A11 užduotis

Realizuotos dalys:

1. Kodavimas
2. Kanalas
3. Dekodavimas

Realizuoti scenarijai:

1. Vektoriaus siuntimas
2. Teksto siuntimas
3. Paveikslėlio siuntimas

Panaudotos trečiųjų šalių bibliotekos:

- javafx-controls, javafx-fxml, javafx-graphics – panaudota vartotojo sąsajos sukūrimui.
- richtextfx – panaudota, kad būtų galima raudonai ar žaliai nuspalvinti simbolius, kuriu pozicijose įvyko klaida.
- Junit-jupiter-api, junit-jupiter-engine – testams sukurti (nepanaudota, tačiau kuriant projektą jau buvo įtraukta ši biblioteka).

Panaudoti įrankiai vartotojo sąsajos kūrimui:

- SceneBuilder

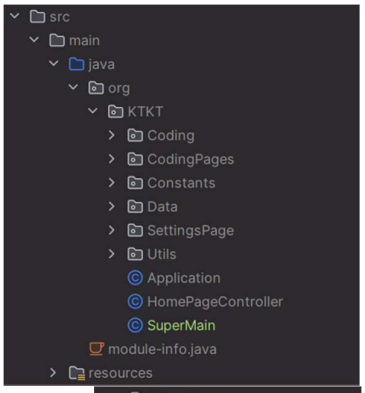
Skirtas laikas užduoties atlikimui (apytiklsiai)

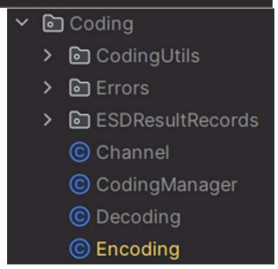
- Vaizdo įrašų žiūrėjimas ir analizavimas – 3h
- Kodo veikimo aiškinimuisi – 2h
- Vartotojo sąsajos kūrimui, validacijai – 10h
- Programavimui – 20h
- Klaidų ieškojimui, taisymui – 5h
- Skaitomumo gerinimui, komentarams – 2h
- Atsiskaitymui pasiruošimas – 8h

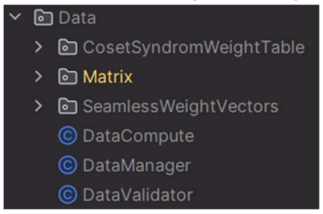
Kaip paleisti programą:

- Programa parašyta Java kalba.
- Programa rasite target aplankale.
- Reikia paleisti KTKT-1.0-SNAPSHOT.jar (galima du kartus paspausti ant failo)
- Paleidus programą nurodytą viršuje, turėtų atsidaryti langas skirtas dirbti su programa.
- Pastaba: jeigų skirtumas tarp n ir k parametrų yra didesnis negu 10, būtina programą paleisti su java -jar KTKT-1.0-SNAPSHOT.jar komanda.

Programos struktūra:



- 

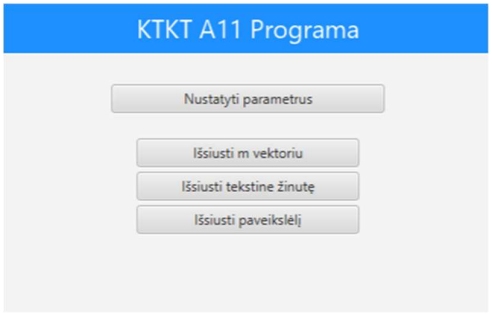
    - Channel – realizuoja kanalą.
    - Decoding – turi du viešus metodus: decodeWithoutReconstruction – iš (x|x*A) vektoriaus išima x dalį; decode – dekoduoją gautą vektorių iš kanalo pagal nurodytą A11 užduoties algoritmą.
    - Encoding – užkoduoja vektorių.
    - CodingManager – atsakingas už vartotojo paveikslėlio, žinutės ir vektoriaus m atitinkamai suskaidymą blokais (vektoriaus siuntimui tai netaikoma), išsiuntimą kanalu ir dekodavimą.
- CodingPages katalogas – saugo savyje visus vartotojo sąsajos elementus (controllers) skirtus 1, 2, 3 scenarijams.
- Constants katalogas – saugo savyje visas konstantas.
- 

