VILNIAUS UNIVERSITETAS

MATEMATIKOS IR INFORMATIKOS FAKULTETAS

PROGRAMŲ SISTEMŲ STUDIJŲ PROGRAMA

Kodavimo teorija

Laboratorinio darbo ataskaita

Užduotis A5

Darbą atliko:

Robertas Buračevskis

Vilnius
2024

Turinys

1. Atliktos užduotys

Realizuotos visos užduoties dalys:

- Užkodavimas
- Kanalo simuliacija
- Dekodavimas
- Vektoriaus siuntimas
- Teksto siuntimas
- Paveikslėlio siuntimas

2. Panaudotos bibliotekos

Programos realizavimui buvo panaudotos šios bibliotekos:

- javax.swing.* - naudota vartotojo sąsajos (UI) kūrimui su „Swing“.
- javax.imageio.ImageIO – naudota paveiksėlių duomenims skaityti ir rašyti.
- java.awt.* - naudota įvairioms UI komponentų funkcijoms.
- java.util.* - naudota sąrašų kūrimui ir atsitiktinių skaičių generavimui.
- java.io.File – naudota darbui su failais ir duomenų srautais.
- java.jfree.chart.* - naudota diagramos braižymui, padedant vizualizuoti eksperimento rezultatus.

3. Programos paleidimas

Programą galima paleisti, nuėjus į ***„ \A5\target“*** ir dukart paspaudus ant **A5-1.0.jar** failo, arba terminale nuėjus į ***„\A5\target“*** ir parašyti ***java -jar A5-1.0.jar***. Programos paleidimui reikalinga Java 23 versija.

4. Programos struktūra

- **com.kt.channel.ChannelSimulator** – klasė, simuliuojanti kanalą, kuriame iškraipomi bitai su tam tikra klaidų tikimybe.
- **com.kt.decoding.Decoder** – klasė, kurioje realizuotas dekodavimo algoritmas, naudojantis greitąją Hadamardo transformaciją.
- **com.kt.encoder.Encoder** – klasė, kurioje realizuotas užkodavimas, naudojant algoritmą, pagrįstą Rydo-Miulerio (Reed-Muller) kodu RM(1,m).
- **com.kt.helpers.ChartHelper** – *klasė, kuri buvo naudojama eksperimentui (grafo atvaizdavimas)*
- **com.kt.helpers.HadamardMatrixGenerator** – pagalbinė klasė dekodavimui, kuri sugeneruoja

Hadamardo matricas.

- **com.kt.helpers.ImageProcessingHelper** – pagalbinė klasė, padedanti konvertuoti paveiksėlį į vektorių sąrašą arba atvirkščiai – vektorių sąrašą į paveikslėlį.
- **com.kt.helpers.MatrixGenerator** – pagalbinė klasė užkodavimui, sukurianti generuojančią Rydo-Miulerio matricą RM(1,m).
- **com.kt.helpers.TextTransformHelper** – pagalbinė klasė, padedanti konvertuoti tekstą į vektorių sąrašą arba atvirkščiai – vektorių sąrašą į paveikslėlį.
- **com.kt.helpers.TransmissionResults** – *klasė, kuri buvo naudojama eksperimentui (kaupiami duomenys)*
- **com.kt.helpers.ValidationHelper** – pagalbinė klasė, skirta validuoti įvestas reikšmes.
- **com.kt.ui.MainUi** – pagrindinė klasė kurioje sukuriama ir kurios dėka paleidžiama grafinė vartotojo sąsaja, skirta žinutėms koduoti, siųsti per kanalą ir dekoduoti naudojant Rydo-Miulerio bei greitąsias Hadamardo transformacijas. Sąsaja turi atskiras korteles skirtingiems scenarijams.
- **com.kt.Main** – *Perteklinė klasė, ji nėra naudojama, tačiau programos kūrimo pradžioje buvo įgyvendintas pirmasis scenarijus su šiek tiek detalesniais konsolės išvedimais (log'inimais).*

5. Vartotojo sąsajos aprašymas

Pasileidę programą, mus pasitinka vartotojo sąsaja, kurioje galime rinktis vieną iš 3 scenarijų (vektoriaus siuntimas, teksto siuntimas bei paveikslėlio siuntimas atitinkamai). Kiekviena kortelė atlieka kodavimo, persiuntimo ir dekodavimo veiksmus su skirtingų tipų duomenimis. Norint grįžti į pagrindinį meniu, apačioje yra mygtukas **Back to Main Menu**. Taip pat svarbu paminėti, jog kiekvienam scenarijui yra priskiriami skirtingi parametrai (m reikšmės bei klaidos tikimybė). Tai yra padaryta patogumo sumėtimais, jog nereikėtų grįžti ir kaskart keisti atitinkamas reikšmes.

