Jaana Kivistö, 829964, Bioinformaatioteknologia, vuosikurssi: 2020, 26.2.2021

Proteiinisynteesi

Tein ohjelman, joka mallintaa proteiinisynteesin päätapahtumia. Ohjelma lukee tiedostosta DNA-sekvenssin, lukee DNA-juovasta RNA:n (transkriptio) ja muodostaa peptidiketjun RNA:n emästen mukaan (translaatio). Graafisessa käyttöliittymässä esitetään (1) tumassa tapahtuva transkriptio ja mahdollinen virhe DNA:n luvussa sekä sekvenssistä mahdollisesti löytyvät intronit ja (2) ribosomissa tapahtuva translaatio yksinkertaistettuna eli RNA-juoste, josta poistettu mahdolliset intronit ja juosteen mukaan tehty peptidiketju. Intronit on värjätty tummemmalla värillä kuin eksonit. Mutaatiot eroavat muusta juosteesta värinsä (kirkas fuksia) ja istumattoman muotonsa puolesta. Peptidiketjun aminohapoille on jokaiselle omat värit. Ohjelmasta suoritettu vaikean vaikeustason mukaan. Yksilöivänä tekijänä ohjelmassa on mahdollisuus valita (tai olla valitsematta) virhemahdollisuuden todennäköisyys DNA:n luvussa.

Käyttöohje

Ohjelmaa ajetaan src/main.py-tiedoston kautta. Ohjelman käynnistyttyä, ensin aukeaa ikkuna (Liite 1), joka kysyy DNA-sekvenssitiedoston nimeä (muotoa .txt, pituudeltaan 500–1000 emästä) ja virhemahdollisuuden (annetaan desimaalina). Ohjelmaa pystyy myös ajamaan ilman virhemahdollisuutta, jolloin kentän voi jättää tyhjäksi. Simulaatio käynnistyy painamalla ”Run”.

Ohjelmaan avautuu seuraavaksi ikkuna (Liite 2), jossa on kuvattuna tuma (Nucleus) ja siellä tapahtuva transkriptio ja sen alapuolelle ribosomi (Ribosome), jossa on kuvattuna RNA-juoste ja peptidiketju. DNA-juosteen mitta riippuu siitä, missä kohtaa tiedostoa sekvenssin aloittava ”TATA-box” sijaitsee. Simulaatio-ikkunan alareunassa olevan ”sliderin” avulla voi tutkia DNA-juostetta ja tarvittaessa etsiä RNA-juosteen, joka ei aina mahdu samaan näkymään peptidiketjun kanssa. Liitteistä 3 ja 4 näkee tummemmaksi värjätyn intronin sekä useamman fuksian värisen ”mutatoituneen” emäksen. Ribosomissa RNA-juosteessa näkyvät mutaatiot on vaihdettu takaisin oikean värisiksi ja intronit on poistettu.

Ohjelman lopetus tapahtuu oikean yläkulman ruksista.