    - CosetSyndromWeightTable – klasės, sindromų ir svorių lentelės realizacijos katalogas.
    - Matrix – matricos ir operacijų su ja realizacijos katalogas.
    - SeamlessWeightVectors – pagalbinės funkcija, skirta klasės, sindromu ir svoriu lentelei sudaryti. Pagrindinė jos funkcija: pagal duotą vektorių nustatyti koks turėtų būti kitas tokio pačio ar vienetu didesniu svoriu vektorius.
    - DataCompute – skirta sugeneruoti stadartinio pavidalo G matricai su atsitiktiniais duomenimis arba be jų. Apskaičiuoja H matricą pagal G matricą. Sugeneruoja klasės, sindromų ir svorių lentelę.
    - DataManager – atsakinga už n, k, G, H, klasės, sindromų ir svorių informacijos saugojimą, vartotojo įvesties užfiksavimą.
    - DataValidator – atsakinga už vartotojo įvesties validumą.

- SettingsPage – saugo savyje visus vartotojo sąsajos elementus (controllers) skirtus vartotojo įvesčiai.
- Utils – langų atidarymo pagalbinės funkcijos.
- Application, SuperMain – programos paleidimas
- HomePageController – pradinio puslapio elementas.

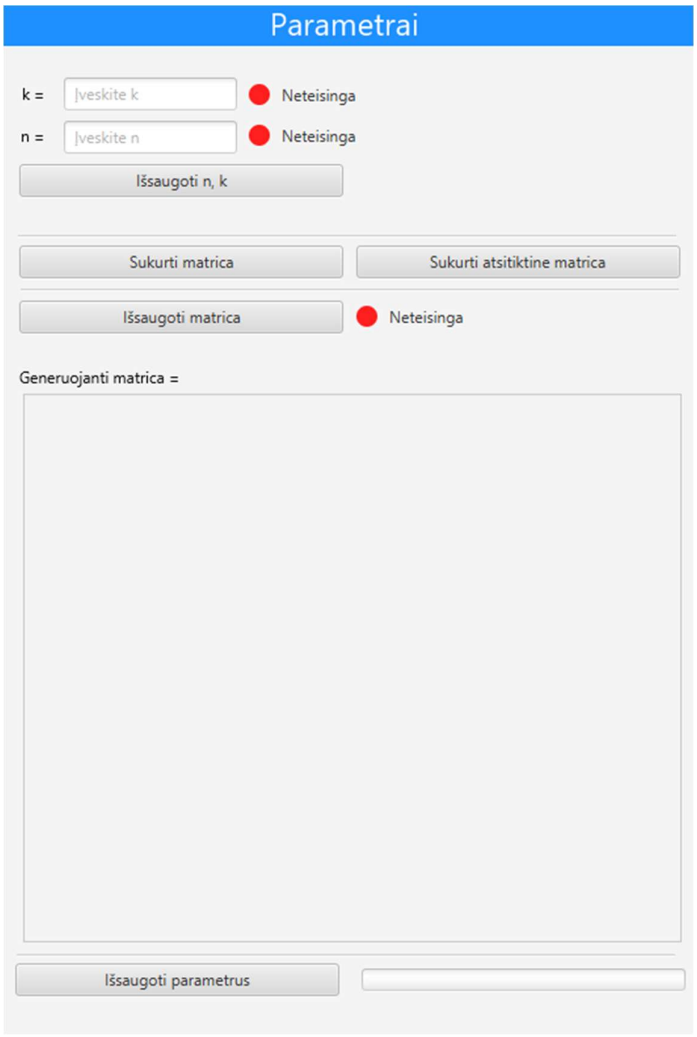
Vartotojo sąsaja:

Pradinis puslapis:



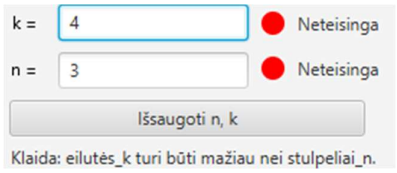
Prieš pasirenkant 1, 2 ar 3 scenarijų, būtina nustatyti parametrus.