5.1 Pirmasis scenarijus (vektorių siuntimas)

Šioje kortelėje vartotojo yra prašoma įvesti m reikšmę kuris yra natūralusis skaičius, didesnis arba lygus už 1 (kodo parametras), norimo persiųsti vektorių (dvejetetainio kodo pavidalu) atitinkamo ilgio ir klaidos tikimybę su tašku (pvz 0.1). Paspaudus **Encode and Transmit mygtuką** yra atliekamos laukų validacijos bei jei viskas tvarkoje, programa užkoduoja pranešimą bei persiunčia jį pro kanalą. Priešingu atvėju programa parodo klaidas ar neatitikimus ir neleidžia vykdyti toliau darbo, kol jos nebus ištaisytos. Taigi, pranešimų lange parodoma informacija kuris scenarijus dabar yra aktyvus, kaip atrodo užkoduotas vektorius, kaip atrodo vektorius po persiuntimo pro kanalą, kiek yra padaryta klaidų ir kokiose vietose tos klaidos yra padarytos. Vartotojas turi galimybę rankiniu būdu pakeisti išėjusį iš kanalo vektorių pagal savo poreikius ir norus. Paspaudus **Decode** mygtuką, programa vėlgi validuoja po kanalo išėjusį vektorių (tam kad įsitikinti,

ar po tarkim vartotojo manipuliacijų vektorius atitinka standartus) ir jei viskas tvarkoje, dekoduoja vektorių ir išveda jį pranešimų lange.

5.2 Antrasis scenarijus (teksto siuntimas)

Šioje kortelėje vartojo vėlgi prašoma įvesti m reikšmę, kuris yra natūralusis skaičius, didesnis arba lygus už 1 (kodo parametras), klaidos tikimybę su tašku (pvz 0.1) bei norimą persiųsti tekstą. M ir klaidos tikimybė yra validuojama, taip pat negalima siųsti tuščio teksto. Teksto lauke galima rašyti viską: dėti tarpus, įvairius simbolius, skaičius ar raides. Paspaudus **Send without encoding** mygtuką, vartotojo įvestas pranešimas yra transformuojamas į binarinę vektorinę išraišką, persiunčiamas pro kanalą ir atkonvertuojamas atgal į simbolius. Paspaudus **Send with encoding** mygutką, vartotojo įvestas tekstas yra transformuojamas į binarinę vektorinę išraišką, užkoduojamas, persiunčiamas kanalu, dekoduojamas ir konvertuojamas atgal į simbolius. Priklausomai nuo paspausto mygtuko, pranešimų lange yra išvedamas metodas, kuriuo buvo persiųsta žinutį (su kodavimu ar be) bei išvedamas dekoduotas tekstas.

5.3 Trečiasis scenarijus (nuotraukos siuntimas)

Šioje kortelėje vartojo vėlgi prašoma įvesti m reikšmę, kuris yra natūralusis skaičius, didesnis arba lygus už 1 (kodo parametras), klaidos tikimybę su tašku (pvz 0.1) bei taip pat prašoma įkelti BMP formato nuotrauką. M reikšmė, klaidos tikimybė ir nuotraukos formatai yra validuojami. Paspaudus **Send without encoding** mygtuką, programa konvertuoja įkeltą nuotrauką į binarinę vektorinę išraišką, persiunčiamas pro kanalą ir atkonvertuojamas atgal į paveiksėlį. Paspaudus **Send with encoding** mygtuką, vartotojo įkelta nuotrauka yra konvertuojama į binarinę vektorinę išraišką, užkoduojama, siunčiama pro kanalą, dekoduojama ir transformuojama atgal į nuotraukos pavidalą. Pranešimų lange yra išspausdinamas „Image transmission complete“ kai nuotrauka yra sėkmingai „praėjusi“ visus etapus.

Pastaba. Pasileidus programą ir paspaudus **send with encoding** mygtuka gali atrodyti, kad programa „pakibo“, tačiau taip iš tiesų nėra. Dekodavimo algoritmas pakankamai ilgai skaičiuoja (dar priklausomai nuo paduotos m reikšmės), todėl tenka šiek tiek palaukti.