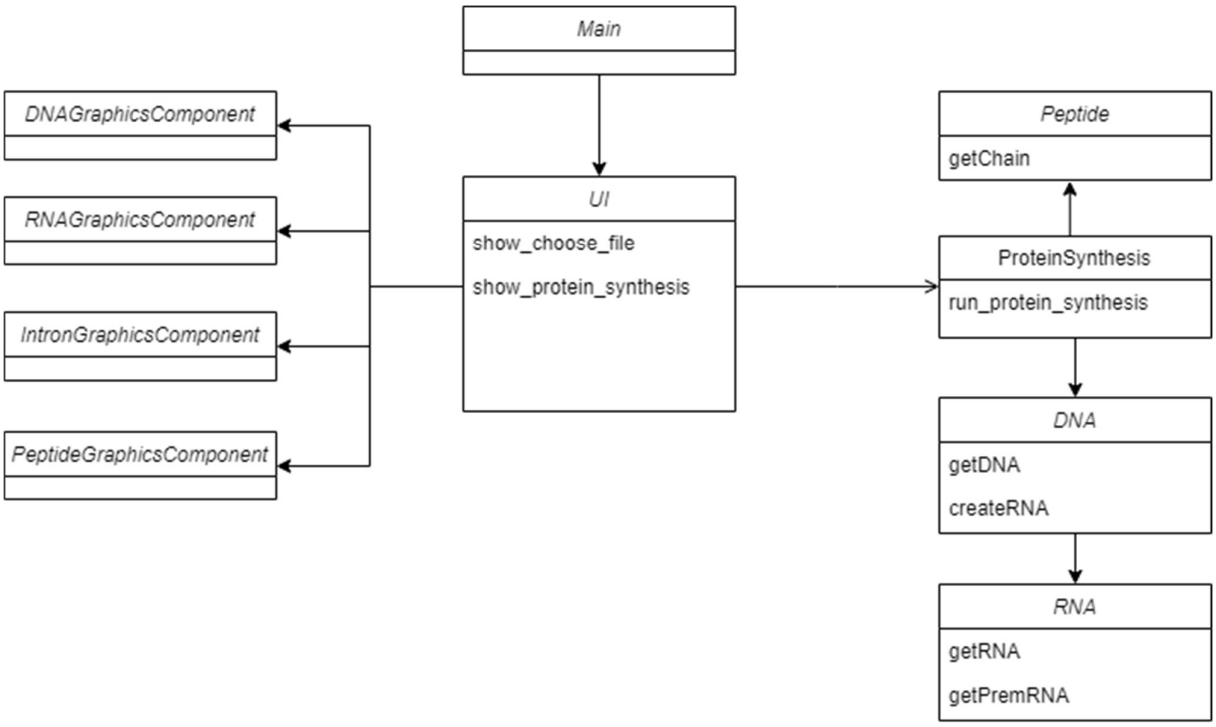
Ulkoiset kirjastot

Ohjelmassa on käytetty ohjeen mukaisesti vain yhtä ulkoista kirjastoa: PyQt. Kyseistä kirjastoa on käytetty graafisen käyttöliittymän toteuttamiseen. Testaamisessa on käytetty myös pytest kirjastoa, mutta ohjelman toimiminen ei vaadi sitä. Kirjaston voi asentaa pip työkalun avulla pip3 pytest install komennolla. Yksikkötestit voi ajaa komentoriviltä käskyllä pytest src.

Ohjelman rakenne

Käyttöliittymä ja sovelluslogiikka on jaettu erillisiin luokkiin. Käyttöliittymän toimintaa ohjaa UI luokka ja sovelluslogiikkaa ohjaa ProteinSynthesis luokka. Käyttöliittymäkomponentteihin kuuluvat UI luokan lisäksi DNAGraphicsComponent, RNAGraphicsComponent, IntronGraphicsComponent ja PeptideGraphicsComponent. Sovelluslogiikkaan kuuluu

ProteinSynthesis luokan lisäksi DNA, RNA ja Peptide luokat, sekä Exceptions-tiedosto, jossa on määritelty kaksi omaa virhetyyppiä, InvalidDNASequence ja EmptyRNAException. Luokat, sekä niiden metodit, joita muut sovelluksen osat kutsuvat ovat kuvassa 1 alla. Sovelluksen toiminnan aloittaa omassa tiedostossaan oleva main, joka luo Qt applikaation, kutsuu UI luokkaa ja käynnistää käyttöliittymän suoritussilmukan.



Kuva 1 Luokkakaavio

UI aukaisee aloitusikkunan kutsumalla omassa initissään show_choose_file metodia. Tämän tehtävänä on pyytää käyttäjältä tarvittavat tiedot, eli mutaation mahdollisuus ja tiedostonimi. Mikäli myöhemmässä vaiheessa tulee virheitä, ne ohjataan näkymään aloitusikkunassa, jotta käyttäjä voi tarvittaessa valita uuden tiedoston ja näkee mistä virhe johtui. Virheet itsessään nostetaan sovelluslogiikan eri osista. Toinen vaihtoehto virheviesteille olisi ollut proteiinisynteesi-ikkuna, mutta mielestäni aloitusikkuna oli parempi vaihtoehto sen takia, että käyttäjän ei tarvitse käynnistää ohjelmaa uudestaan.

