Paleidimas

Paleidimo failai yra /out/production/ReedMullerCodeSimulator/ kataloge. Paleidimas vykdomas start.bat failu. Jeigu reikia perkompiliuoti porogramą galima naudotis pagrindiniame kataloge esančiu compile.bat failu.

Pradiniai tekstai

Pradinius tekstus galima rasti /src/ kataloge, atitinkamai /src/Logic/ ir /src/GUI kataloguose. Klasės „Logic“ pakete atsakingos už veiksmus koduojant/dekoduojant, „GUI“ paketas atsakingas už vartotojo sąsają.

„Logic“ paketo atsakomybės:

- BinaryConverter.java – veiksmai su dvejetainio kūno elementais (konvertavimas iš UTF-8 į dvejetainį tekstą ir atvirkščiai bei dvejetainės simbolių eilutės skaidymas į vektorius).
- Channel.java – realizuotas siuntimas kanalu ir informacijos iškraipymas kanale priklausomai nuo nustatytos klaidos tikimybės.
- Encoder.java – realizuotas kodavimas pirmos eilės Rydo-Miulerio kodu.
- Decoder.java – realizuotas dekodavimas pirmos eilės Rydo-Miulerio kodu užkoduotos informacijos.
- Matrix.java – realizuoti veiksmai su matricomis (daugyba, daugyba moduliu 2, Kronecker produktas bei pagalbiniai veiksmai).
- MatrixFactory.java – atsakingas už konkrečių tipų matricų kūrimą (Rydo-Miulerio pirmos eilės kodo generuojančios matricos, vienetinės matricos, 2 eilės Hadamardo matricos).
- Vector.java – konkretus matricos variantas, kai matrica turi tik 1 stulpelį.

„GUI“ pakete esančios klasės atsakingos už konkrečius langus vartotojo sąsajoje.

- HomeFrame.java – pradinis langas
- VectorFrame.java – vektoriaus siuntimo kanalu sąsajos langas.
- TextFrame.java – teksto siuntimo kanalu sąsajos langas.

Vartotojo sąsaja

Pasileidus programą atsidaro pradinis langas su dviem mygtukais: „Vector“ ir „Text“. Atitinkamai, paspaudus vieną iš jų atsidarys dar vienas langas, kuriame bus galima kanalu siųsti vektorių arba tekstą.

Vektoriaus langas:

1. Įvestis:
    - M – Rydo-Miulerio kodo R(1,m) parametras. Turi būti sveikas skaičius didesnis už 0.
    - Error probability – klaidos tikimybė kanale. Intervale [0,1], skaičius rašyti su tašku (0.1 ne 0,1).
    - Original vector – tekstas, parašytas tik 0 ir 1 simboliais. Teksto ilgis turi būti lygus M + 1 arba dalintis iš M + 1.
2. Išvestis:
    - Encoded vector – užkoduotas vartotojo įvestas vektorius

- Received vector – iš kanalo išėjęs užkoduotas vektorius. Dekodavimas dar nepadarytas. Klaidingos vietos užrašytos raudonu šriftu.
- Decoded vector – dekoduotas iš kanalo išėjęs vektorius.

3. Naudojimas:
    - Suvedami visi 3 įvesties parametrai, laikantis įvesties skiltyje apibrėžtų reikalavimų.
    - Spaudžiamas „Send vector“ mygtukas.
    - Iš kanalo išeina užkoduotas vektorius kuriame yra klaidų, jis rodomas „Received vector“ išvesties lange.
    - Galima pakeisti iš kanalo išėjusį vektorių tiesiog redaguojant tekstą esantį „Received vector“ lauke.
    - Norint dekoduoti, spaudžiamas „Decode vector“ mygtukas. Tekstas dekodavimui bus imamas iš „Received vector“ lauko todėl jis būtinai turi būti netuščias, turėti tik 0 arba 1 simbolius ir būti tokio ilgio kokio jį gražino kanalas.
    - Dekoduotas vektorius atsiras „Decoded vector“ lauke.
    - Galima keisti įvesties parametrus (reikalavimų ribose) ir kartoti siuntimą su naujais arba tais pačiais duomenimis.