6. Programiniai sprendimas

- **Generuojančios matricos sukūrimas (MatrixGenerator.generateEncodingMatrix(m))** – gavus m reikšmę, generuojanti matrica RM(1,m) yra sukūriama tokiu principu: apibrėžiami matricos eilučių ir stulpelių skaičiai (jos dydis). Kadangi pamačiau matricos šabloną (pattern), tai visada pirmąją matricos eilutę užpildau vienetais, o su sekančiom eilutėm ir stulpeliais, apsiskaičiuoju periodą (kas kiek stulpelių turi keistis reikšmės), ir naudodamasis mod 2 (kadangi turim F2 kūną), užpildau matricos stulpelius ir eilutes atitinkamai pagal periodą.
- **Vektoriaus užkodavimas(Encoder.encode())** – gautas vektorius (arba įvestas ir konvertuojas, priklauso nuo scenarijaus), yra dauginamas su generuojančia matrica (kiekvienas vektoriaus

elementas su atitinkamos matricos stulpelio visomis eilutėmis ir reikšmė yra sumuojama bei atliekama mod 2 dalyba).

- **Paprasto vektoriaus konvertavimas** – įvestas vektorius yra konvertuojamas iš string tipo (dėl įvesties) į dvejetainį pavidalą, bei konvertuojamas atgal į string – išvedimui.
- **Vektoriaus siuntimas kanalu(ChannelSimulator.simulateChannel())** – kanalo simuliacijoje pasitelkus sugeneruota ir iš pagrindinės klasės paduota random() reikšme bei įvesta klaidos tikimybe, tikrinama ar klaidos tikimybė yra didesnė už atsitikinę reikšmę. Jeigu taip, apverčiame vektoriaus bitus (0 -> 1 arba 1 -> 0). Taip pat kartu randame to bito poziciją . Apverstas vektorius atgal konvertuojamas į string tipą ir grąžinamas atgal.
- **Vektoriaus dekodavimas(Decoder.decode())** – pradžioje, gavus vartotojo įvestą m reikšmę, yra sugeneruojamos Hadamardo matricos(**HadamardMatrixGenerator.generateHadamardMatrices()**), skirtos dekodavimui. Apibrėžiama standartinė Hadamardo matrica H bei aprašoma vienetinių matricų konstravimo logika (**HadamardMatrixGenerator.generateIdentityMatrix())**.Vėliau pagal literatūroje duotą formulę yra konstruojamos H1_m, H2_m ir t.t. Hadamardo matricos, sudauginant prieš tai esančias Hadamardo matricas su įstrižainėm matricom (**HadamardMatrixGenerator.generateIntermediateMatrix())**. Pagrindinė dauginimo logika dvieju matricų yra įgyvendinta taip, jog matricos yra dauginamos „gabalais“, pasitelkus Kroneckerio sąndauga (**HadamardMatrixGenerator.kroneckerProduct()).** A Matricos pirmosios eilutės ir stulpelio reikšmė yra dauginama iš visos B matricos reikšmių, tada praplečiama gauta matrica, pereinama prie pirmosios eilutės antrojo stulpelio ir atliekama vėl sandauga na ir t.t., kol nebus sudauginti visi elementai. Kai turime visas reikiamas Hadamardo matricas, dekodavimui paduotą vektorių transformuojame – nulius keičiame į -1, o vienetus – paliekam, taip gauname w0 vektorių. Tada pagal literatūroje apibrėžtus žingsnius, w0 vektorių dauginame su H1_m Hadamardo matrica **(Decoder.multiplyVectorWithMatrix()**) ir gauname w1 na ir taip toliau dauginame, kol neprieiname paskutiniojo wm elemento. Jame randame didžiausią absoliučią reikšmę. Ją radus, randamas tos reikšmės indeksas. Indeksas yra verčiamas į dvejetainę išraišką, tačiau atvirkštine tvarka (mažesnio eiliškumo skaičiai priekyje). Tada pagal tai, kokia didžiausia reikšmė (teigiama arba neigiama) buvo wm vektoriuje, nustatome kokį skaičiu mum dar reikia prirašyti prie mūsų išversto indekso priekio (jei buvo teigiama reikšmė – prirašom 1, jei neigiama – 0).
- **Teksto transformacija į vektorius(TextTransformHelper.splitTextIntoVectors())** – teksto simbolis yra verčiamas į 16 bitų (UTF-16) binarinio pavidalo stringą ir jeigu vektorius yra trumpesnis nei 16 bitų, jo tuščios vietos yra konvertuojamos į nulius. Vėliau vienas ilgas stringas yra dalijamas į mūsų reikiamo vektoriu ilgius (m+1). Tai labiau yra aktualu paskutintam vektoriui iš to ilgojo stringo, bet

jeigu padalinus jis yra trumpesnis nei reikiamas vektoriaus ilgis, prie jo pridedame nulius iki kol jis bus reikiamo ilgio.