Aloitusikkunan ”Run” -napin painaminen kutsuu UI-luokan handle_load_file-metodia, joka luo ProteinSynthesis-olion ja sen jälkeen aloittaa varsinaisen proteiinisynteesiprosessin kutsumalla luokan metodia run_protein_synthesis. ProteinSynthesis-luokka siis kuvaa ja koordinoi proteiinisynteesiprosessia, minkä lisäksi sen tehtävänä on tarjota UI-luokalle rajapinta piirtämiseen tarvittaviin tietoihin, eli RNA-, DNA- ja peptidiketjuihin.

Luokka kutsuu ensin DNA-luokkaa ja antaa sille parametrina tiedostonimen. DNA-luokan tehtävä on kuvata DNA-ketjua, sekä tallentaa siihen liittyvä RNA-olio ja aloittaa ProteinSynthesis-luokan

pyynnöstä RNA:n generointi. DNA-luokka parsii annetun tiedoston ja tallentaa siinä olevan DNA-ketjun stringinä. Mikäli tiedosto ei ole sopiva, se nostaa virheen, jonka ProteinSynthesis välittää käyttöliittymälle.

DNA:n luonnin jälkeen ProteinSynthesis kutsuu DNA-luokan metodia createRNA, DNA parsii ensin omasta DNA-ketjustaan sen osan, joka on relevantti RNA:n luomista varten ja sitten luo RNA olion. RNA-luokka kuvaa RNA-juostetta. Sen tehtävät ovat RNA-juosteen muodostaminen DNA:sta, intronien erottaminen, mutaatioiden tekeminen ja RNA-ketjun tallentaminen. Virhetilanteessa nostetaan virhe, joka välittyy käyttöliittymälle.

RNA:n luonnin jälkeen ProteinSynthesis luo peptidiketjun luomalla Peptide-olion luodun RNA:n perusteella. Peptide-luokka kuvaa yhtä peptidiketjua. Sen tehtävänä on luoda peptidiketju RNA:n perusteella ja tallentaa ketju.

DNA, RNA ja peptidi toiminnallisuudet olisi voinut tehdä myös niin, että ProteinSynthesis olisi hoitanut myös RNA:n tallentamisen ja luomisen suoraan, eikä DNA-luokan kautta. Kuitenkin RNA kuuluu loogisesti tiettyyn DNA:han, minkä lisäksi tämä saattaisi auttaa ohjelman laajentamisessa niin, että yhdellä DNA:lla voi olla monta proteiineja koodaavaa RNA-juostetta. Peptidin olisi voinut samalla tavalla kiinnittää RNA-luokkaan, mutta toteutuksesta tuli yksinkertaisempi näin. Tällainen muutos saattaisi silti olla laajennettavuuden näkökulmasta perusteltua.

Kun proteiinisynteesi on onnistuneesti suoritettu, UI luokka käyttää show_protein_synthesis metodia synteesin piirtämiseen. ProteinSynthesis-olio tallennetaan luokkamuuttujaksi ja siitä haetaan tarvittaessa synteesiin liittyvät tiedot.

Piirtäminen tapahtuu QGraphicsScenen avulla. Pohjalle piirretään ensin laatikot ja tekstit. Tämän jälkeen ohjelma piirtää DNA:n ja RNA:n silmukassa. Jokainen piirrettävä DNA-objekti on DNAGraphicsItem luokan olio. Luokka perii QGraphicsPolygonItemin ja sen tehtävä on määrittää annetun DNA-tyypin perusteella objektin muoto ja väri. UI luokka piirtää erikseen DNA:ta kuvaavan kirjaimen luodun objektin päälle.