Teksto langas:

1. Įvestis:
    - M – Rydo-Miulerio kodo R(1,m) parametras. Turi būti sveikas skaičius didesnis už 0.
    - Error probability – klaidos tikimybė kanale. Intervale [0,1], skaičius rašyti su tašku (0.1 ne 0,1).
    - Original text – tekstas kurį norima siųsti kanalu. Galima naudoti visus UTF-8 simbolius, įskaitant tarpus ir naujas eilutes.
2. Išvestis:
    - Received non R(1,m) encoded text – iš kanalo išėjęs tekstas, kurio siuntimui kanalu nebuvo naudojamas Rydo-Miulerio kodas.
    - Received R(1, m) encoded text – iš kanalo išėjęs tekstas, kurio siuntimui buvo naudojamas Rydo-Miulerio kodas.
3. Naudojimas:
    - Suvedami visi 3 įvesties parametrai, laikantis įvesties skiltyje apibrėžtų reikalavimų.
    - Spaudžiamas „Send text“ mygtukas.
    - Atitinkamuose išvesties laukuose pasirodo rezultatai: tekstas siųstas nenaudojant kodavimo ir naudojant kodavimą.
    - Galima keisti įvesties parametrus (reikalavimų ribose) ir kartoti siuntimą su naujais arba tais pačiais duomenimis.

Programiniai sprendimai

1. Teksto paruošimas kodavimui ir kontrolinių duomenų apsaugojimas

Prieš kodavimą tekstas yra konvertuojamas į dvejetainį pavidalą naudojant BinaryConverter klasės, getBinaryString metodą. Kadangi kiekvienas simbolis konvertuojamas į 8 bitų ilgio

dvejetaine reprezentaciją, gali iškilti problemų skaidant eilutę į vektorius, iš kurių ilgių nesidalina dvejetainės simbolių eilutės simbolių skaičius.
Problema sprendžiama prie dvejetainės simbolių eilutės prirašant nulius pradžioje, kad simbolių skaičius dalintųsi iš vektoriaus ilgio. Reikalingų prirašyti 0 skaičius randamas pagal lygybę: „diff“ = (M + 1) – (teksto_ilgis mod (M + 1)). Papildytas tekstas siunčiamas kanalu, bet prieš dekodavimą, kviečiama funkcija, kuri pirmus „diff“ skaičių pakeičia į 0 nepriklausomai nuo jų turinio, taip ištaisant klaidas kurias kontrolinėje informacijoje galėjo padaryti kanalas. Ištaisytas tekstas toliau siunčiamas dekodavimui.

2. Teksto skaidymas vektoriais

Naudojama BinaryConverter klasė. Vartotojo įvestas tekstas paduodamas į getBinaryString metodą, kuris gražina atlieka tipų konvertavimus tarp char ir int, užtikrina kad kiekvienas simbolis būtų reprezentuojamas lygiai 8 dvejetainiais simboliais (tai reikalinga vėliau norint dvejetaine eilutę paversti atgal į UTF-8 simbolių seką). Gautas tekstas siunčiamas į Encoder klasės encode metodą, kuris naudoja tą pati BinaryConverter kad tekstą išskaidytų į Vector tipo objektų masyvą. Toliau kodavimas vykdomas su gautais objektais.

3. Vektorių siuntimas kanalu

Sukuriamas Channel klasės objektas į konstruktorių paduodant klaidos tikimybę. Turint objektą, jam kviečiamas sendData metodas, į kurį paduodamas dvejetainė simbolių eilutė, reprezentuojanti siunčiamus duomenis. Channel objektas kiekvienam simboliui generuoja skaičių nuo 1 iki 100,000. Jeigu gautas skaičius mažesnis už tikimybė * 100,000, simbolis yra pakeičiamas (1 -> 0, 0 -> 1). Gautas rezultatas gražinamas kaip String kintamasis.