- **Vektorių vertimas į tekstą(TextTransformHelper.reconstructTextFromVectors())** – panašiu principu vyksta konvertavimas atgal – sudedame visus vektorius į vieną binarinį stringą ir tada juos daliname po 16 bitų. Kiekvieną gautą binarinį skaičiu verčiame į char simbolį taikant mod 2.
- **Nuotraukos transformacija į vektorius(ImageProcessingHelper.splitImageToVectors())** – Yra paimamas nuotraukos ilgis ir plotis, taip pat imamas kiekvienas pixelis ir gaunama to pixelio RGB reikšmė. Raudona yra shift right-inama per 16 pozicijų ir naudojamas loginis AND su 0xFF reikšme, žalia – per 8 bei taip pat loginis AND su 0xFF reikšme na ir mėlyna yra tiesiog loginis AND su 0xFF reikšme. Vėliau gautos reikšmės yra konvertuojamos į UTF-8 binarinį pavidalą (to užtenka RGB nuotraukom) ir tuščios vietos (jeigu pixelio spalvos kodas yra per trumpas reikiamam ilgiui) užpildomas nuliais. Na ir taip pat kaip su teksto konvertavimu, daliname vieną ilgą stringą į mūsų reikiamo ilgio stringus. Labiau aktualu paskutintam vektoriui, bet jei vektoriaus ilgis neatitinka reikiamo ilgio, užpildome jį nuliais.
- **Vektoriaus transformacija į nuotrauką(ImageProcessingHelper.reconstructImageFromVectors())** – Iš esmės viskas vyksta atvirkštine tvarka, tai jei mes turėjom UTF-8 formatą, tai kas 8 pozicijas turim apsibrėžę atitinkamai raudoną, žalią bei mėlynas spalvų RGB reikšmes, todėl tenka kas 8 pozicijas pasislinkti norint konstruoti atgal reikiamą stringą. Esamos reikšmės iš binary yra keičiamos į UTF-8 formatą ir tada atitinkamai shift left‘inamos, taip nustatant reikiamas pixelio spalvas. (raudona per 16 į kairę, žalia per 8 į kairę ir mėlyna – likusios 8 pozicijos). Po manipuliacijų pereinama 24 pozicijom į priekį. Na o jeigu bitų nepakanka, užpildome mėlyną spalvą nuliais (x,y,0)

7. Atliktas eksperimentas

7.1 Tikslas

Eksperimento tikslas buvo ištirti kiek laiko truks dekoduoti nuotrauką priklausomai nuo laipsnio m reikšmės, kur dekodavimo algoritmui buvo naudojamos greitosios Hadamardo transformacijos ir kai nuotrauka ir klaidos tikimybė yra nekintantys parametrai.

