekis	Duomenys
8 bit	32 bit su ženklu	32 bit be ženklo	16 bit blokų masyvas

2 lentelė. ***Vidinės slankaus kablelio būsenos aprašymas***

Būsena	Šešioliktainė reikšmė	Dvejetainė reikšmė	Skaičiaus apibūdinimas
NUM**NEG**	0 x 8	1000	Neigiamas
NUM**INF**	0 x 4	0100	Begalinis
NUM**NAN**	0 x 2	0010	Ne skaičius
NUM**MEM**	0 x 1	0001	Atminties klaida
NUM**NULL**	0 x 0	0000	Tuščias

Klasės detalizavimas

Klasė *Number* realizuoja sudėties, atimties, daugybos, dalybos ir kėlimo sveikuoju laipsniu algoritmus. Klasės atributai ir metodai apibūdinti 3 lentelėje.

3 lentelė. ***Klasės detalizavimas***

Atributai	mFlags – skaičiaus būsena (NEG, INF, NAN, MEM, NULL) mExponent – eksponentės pozicija mDigits – skaitmenų kiekis mUnits – duomenų blokai mPrecision – rezultato tikslumą nusakantis parametras
Metodai	AllocUnits() – darbinės atminties skyrimas ReleaseUnits() – darbinės atminties išlaisvinimas AddUnits() – duomenų blokų sumatorius AddOperation() – sumavimo/atimties algoritmas MultiplyOperation() – daugybos algoritmas DivideOperation() – dalybos algoritmas PowerOperation() – kėlimo sveikuoju laipsniu algoritmas GetInt() – 32bit skaičiaus ištraukimas iš duomenų blokų (jeigu užtenka diapazono) GetDigits() – naudojama skaitmenų kiekiui po aritmetinės operacijos gauti, atmetant tuščius skaičių blokus ExtNumberPart() – gaunama skaičiaus dalis iš duomenų bloko TrimZeros() – kosmetinis skaičiaus sutvarkymas, nulių atmetimas nuo skaičiaus galo Finalize() – koeficientų nustatymas, apvalinimas (priklauso nuo mPrecision) LoadFromString() – skaičiaus įkrovimas iš teksto SaveToString() – skaičiaus konvertavimas į tekstą Clear() – duomenų išvalymas SetPrecision() – rezultato tikslumą nusakančio parametro nustatymas IsMemoryError() – tikrina ar nebuvo atminties klaidos (NUMMEM būsena) operator=() – perdengtas reikšmės iniciavimo operatorius operator+() – perdengtas sumos operatorius operator-() – perdengtas atimties operatorius operator*() – perdengtas daugybos operatorius operator/() – perdengtas dalybos operatorius operator^() – perdengtas kėlimo laipsniu operatorius

Kadangi klasėje vartojami perkrauti aritmetiniai, priskyrimo operatoriai, todėl realizuoto duomenų tipo veikimo principas panašus į standartinio duomenų tipo veikimą.