RNA:n piirtäminen tapahtuu samassa silmukassa. DNA-luokalta saadaan tieto siitä, millä kohtaa RNA:n piirtämisen pitää alkaa. Piirrettävät RNA-objektit ovat joko RNAGraphicsItem- tai IntronGraphicsItem-luokan olioita riippuen siitä onko kyseessä eksoni vai introni. Tämä tieto saadaan RNA-luokalta. Molemmat grafiikkaluokat perivät QGraphicsPolygonItemin ja niiden tehtävänä on määrittää RNA-tyypin perusteella objektin muoto ja väri. Lisäksi ne saavat RNA:lta tiedon siitä, onko kyseisessä kohdassa mutaatiota, joka piirretään eri värillä.

RNA:n ja DNA:n piirtämisen jälkeen ribosomi-osuus, jossa RNA piirretään uudelleen käyttäen RNAGraphicsItem-luokkaa. Tällä kertaa intronia ei piirretä. Lisäksi piirretään Peptidi-luokan perusteella peptidiketju. Tässä on apuna PeptideGraphicsItem luokka, joka perii QGraphicsEllipseItemin. Luokka on vastuussa peptidin muodosta ja sillä on lisäksi dictionary, jonka perusteella peptidien väri määritellään. Peptidien piirtämisen jälkeen uusi näytetään uusi näkymä, johon prosessi päättyy.

Piirtämisessä vaihtoehtoinen tapa olisi ollut piirtää RNA:t yhdessä luokassa lisäämällä parametriksi tiedon siitä onko kyseessä introni vai ei, mikä olisi saattanut hieman yksinkertaistaa koodia. Myös Intronien toteutuksen olisi voinut tehdä eri tavalla. Nykyisessä muodossaan, introneja ei löydy yhtä useasti pre-mRNA:sta. Intronien aloituksen olisi myös voinut etsiä käymättä juostetta läpi kolmen emäksen pätkissä. Introneja saattaisi sekvenssistä riippuen löytyä useampi. Halusin kuitenkin, että

peptidiketjut ovat keskimäärin pidempiä kuin kolme aminohappoa, joten päädyin yhteen introniin per sekvenssi. Tähän päätökseen vaikutti myös se, että testisekvenssit eivät ole aitoja tai toimi aidon sekvenssin tavoin sekä ovat huomattavasti lyhyempiä kuin aidot sekvenssit. Useamman pienen intronin etsiminen olisi useassa tapauksessa tarkoittanut sitä, että lopullisesta RNA:sta olisi välistä täytynyt poistaa ylimääräisiä emäksiä, jotta kolmen jaollisuus säilyisi ja peptidiketjun muodostaminen lopetuskodoneineen toteutuisi.

Algoritmit

Ohjelman käyttämät algoritmit ovat melko yksinkertaisia, mutta tässä on esitelty niistä tärkeimmät.

DNA:n luominen

DNA luodaan etsimällä ensin silmukassa parsimalla TATA-merkki, jonka jälkeinen osa tallennetaan DNA-merkkijonoksi.

RNA:n luominen

DNA-merkkijonoa parsitaan kolmen indeksin välein, kunnes löydetään ”tac” merkkijono. Tämän jälkeen etsitään samalla tavalla kolmen indeksin välein silmukassa parsimalla lopetusmerkkijono, joka on ”att”, ”atc” tai ”act”. Näiden välisestä osuudesta muodostetaan RNA-merkkijono.

RNA:lle on lisäksi DNA-luokassa rna_min_length muuttuja, jonka avulla voi asettaa minimin sille kuinka lyhyeksi RNA voi jäädä. Tämä toteutuspäätös johtui siitä, että monella tiedostolla RNA jäi todella lyhyeksi ilman tätä. Se on vakiona asetettu 10:n merkkiin, mutta on helposti muokattavissa.

Intronit

Tallennettu RNA-merkkijono parsitaan silmukassa kolmen indeksi välein etsien intronin aloittavaa merkkijonoa ”gu” jokaisen kodonin, eli kolmen RNA-merkkijonossa olevan merkin alusta. Mikäli tällainen löytyy, etsitään lopettava ”ag” merkkijono vastaavasti parsimalla kolmen merkin välein RNA-merkkijonoa, joskin ”ag” merkkiä etsitään kodonin lopusta. Näiden indeksien väliin jäävä osuus on introni ja muut osuudet RNA-merkkijonosta ovat eksoneita.