4. Matricos ir vektoriai

Matricos ir vektoriai reprezentuojami Matrix ir Vector klasių objektais. Kadangi vektorius iš esmės yra matrica kurios dimensijos yra 1 x N, Vector klasė paveldi metodus iš Matrix, skirtumas tik tas kad Vector klasė reikalauja tik 1 dimensiijos parametron konstruktoriuje, bet taip pat reikalauja reikšmių masyvo, skirtingai negu Matrix klasė, kuri inicializuojama nuliais.

5. Matricų daugybos operacijos

Realizuojamos į pagalba pasitelkiant metodus, gražinančius tam tikrą matricos eilutę ir stulpelį.
Daugyba moduliu du daroma paprasčiausiai pritaikius =% operatorių su operandu 2 visoms matrices reikšmėms.

6. Kronecker produkto operacija

Realizuotas naudojant pagalbinę operaciją getPartialMatrices. Ši operacija gražina masyvą matricų iš kurio sudėliojamas galutinis operacijos rezultatas. Kiekviena iš šių matricų gaunama realizuojant matricos daugybą iš skaliaro (ši operacija yra privati ir naudojama tik kronecker produkto radimo operacijoje). Gautos matricos sudedamos į rezultato matricą naudojant insert metodą.

7. Kodavimas ir generuojanti matrica

Kodavimas vykdomas Encoder klasės, į ją padavus M parametrą per konstruktorių ir vėliau kviečiant encode metodą, į jį paduodant duomenis. Encoder klasė naudoja MatrixFactory generuojančios matricos gavimui. MatrixFactory gauna parametrą m į metodą

createGeneratorMatrix ir generuoja matricą naudodamas privačius metodus getFilledRow (sugeneruoja skaičių sekas, kur 1 ir 0 santykis eilutėje yra 1 : 1) ir getMaxConsecutive (gražina skaičių pasakantį kiek to paties simbolio rašyti iš eilės). Iš MatrixFactory gautą generuojančią matricą Encoder klasė naudoja koduodama duomenis paprasčiausiai daugina kiekviena vektorių iš generuojančios matricos pasitelkiant Matrix klasės multiplyMod2 metodą.

8. Dekodavimas

Realizuotas panašiai kaip ir kodavimas, iš tos pusės kad į konstruktorių paduodamas tas pats M parametras ir vėliau kviečiamas decode metodas į jį paduodant gautus duomenis. Kuriant Decoder objektą naudojamas createHmiMatrices metodas kuris kuria matricas aprašytas [HLL91, §3.8–3.9, p. 89–95] dekodavimo sekcijoje (94 psl.). Toliau 3.9.4. algoritme minėti žingsniai atitinkamai realizuojami metodais alterVectors, computeWMVectors (realizuotas naudojant rekursiją) , getLargestAbsoluteValuePosition. Galutiniai vektoriai gaunami sujungiant į eilutę “1” arba “0” su getReverseBinary rezultatu (priklausomai nuo to ar didžiausia vektoriaus koordinatė buvo daugiau ar mažiau už 0).