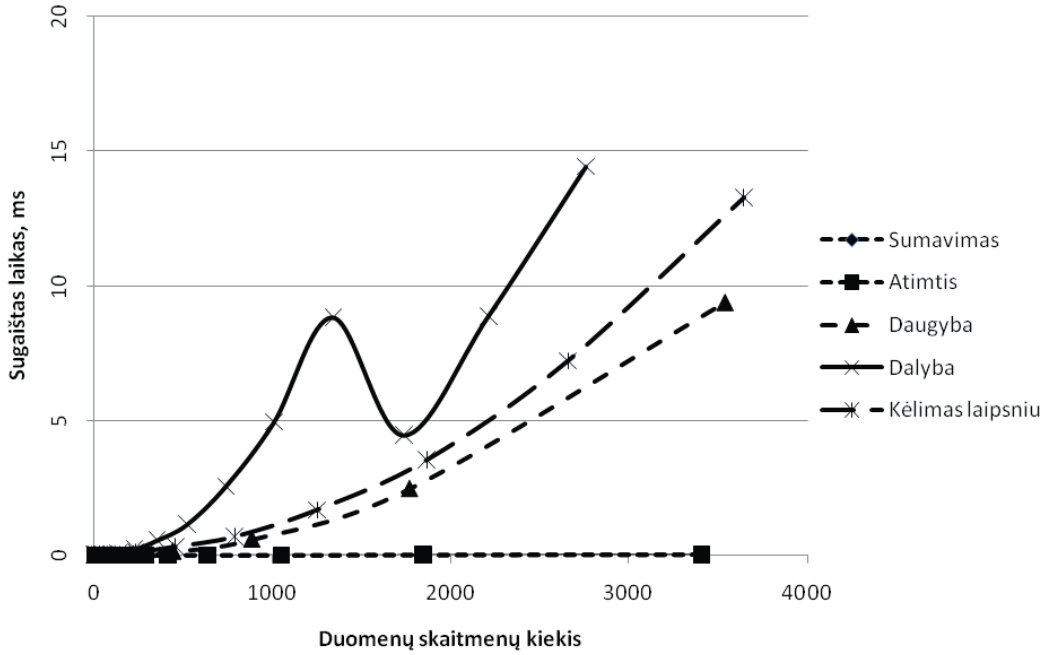
Klasės testavimas

Testavimo metu pašalintos sisteminės ir loginės klaidos:

- Atlikus sumavimo algoritmo testavimą, aptikta ir ištaisyta klaida, kuomet išlaisvinta atmintis būdavo išlaisvinama dar kartą.
- Atlikus atimties algoritmo testavimą, aptiktos kelios klaidos. Pasitaikė atvejų, kai sistema grąžindavo nulį su ženklu.
- Rezultato apvalinimo testavimo metu nustatyta nedidelė paklaida, kuri atsirasdavo dėl to, kad operandai prieš operaciją būdavo apvalinami ir tik tada atliekama aritmetinė operacija. Todėl buvo atjungta funkcija, atsakinga už operandų apvalinimą įkrovimo metu.
- Atlikus daugybos algoritmo testavimą, pastebėtas eksponentės skaitiklio persipildymo atvejis, kadangi eksponentei yra skirta 32 bitai su ženklu, tai jos ribos yra nuo -2147483648 iki 2147483647.
- Atlikus dalybos algoritmo testavimą, neužfiksuota žymesnių klaidų. Tačiau nustatyta, kad, didinant tikslumo parametrą, tiesiogiai didėja algoritmo vykdymo laikas, todėl vidinėje algoritmo struktūroje buvo įdėtas tikslumo parametro ribojimas.

- Atlikus kėlimo sveikuoju laipsniu algoritmo testavimą, aptikta klaida: keliant skaičių neigiamu laipsniu algoritmo pabaigoje šis skaičius turi būti invertuojamas. Šiam procesui atlikti vartojamas dalybos algoritmas. Problema kildavo dėl to, kad minėtame algoritme neapibrėžto skaičiaus nustatymo modulyje būdavo kreipiamasi į jau išlaisvintą atmintį.

Našumo testavimo tikslas – tirti, kaip algoritmų vykdymo laikas priklauso nuo duomenų skaitmenų kiekio, bei šiuos duomenis palyginti tarp skirtingų algoritmų. Atlikus seriją testavimų ir paskaičiavus rezultatų vidurkius, sudarytas algoritmų našumą palyginantis grafikas (2 pav.). Mažiausiai laiko tenka sumavimo ir atimties algoritmams, jų tiesė beveik sutampa. Daugybos algoritmui reikia kur kas daugiau laiko, nes jis, atlikdamas skaičiavimus daug kartų kreipiasi į sumavimo algoritmą. Kėlimo laipsniu algoritmas „sugaišo“ daugiau laiko, negu daugybos algoritmas, nes, atlikdamas matematines operacijas savyje, kreipiasi į daugybos algoritmą. Nors dalybos algoritmas skaičiavimams „naudoja“ atimties algoritmą, jis buvo pats lėčiausias. Atlikus detalesnę analizę, nustatyta tikimybė, jog dalijant rezultatas bus begalinis skaičius, todėl daug laiko algoritmas praranda surašydamas skaičiavimo rezultatus į atmintį. Analizuojant grafiką, išaiškintas neįprastas dalybos algoritmo veikimo pagreitėjimas nuo ~1300 skaitmenų kiekio. Taip atsitinka dėl algoritme įdėtos papildomos apsaugos nuo anksčiau minėtų begalinių skaičių. Pasiekęs atitinkamą ribą, dalybos algoritmas pereina į eksponentės formą. Dalybos metu naujai eksponentės reikšmei apskaičiuoti reikia tik vienos atimties operacijos.