Mutaatiot

Mutaatiot luodaan käyttämällä pythonin random-kirjastoa. Koko RNA-merkkijono parsitaan läpi niin, että jokaisen kohdalla otetaan random-kirjaston random funktion avulla float luku välillä 0-1. Tätä lukua verrataan käyttäjän antamaan mutaation todennäköisyyteen. Mikäli satunnaisluku on pienempi, muutetaan kyseinen merkki toiseksi mahdolliseksi merkiksi. Merkin vaihtaminen tapahtuu arpomalla random-kirjaston randint funktiolla kokonaisluku väliltä 0-3, jota käytetään uuden RNA-merkin valitsemiseen ja toistamalla tätä kunnes merkki vaihtuu.

Peptidit

Peptidit luodaan RNA-merkkijonosta ottamalla siitä silmukassa kolme merkkiä kerralla ja muuttamalla ne dictionary-tietorakenteen avulla peptidejä kuvaaviksi merkkijonoiksi. Prosessi lopetetaan, kun RNA-merkkijono loppuu tai vastaan tulee yksi prosessin lopettavista peptideistä.

Tietorakenteet

Ohjelmassa on käytetty monipuolisesti erilaisia tietorakenteita. DNA- ja RNA-juosteet ovat stringejä (immutable), peptidit ja emäkset on varastoitu sanakirjoihin (mutable), peptidiketju on taulukkomuodossa (mutable) ja mutaatioiden indeksit on tallennettu set-tietotyyppiin (mutable).

DNA- ja RNA-juosteiden käsittely string-muodossa on mielestäni helpointa pythonin sisäänrakennettujen stringin käsittelymetodien vuoksi. Sama pätee peptidiketjun taulukkomuotoon. Sanakirjojen rakenne puolestaan toimii täydellisesti, kun tietylle kirjaimelle tai kirjainyhdistelmälle etsitään oikeaa vastinetta. Set-tietotyyppi on nopeampi, kun indeksien määrä kasvaa isommaksi, mutta tähän käyttötarkoitukseen setin tilalla olisi voinut olla lista tai taulukko. Myös peptidiketju olisi voinut olla listamuodossa taulukon sijaan.

Tiedostot

Ohjelma käsittelee ainoastaan tekstitiedostoja (.txt), joissa on DNA-sekvenssiä kuvaava juoste (biologiasta poiketen vain yksöisjuoste). Oletettu tiedostojen pituus on 500-1000 emästä, tosin ohjelma toimii myös muilla pituuksilla (muilla pituuksilla lopputulos ei välttämättä ole enää optimaalinen). Testitiedostoja pystyy helposti tuottamaan Sequence Manipulation Suite: Random Sequence-generaattorilla (https://www.bioinformatics.org/sms2/random_dna.html).

Testaus

Testaus vastaa suunnitelmaa. Sovelluksen ja erityisesti käyttöliittymän toiminta testattiin manuaalisesti erilaisilla tiedostoilla ja syötteillä, jotka olivat eri pituisia ja sisällöltään erilaisia. Osa Tiedostoista oli myös virheellisiä. Manuaalisen testauksen yhteydessä käytettiin myös konsoliprinttejä, joiden avulla piirrettyä kuvaa ja laskettuja arvoja voitiin verrata.

Manuaalisen testauksen lisäksi käytettiin yksikkötestausta. Yksikkötestit luotiin protein_synthesis.py, dna.py, rna.py ja peptide.py tiedostoille ja niissä oleville luokille ja metodeille. Testejä on 17, ja ne testaavat luokkien toimintaa, sekä erilaisia mahdollisia poikkeuksia ja niistä nousevia virheitä.

Ohjelman tunnetut puutteet ja viat

Tietyissä tilanteissa käyttäjälle näytettyjen virheviestin sisältö ei ole aivan täydellinen. Erityisesti jos mutaation johdosta RNA:ssa ei ole ollenkaan eksonia, ei käyttäjää tarvitsisi pyytää antamaan uutta tiedostoa. Annetusta virhetekstistä ilmenee silti syy.