Užduoties atlikimas

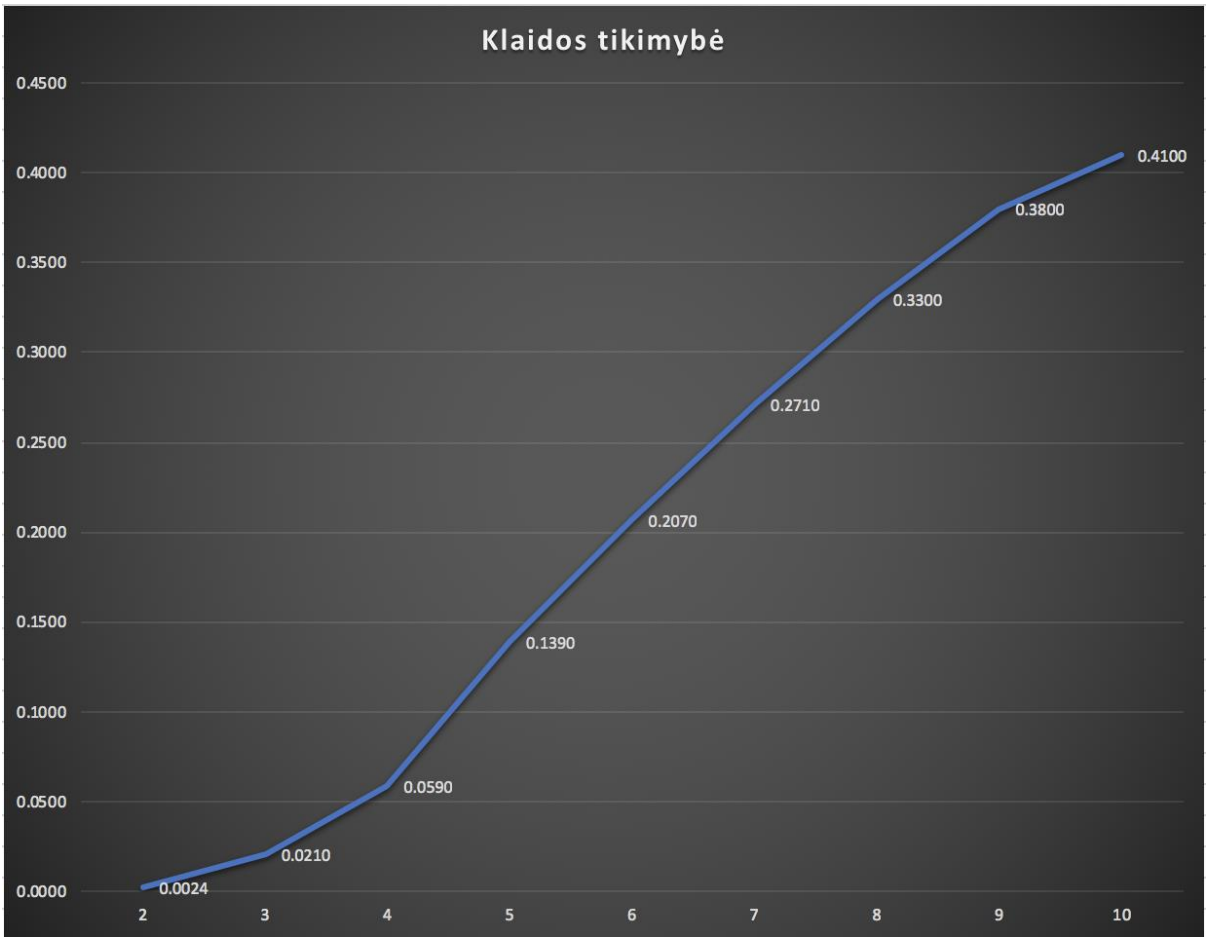
Nebuvo įgyvendintas 3 scenarijus (paveikslėlio siuntimas kanalu).

Atlikti eksperimentai

Buvo tikrintas kodo gebėjimas taisyti teisingai ištaisyti duomenis esant tam tikrai klaidos tikimybei. Testai buvo vykdyti kartojant kodavima, siuntimą kanalu ir dekodavimą daug kartų iš eilės su tuo pačiu m ir klaidos tikimybe. Jei būdavo teisingai dekoduojama bent 99 proc. vektorių, buvo didinama klaidos tikimybė iki tol, kol teisingai buvo dekoduojama mažiau nei 99 procentai vektorių. Galutinė klaidos tikimybė buvo laikoma kodo su parametrais R(1, m) maksimalia klaidos tikimybe. Testas buvo kartotas su m nuo 2 iki 10 įskaitinai. Didinant m buvo pradėtas mažinti iteracijų skaičius bei klaidos tikimybės pradėtos tikrinti didesniais intervalais, kadangi su didesniais m testai trukdavo labai ilgai.

Kodą iš kurio buvo gauti eksperimentų rezultatai galima rasti failuose /src/Main.java (užkomentuotas main() funkcijoje bei funkcijos po main()) ir /src/Tests/AccuracyTest.java.

Grafikas 1



Naudota literatūra

Paskaitų konspektai http://klevas.mif.vu.lt/~skersys/18r/ktkt/KTKT.pdf
A5 užduočiai rekomenduota literatūra [HLL91, §3.8–3.9, p. 89–95]