2 pav. Algoritmų darbo greičio palyginimas

Išvados

1. Atlikus pateikto uždavinio sprendimo analizę ir pasirinkus šiam darbui tinkamiausius sprendimus buvo realizuota slankaus kablelio skaičiaus duomenų tipo biblioteka, skirta darbui su dideliais skaičiais. Šiam tipui suprojektuoti ir realizuoti sudėties, atimties, daugybos, dalybos darbui su sveikaisiais ir slankaus kablelio skaičiais bei kėlimo sveikuoju laipsniu aritmetiniai algoritmai. Kadangi yra tikimybė, kad šis projektas gali būti vystomas ateityje, buvo pasirinkta jį realizuoti objektiškai orientuota C++ kalba. Todėl ateityje nebus sunku integruoti naujus aritmetinius algoritmus, kurie veiktų jau realizuotų aritmetinių algoritmų pagrindu.
2. Atliekant testavimą, kiekvienas algoritmas buvo tikrintas su statistiškai patikimu įvairaus ilgio skaičių kiekiu. Testuojant algoritmų našumą, nustatyta, kad algoritmų atliekamų operacijų sugaištas laikas yra tiesiogiai proporcingas operandų, su kuriais atliekama aritmetinė operacija, skaitmenų kiekiui.

Literatūra

1. IEEE 754: Standart for Binary Floating-Point Arithmetic. Prieiga per internetą <http://grouper.ieee.org/groups/754>.
2. How to implement bignum arithmetic <http://chaosradio.ccc.de/23c3_m4v_1658.html>.
3. Long division algorithm. <http://www.mathpath.org/Algor/algor.long.div.htm>.
4. Exponentation <http://en.wikipedia.org/wiki/Exponentiation>.

5. Wolfram Alpha. Computational knowledge engine. Prieiga per internetą: <http://www.wolframalpha.com>.
6. TTMath – a binum library for C++, By Tomasz Sowa. Prieiga per internetą: <http://www.ttmath.org/ttmath>.
7. Floating point. Wikipedia, 2009, prieiga per internetą: <http://en.wikipedia.org/wiki/Floating_point>.
8. Integer (computer science). Wikipedia, 2009, prieiga per internetą: <http://en.wikipedia.org/wiki/Integer_(computer_science)>.
9. The GNU MP Bignum Library. Prieiga per internetą: <http://gmplib.org>.

FLOATING POINT DATA TYPE LIBRARY FOR ARBITRARY PRECISION ARITHMETIC

Linas Pašviestis, Marijus Bernotas

Summary

Searching on the Internet you can find lots of arithmetic libraries suitable for work with large numbers. Unfortunately, some of them lack support for floating point numbers or are limited for arbitrary precision arithmetic. There is big demand for such libraries in cryptography, scientific computation, financial sector, etc. They need effective, accurate arithmetic operations with large numbers.

The main purpose was to make a data type library for arbitrary precision arithmetic, which could operate on integer and floating point numbers. There are no practical limits to precision except the ones implied by the available memory on the machine. This library provides standard arithmetic operations: adding, subtracting, multiplying, dividing and power of integer degree. It allows using large numbers in the same way as the standard data types (integer, float, etc.).

Keywords: algorithms, classes, arbitrary precision.

SLANKAUS KABLELIO DUOMENŲ TIPO BIBLIOTEKA DARBUI SU DIDELIAIS SKAIČIAIS

Linas Pašviestis, Marijus Bernotas

Santrauka

Internete galima aptikti bibliotekų, gebančių atlikti matematinius veiksmus su dideliais skaičiais. Tačiau ne visos bibliotekos pritaikytos darbui su slankaus kablelio skaičiais, kai kurios jų yra riboto reikšmių diapazono, nepilnai optimizuotais algoritmais, o tai lemia lėtą sistemos darbą. Tai labai aktuali problema kriptografijos, mokslinių skaičiavimų, finansų sektoriuose, kuriuose reikalaujama matematinių veiksmų atlikimo su dideliais skaičiais, efektyvumo, tikslumo.

Pagrindinis tikslas – realizuoti duomenų tipo biblioteką, kuri būtų pritaikyta darbui su dideliais sveikojo ir slankiojo kablelio duomenų tipo skaičiais. Ši biblioteka geba atlikti sumavimo, atimties, daugybos, dalybos ir kėlimo sveikuoju laipsniu aritmetines operacijas. Maksimalus skaitmenų ilgis tiesiogiai priklauso nuo kompiuterio laisvos darbinės atminties kiekio.

Prasminiai žodžiai: algoritmai, klasės, sutartinis tikslumas.

Įteikta 2009-09-02