Introneja haetaan ohjelmassa niin, että alkumerkkiä etsitään kodonien alusta ja lopetusta kodonien lopusta. Tämä on tavallaan puute mikäli intronin pitäisi pystyä alkamaan kodonin lopusta. Tämä oli kuitenkin myös suunnitteluvalinta, joka perustui siihen, että tiedostoista generoidut eksonit jäivät helposti hyvin lyhyiksi muuten.

Parhaat ja heikoimmat kohdat

Mielestäni simulaatio on kaunis ja värit harkittuja. Tekstien paikkojen, muotojen ja värien hiomiseen on käytetty aikaa ja ajatusta. Ohjelma pitäisi olla myös vaikea kaataa laajan virhetarkastelun vuoksi. Käyttäjä saa myös ilmoituksen virheestä aloitusikkunan yhteydessä sekä mahdollisuuden antaa uuden tiedoston ohjelmalle. Olen myös erityisen ylpeä peptidiluokasta. Mielestäni kyseisen luokan koodi on yksinkertainen, mutta elegantti.

RNA:n piirtäminen intronеineen ja mutaatioineen voisi varmasti olla kirjoitettu kompaktimpaan muotoon. Tällä hetkellä koodissa on paljon toistoa ja se on tarpeettoman pitkä. Ohjelman ajaminen main-tiedostosta protein_synthesis-tiedoston sijaan on ehkä turhaa. Tarkoituksena oli tehdä ohjelman ajamisesta selkeämpää erillisen main-tiedoston avulla.

Poikkeamat suunnitelmasta

Ohjelma vastaa mielestäni suunnitelmaa hyvin, tosin suunnitelma oli myös alustava eikä erityisen tarkka. Pieniä muutoksia alkuperäiseen verrattuna olivat esimerkiksi ohjelman pyörittäminen main-tiedoston kautta protein_synthesis-tiedoston sijaan ja peptidiketjun luominen kokonaisuudessaan Peptide-luokassa. Tällaiset päätökset muodostuivat ohjelmaa tehdessä ja tuntuivat luontevammille. Ohjelman rakenne ja toiminta ovat suunnitelman mukaiset.

Suunnitelman aika-arvio vastasi pääosin hyvin todellisuutta. Saatoin käyttää hieman enemmän aikaa luokkien toiminnallisuuksien miettimiseen kuin mitä alkuperäisen arvio mukaan suunnittelin. Toteutusjärjestys sen sijaan poikkesi suunnitellusta, mikä pääosin johtui epäselvyyksistä ohjelman biologisen toteutuksen suhteen.

Toteutunut työjärjestys ja aikataulu

Rakensin ensin luokat pääpiirteittäin ja siirryin sitten käyttöliittymän toteutukseen. Luokkien rakentaminen tapahtui viikolla 16 ja käyttöliittymä viikolla 17. Kun käyttöliittymä oli suurimmilta osin tehtynä, täydensin puuttuvat osat luokkatoiminnallisuuksista ja ihan lopuksi tein yksikkötestit. Viimeisten osuuksien toteutus tapahtui viikolla 18. Alun perin olin suunnitellut tekeväni ensin luokkien toteutuksen ja sen perään yksikkötestit ja lopuksi käyttöliittymän. Toteutuksen järjestys siis vaihtui hieman.

Arvio lopputuloksesta

Projektissa luotiin ohjelma, joka simuloi proteiinisynteesiä ja visualisoi sen. Ohjelmassa toimintalogiikka ja käyttöliittymä ovat erotettu toisistaan erillisiin tiedostoihin ja luokkiin. Toiminnallisuus on lisäksi jaettu pienempiin luokkiin niin, että yksi luokka vastaa aina yhdestä biologisesta komponentista, minkä lisäksi käyttöliittymää koordinoi UI-luokka ja sovelluslogiikka koordinoi ProteinSynthesis-luokka.