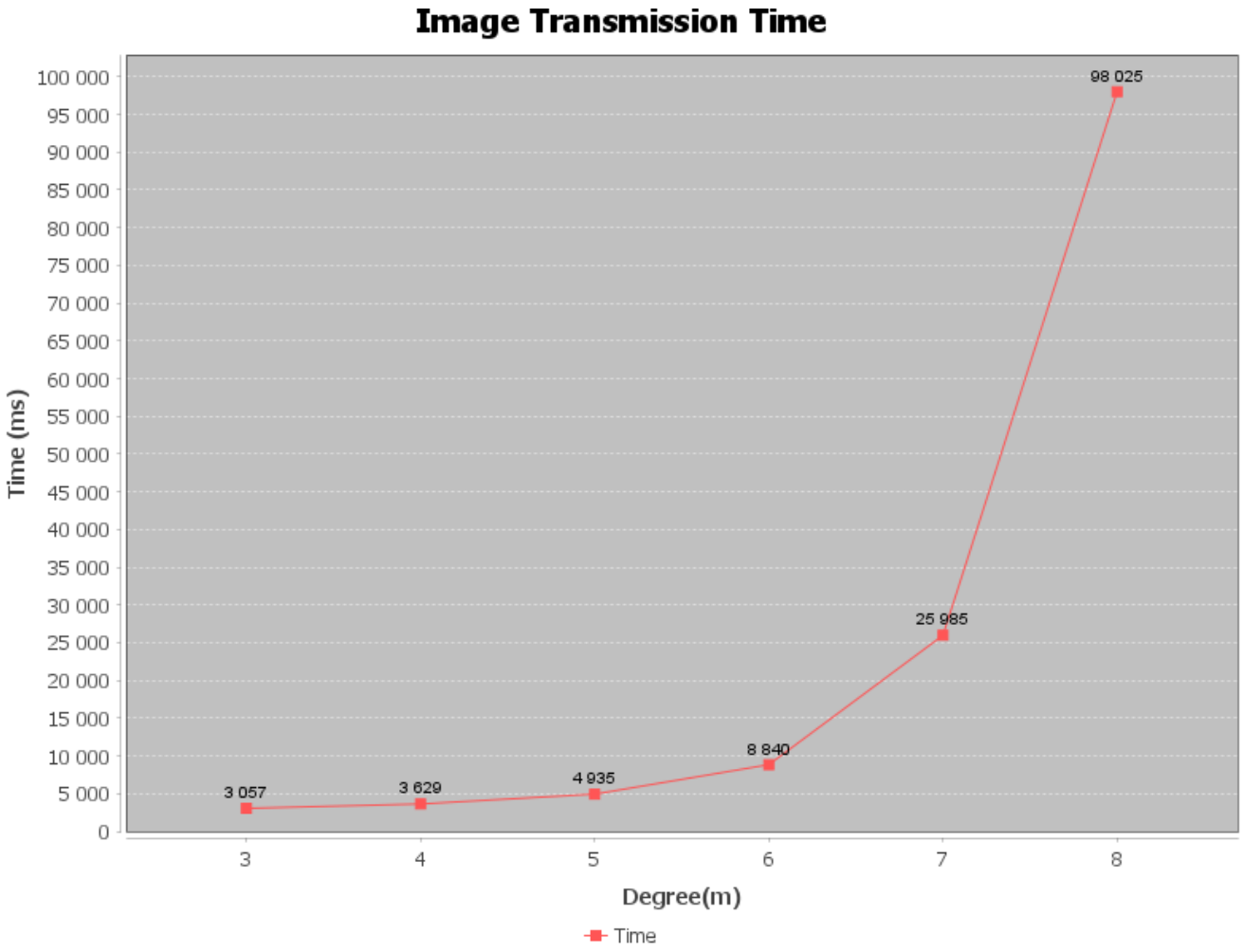
7.2 Eiga

- Nuotraukos konvertavimas į binarinę formą atitinkamo m+1 ilgio. Nuotrauka yra konvertuojama į binarinę vektorių formą naudojant UTF-8 koduotę bei padalinama į m+1 ilgio vektorius.
- Kodavimas. m+1 ilgio vektoriai užkoduojami naudojant Rydo-Miulerio generuojančią matricą.
- Siuntimas kanalu. Užkoduotas vektorius siunčiamas kanalu su tam tikra m reikšme.
- Dekodavimas. Gautas iš kanalo vektorius dekoduojamas naudojant greitąsias Hadamardo transformacijas.

- Rezultatų įvertinimas. Apskaičiuojamas sugaištas dekodavimo laikas.

7.3 Rezultatai

Eksperimentas atskleidė, kad naudojant 400x375 BMP formato nuotrauką su 0.15 klaidos tikimybe, dekodavimo laikas eksponentiškai auga didėjant laipsnio m reikšmei, o tai reiškia, jog dekodavimo algoritmas yra itin jautrus duomenų dydžio bei transformacijos sudėtingumo pokyčiams. Didžiausią įtaką dekodavimo laiko augimui turi greitųjų Hadamardo transformacijų matricų sudėtingumas, kuris proporcingai didėja kartu su m reikšme. Kai m = 3 dekodavimo laikas siekia vos 3067 ms, tai rodo, jog mažų dimensijų transformacijos yra efektyvios laiko atžvilgiu. Kai m = 6, laikas išauga iki 8840 ms, dvigubai daugiau nei palyginus su m = 5, kur laikas buvo 4935 ms. Tai rodo, kad laiko augimo tempas didėja nenuosekliai. Nors ir su mažesniom m reikšmėm laiko sugaištama mažiau, verta nepamiršti, jog maksimalus klaidų skaičius taip pat priklauso nuo m reikšmės.