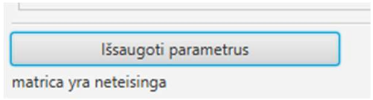
Parametrų puslapis:



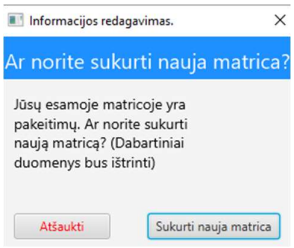
Atkreipkite dėmesį į tai, kad įvedus n, k ar kitus parametrus būtina išsaugoti juos.
Pavyzdžiui, jeigu įvesime n ir k, bet jų neišsaugosime, tai programa neleis sukurti matricos.
Atitinkamai jeigu sukursime matricą, įvesime korektiškus duomenis, bet matricos neišsaugosime, tai programa neleis mums išsaugoti parametrų kol neišsaugosime matricos.

Esant problemai programa visada praneš kas ne taip:

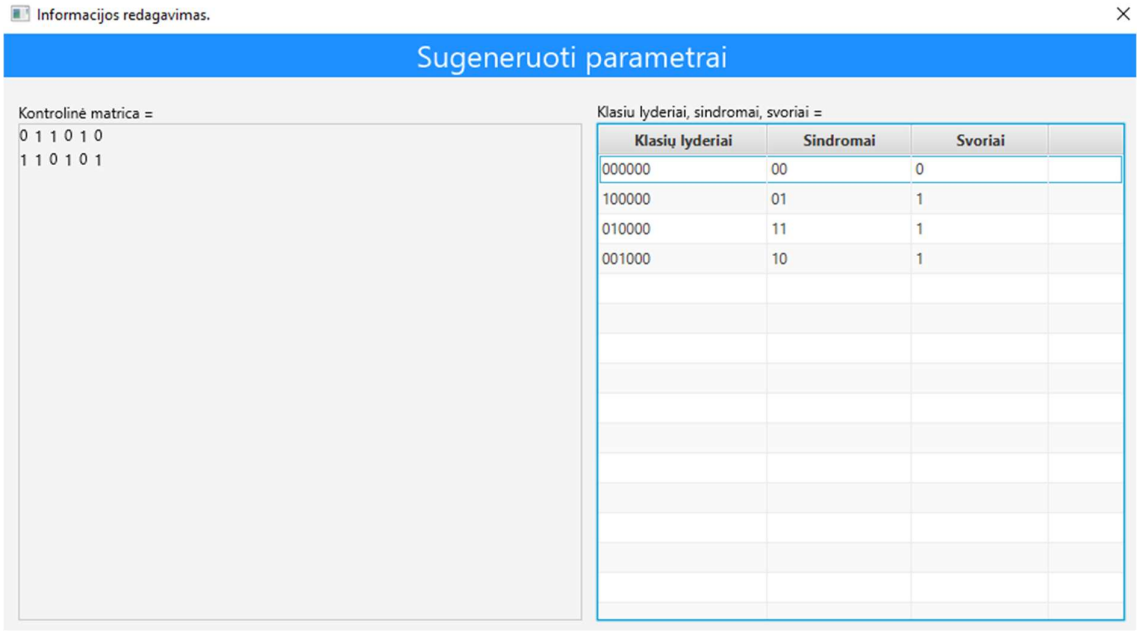




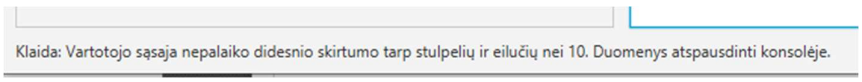
Jeigu bus bandoma sukurti naują matricą, kai esamoje matricoje yra atlikta pakeitimų, bus paklausta vartotojo ar jis tikrai nori sukurti naują matricą.