Olen jakoon tyytyväinen ja se vaikutti toimivan hyvin. Lisäksi se mahdollistaisi ohjelman laajentamisen koskemaan pidempää DNA-juostetta, jossa on useampia eksoneita ja introneita kohtalaisen pienellä vaivalla. Tämän voisi toteuttaa esimerkiksi säilyttämällä RNA-olioita listassa DNA:ssa ja tekemällä pieniä muutoksia UI-luokkaan, jotta kaikki RNA-juosteet tulisivat piirretyksi.

Samassa yhteydessä saattaisi olla järkevää siirtää RNA vastaamaan siihen liittyvästä peptidistä. Yleisesti laajennettavuuteen ja käytettävyyteen vaikuttaa se, että luokat valittiin sekä käyttöliittymässä, että logiikassa vastaamaan prosessin biologisia komponentteja, mikä toimi hyvin.

Ulkonäöstä tuli mielestäni myös hieno ja näin sen ja värien eteen paljon vaivaa. Intronit, eksonit ja mutaatiot on helppo tunnistaa ja värit toimivat mukavasti yhteen. Piirtämiseen liittyvää koodia olisi silti voinut hieman yksinkertaistaa erityisesti RNA:n piirtämisen osalta. Muotoihin ja väreihin liittyvät konfiguroinnit ovat omissa luokissaan, joten niiden muuttaminen on myös selkeää.

Olen myös tyytyväinen virhetarkasteluun ja siihen, miten virheet näytetään käyttäjälle. Erilaiset virheet antavat mielestäni hyviä ohjeita käyttäjälle mikä meni vikaan ja käyttäjällä on mahdollisuus antaa uusi tiedosto käynnistämättä ohjelmaa uudestaan. Yksikkötestauksesta tuli myös melko laaja ja se auttoi huomaamaan muutamia virheitä koodissa. Pytestin käyttäminen siinä osoittautui käteväksi, koska sillä sai ajettua kaikki testit nopeasti muutosten jälkeen ja näki menikö jotain rikki.

Satunnaisgeneroitujen tiedostojen sisältö oli todella satunnainen ja välillä hyvin kaukana siitä, millainen oikea DNA-juoste olisi. Tämä aiheutti hieman pohtimista ja vaikutti moneen suunnittelupäätökseen, jotta melko satunnaisesta syötteestä sai kohtalaisen järkevän näköisiä lopputuloksia.

Yleisesti ohjelmasta tuli mielestäni oikein hyvä.

Viitteet

Qt dokumentaatio, osoitteesta: https://doc.qt.io/qt-5/index.html

Python dokumentaatio, osoitteesta: https://docs.python.org/3/

Pytest dokumentaatio, osoitteesta: https://docs.pytest.org/en/6.2.x/

Taulukko RGB-väreistä, osoitteesta: https://www.rapidtables.com/web/color/RGB_Color.html

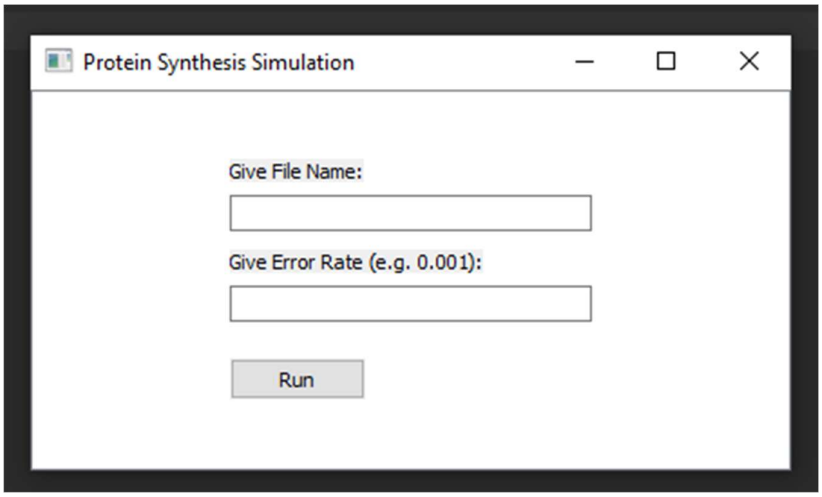
B. Alberts, A. Johnson, J. Lewis, and M. Raff 2015: Molecular, Biology of the Cell (6th Edition). Garland Science, New York.