1 pav. *Eksperimento grafikas*

7.3 Išvados

Atlikto eksperimento rezultatai parodo, jog didėjant laipsnio m reikšmei, eksponentiškai auga dekodavimo algoritmo sugaištas laikas. Todėl iš pirmo žvilgsnio atrodo, jog geriau naudoti mažesnes m reikšmes, tačiau reikia nepamiršti, jog maksimalus taisomų klaidų skaičius taip pat priklauso nuo m. Tokiu atveju geriau aukoti laiką, tačiau gauti tikslesnį atsakymą (geriau ištaisytą) arba ieškoti kitų dekodavimo algoritmų alternatyvų.

8. Literatūra ir šaltiniai

[HLL91] D.G.Hoffman, D.A.Leonard, C.C.Lindner, K.T.Phelps, C.A.Rodger, J.R.Wall. Coding Theory: The Essentials. Dekker, New York, 1991. 3.8–3.9, p. 89–95

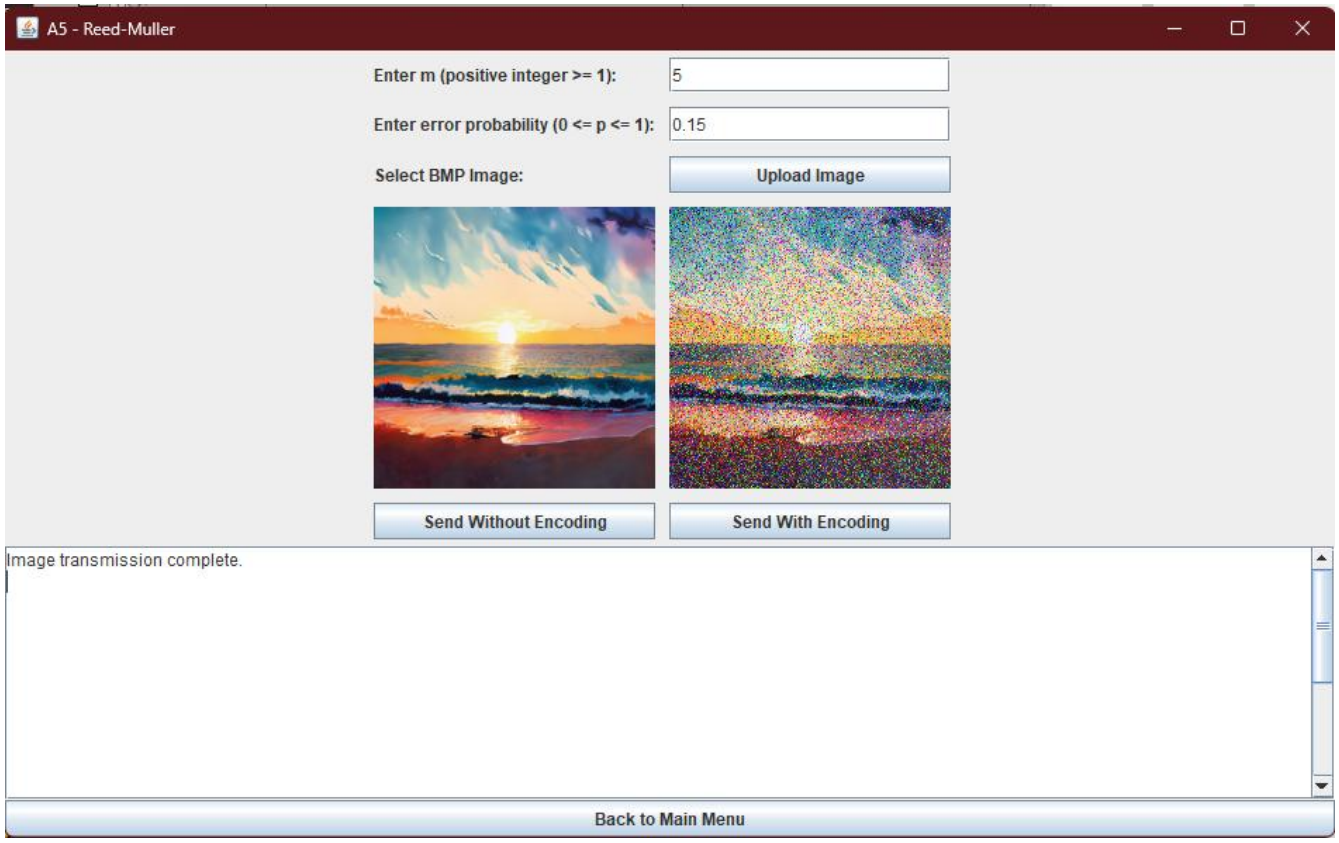
[Ske21] G. Skersys. Klaidas taisančių kodų teorija. Paskaitų konspektai, 2021.