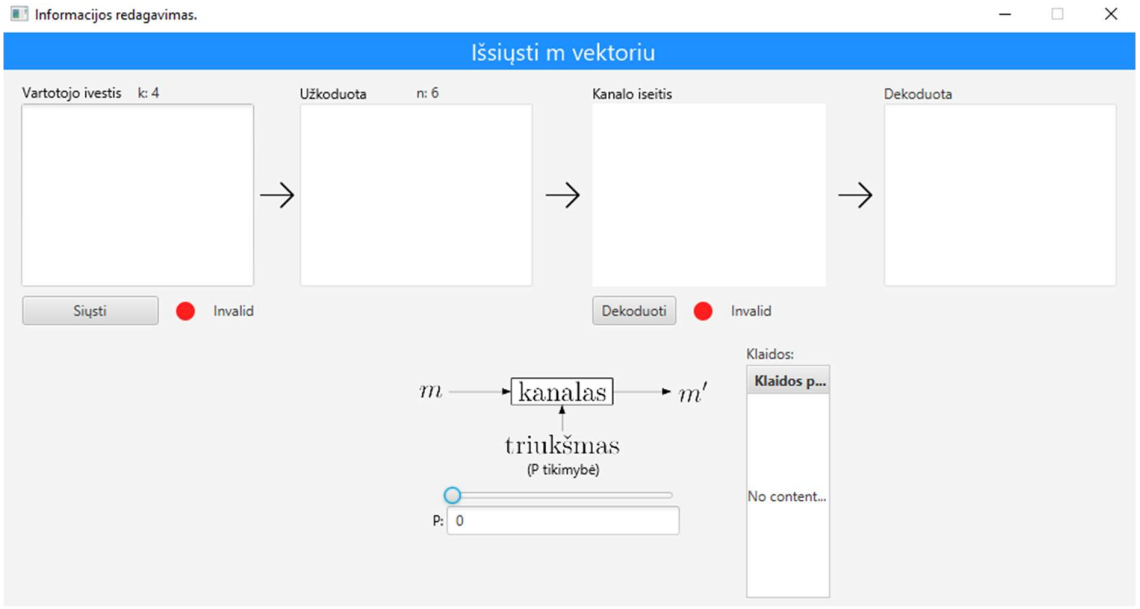
Kai visi parametrai yra korektiški ir išsaugomi, atsiranda papildomas langas rodantis sugeneruotus parametrus iš vartotojo įvesties.



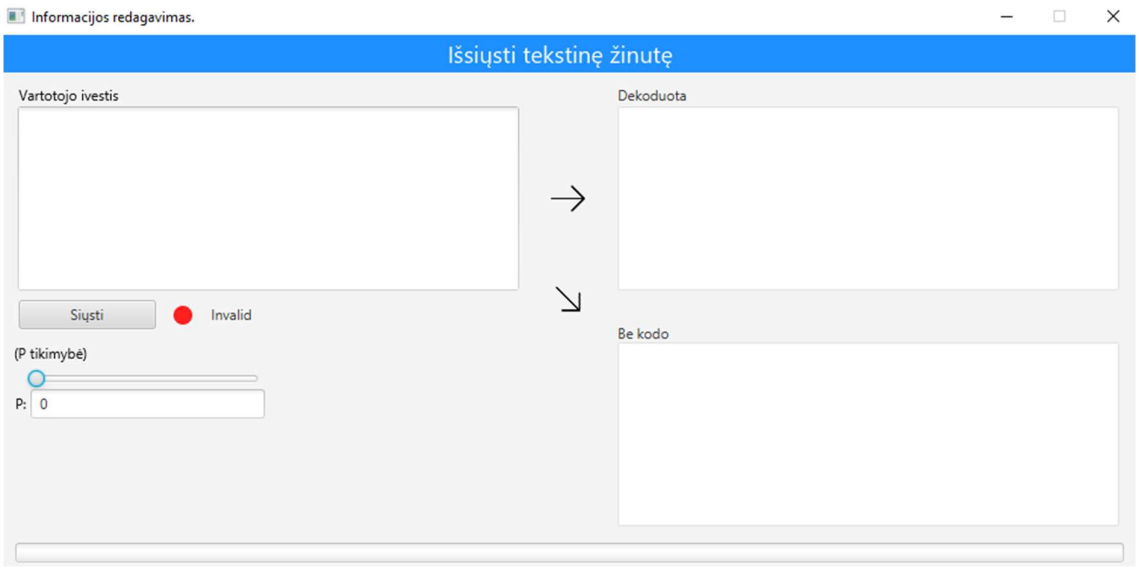
Atkreipkite dėmesį, kai n – k > 10, programa atvaizduos klasių lyderių, sindromų ir svorių lentelę terminale, tokiu atveju **būtina** paleisti programą naudojant java -jar KTKT-1.0-SNAPSHOT.jar komandą.