DNA-sekvenssit, osoitteesta: https://www.bioinformatics.org/sms2/random_dna.html

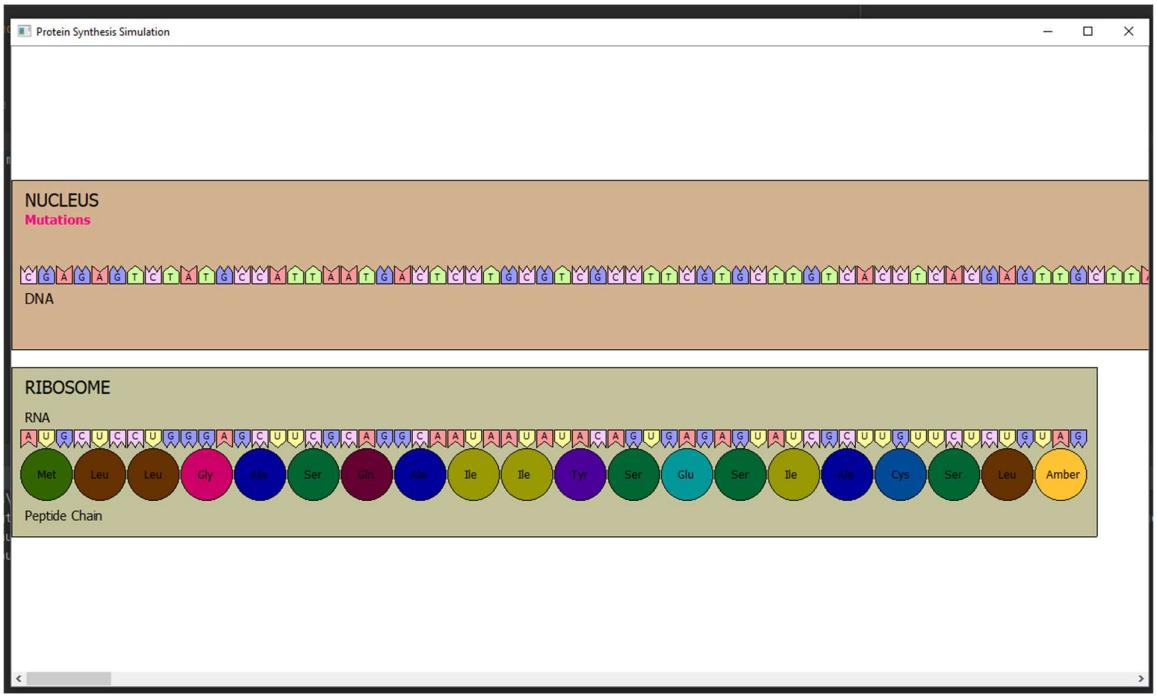
Aminohappo-taulukko, osoitteesta: https://en.wikipedia.org/wiki/Start_codon#ref_startA

Liitteet

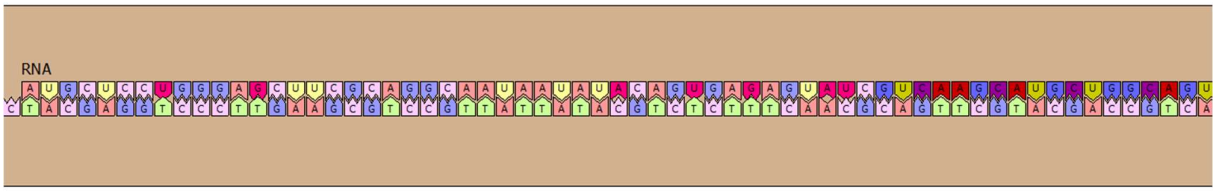
Liite 1.



Liite 2.



Liite 3.



Liite 4.