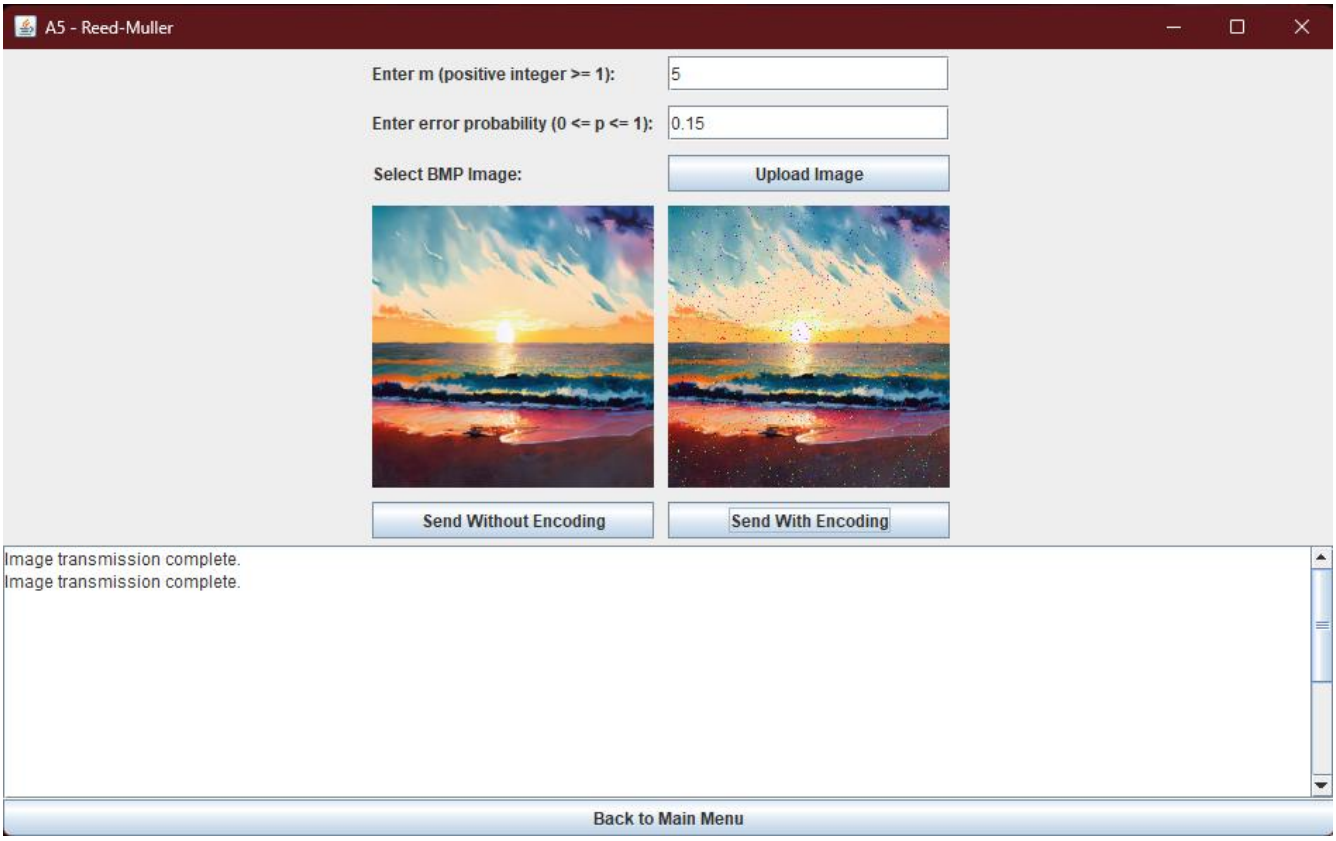
[Sta07] V. Stakėnas. Kodai ir šifrai. Vilnius, 2007.