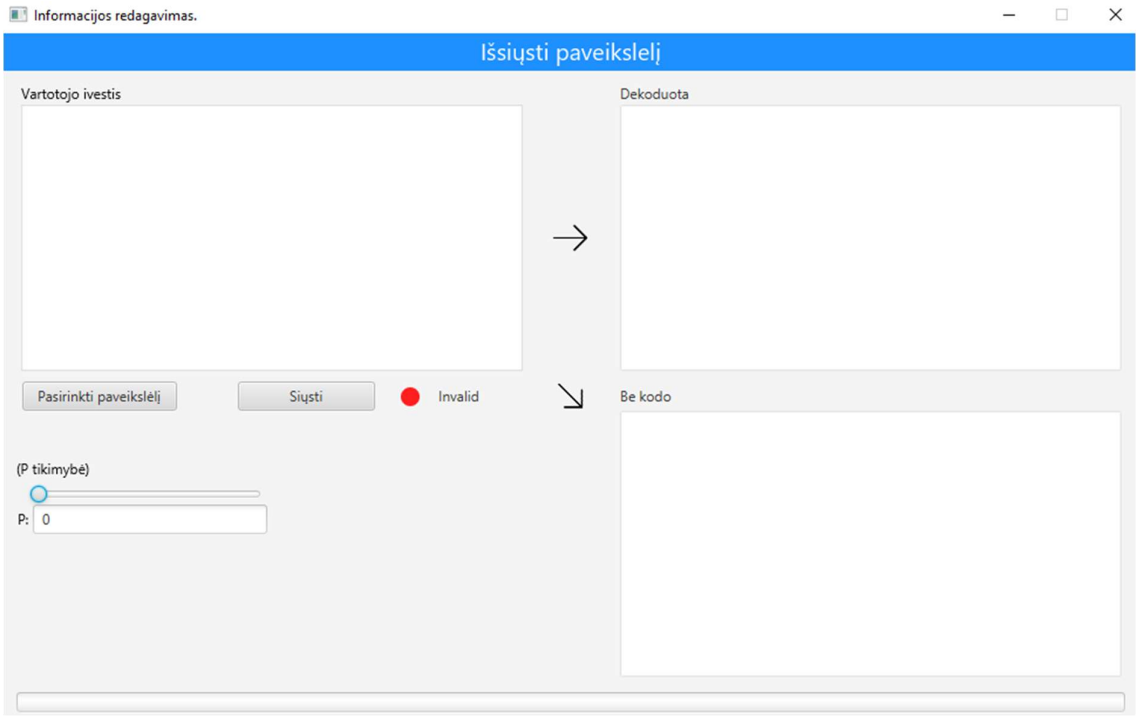
1 scenarijus



2 scenarijus



3 scenarijus



Apkreipkite dėmesį į tai, kad galima pasirinkti ir didelės raiškos (didesnius už 128*128) paveikslėlius, tačiau šių paveikslėlių apdorojimas gali užtrukti. Paveikslėlių siuntimą galima nutraukti uždarant scenarijaus langą.

Padaryti programiniai sprendimai:

- 2 ir 3 scenarijuje tekstą arba paveikslėlį pavertus vektoriumi patikriname ar šį vektorių galėsime suskaidyti blokais. Jeigu gausis nepilnas vektorius po suskaidymo, tai prie pagrindinio vektoriaus pridėsime tiek nulių, kad visi suskaidyti vektoriai būtų pilni.

Dekoduojant visi vektoriai yra sujungiami atgal į vieną ir yra pašalinama tiek simbolių, kiek buvo pridėta jų prieš užkodavimą blokais.

- Tekstas suskaidomas į baitus naudojantis getBytes() funkcija. Paverčiamas atgal į tekstą

```
java.lang.String

@NotNull ↗
@Contract(pure = true) ↗
public byte[] getBytes(
    @NotNull ↗ java.nio.charset.Charset charset
)
```

naudojant new String(byteArray, StandardCharsets.UTF_8) konstruktorių.