[ASY20] E.Abbe, A.Shpilka, M.Ye Reed-Muller Codes: Theory and Algorithms, 2020

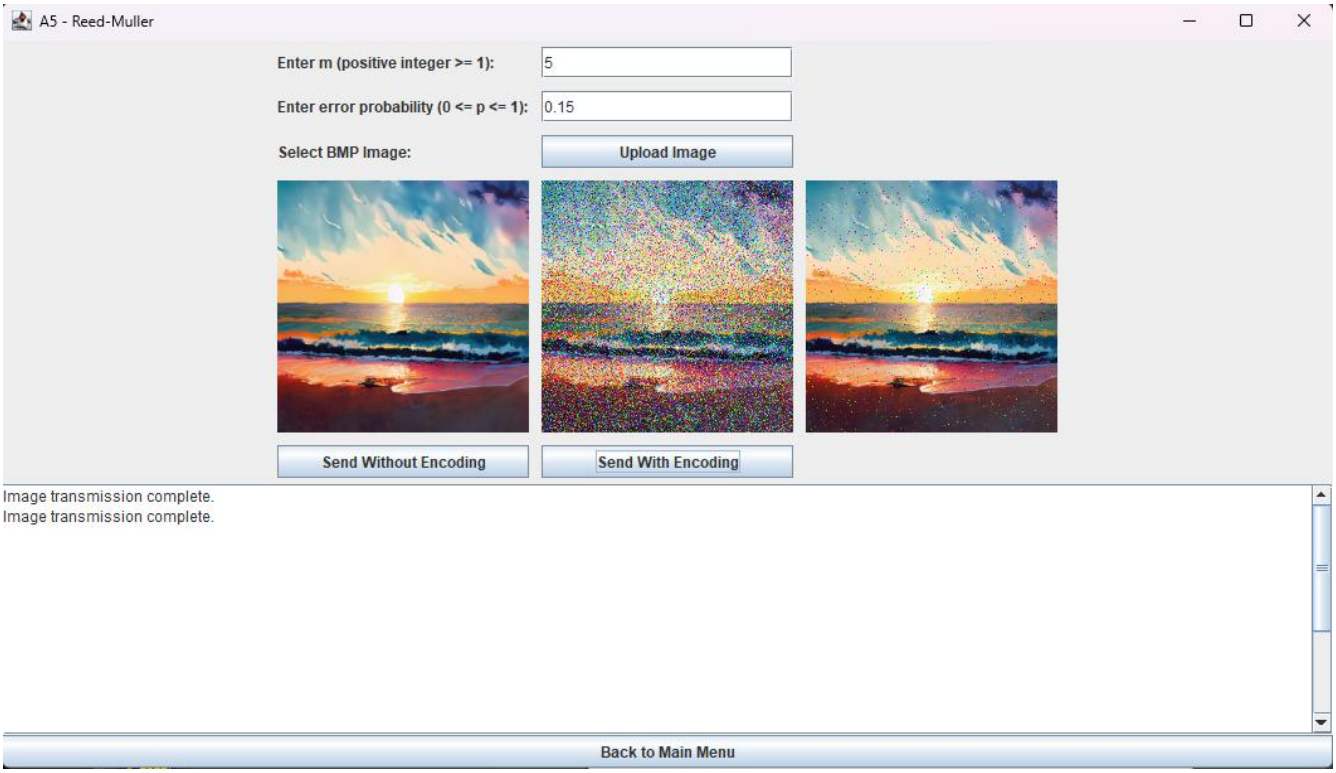
9. Atlikti pakeitimai po atsiskaitymo

- Ataskaitoje, vartotojo sąsajos poskyryje trečiajame scenarijuje buvo pridėta pastaba.
- Šiek tiek pakeistas kodas trečiame scenarijuje, jog būtu matomi iškart 3 paveikslėliai – skirtumas tarp jų (1 – originalas 2 - paveikslėlis prasiųstas pro kanalą be kodavimo/dekodavimo 3 - paveikslėlis su kodavimu/dekodavimu).
    - Buvo:




A5 - Reed-Muller
Enter m (positive integer >= 1):
5
Enter error probability (0 <= p <= 1):
0.15
Select BMP Image:
Upload Image
Send Without Encoding
Send With Encoding
Image transmission complete.
Image transmission complete.
Back to Main Menu

- Tapo:


A5 - Reed-Muller
Enter m (positive integer >= 1):
5
Enter error probability (0 <= p <= 1):
0.15
Select BMP Image:
Upload Image
Send Without Encoding
Send With Encoding
Image transmission complete.
Image transmission complete.
Back to Main Menu