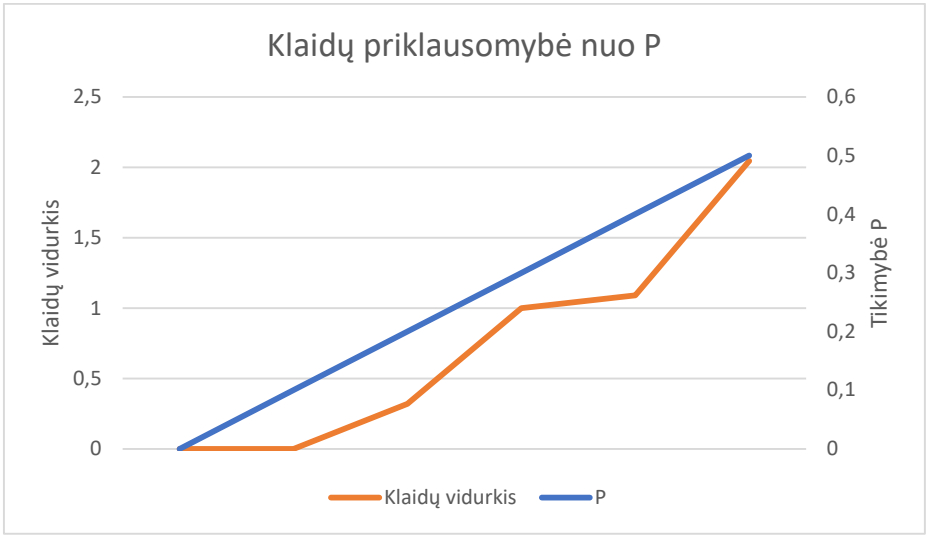
```
java.lang.String

@Contract(pure = true) ↗
public String(
    @NotNull ↗ byte[] bytes,
    @NotNull ↗ java.nio.charset.Charset charset
)
```

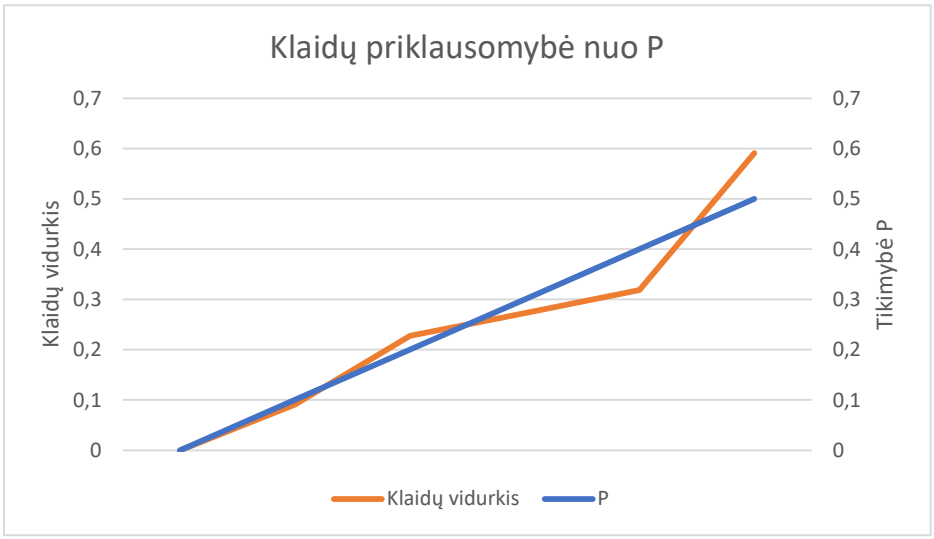
- Nuotrauka skaidoma į bitų seką taip: iš nuotraukos nuskaitomos RGB reikšmės x, y pozicijoje. Šios RGB reikšmės verčiamos iš Int reikšmės į bitų seką. Nuotrauka iš bitų sekos nuskaitoma šitaip: nuskaitomos tris kartus po 8 bitus sekos, kurios verčiamos į RGB reikšmes. Nuotraukos aukštis ir plotis siunčiamas mūsų sutartu tarnybiniu kanalu.
- Kai n ir k skiriasi daugiau negu 10, atspausdinti rezultatus terminale. Šitaip pasielgta todėl, nes kai n ir k skiriasi daugiau negu 10, eilučiu skaičius stipriai padidėja ir javafx vartotojo sąsaja nebesugeba apdoroti šį didelį eilučių skaičių. Pavyzdžiui kai k = 1 ir n = 12, tai n-k = 11, o tai yra 2^{11} = 2048 eilutės, kurių įterpimas į javafx lentelės elementą gali nulaužti programą.
- k ir n negali būti didesni už 50. Kaip ir praeitame sprendime, buvo nuspresta apriboti šiuos parametrus, kad vartotojo sąsajos darbas nesuletėtų.

Eksperimentai:

k = 4; n= 16; G = (I|A), kur A atsitiktinai sugeneruotas; Naudoti pranešimai 0000 ir 1111. Tikimybės žingsnis = 0,1. Pranešimai 0000 ir 1111 kartoti 11 kartų. Rodomas klaidų vidurkis.

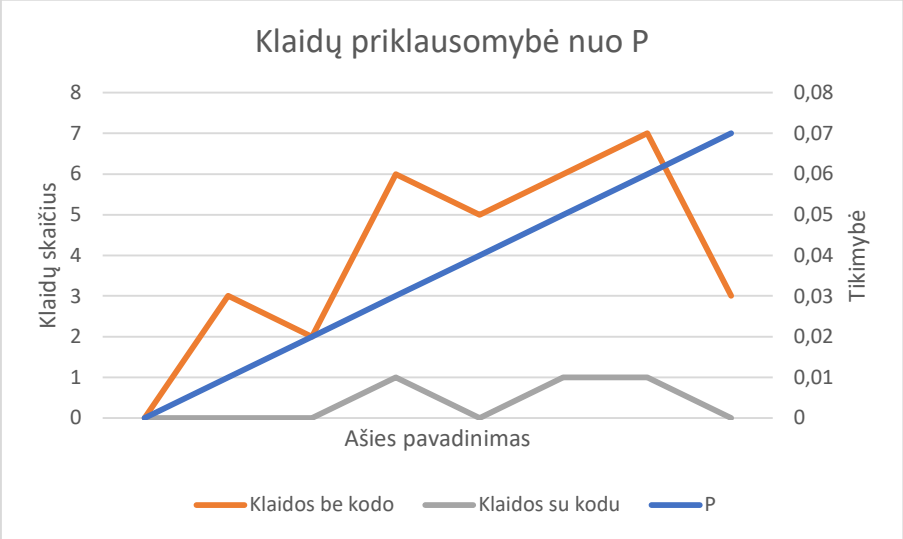


k = 1; n = 16; G = (1000000000000000); Naudoti pranešimai 1 ir 0. Tikimybės žingsnis = 0,1. Pranešimai 1 ir 0 kartoti 11 kartų. Rodomas klaidų vidurkis.



k = 4; n= 16; G = (I|A), kur A atsitiktinai sugeneruotas; Pranešimams naudotas paveikslėlis  . Nuo 0 iki 0,4 tikimybės žingsnis = 0,05. Nuo 0,4 ir toliau tikimybės žingsnis = 0,1.

k = 4; n= 16; G = (I|A), kur A atsitiktinai sugeneruotas; Pranešimas = Kodavimo teorija. Tikimybės žingsnis = 0,01. Pranešimas buvo siunčiamas tik vieną kartą kadangi duomenis palyginti buvo sunku, kai kurie simboliai tampa nematomi.



Pastebėjimai:

- Kai A=0 ir k=1 tai tik pirmas bitas iš kanalo nusako ar pranešimas bus dekoduotas kaip 1 ar 0. Kai k>1 ši taisyklė nebegalioja. Siunčiant tekstą gali atrodyti, kad mūsų algoritmas neveikia kadangi tiek su kodu, tiek be kodo gaunamas tas pats pranešimas, bet tokie rezultatai gaunami būtent, nes A = 0 ir k = 1. Analogiškai ir su paveikslėliu tiek su kodu, tiek be kodo gauname vienoda rezultatą kai A = 0 ir k = 1.
- Siunčiant nuotrauką, kai kuriose vietose kodas pikselius ištaiso dalinai. Taip yra todėl, kad pikselių vertės skirstomos į RGB (raudoną, žalią, mėlyną), todėl net jeigu ir įvyko klaida keliose reikšmėse, tai kodas galėjo ištaisyti bent vieną iš tų reikšmių.
- Kai p tampa didesnis už 0,5 pranešimas be kodo turi mažiau triukšmo negu pranešimas su kodu.

p = 1; kairėje – su kodu, dešinėje – be kodo

- Kai p tampa didesnis už 0,6, iš laiko priklausomybės nuo p grafiko galime matyti, kad apskaičiavimų trukmė sumažėja. Tačiau kai p = 1 skaičiavimų trukmė nėra lygi trukmei su p = 0.

Kitus eksperimentus galite rasti „eksperimetai“ pdf faile.

Naudota literatūra:

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