Projektityön dokumentti

CS-A1121 - Ohjelmoinnin peruskurssi Y2
Iida Lähdekorpi 712495
Bioinformaatioteknologia

6.5.2020

1 Yleiskuvaus

Ohjelmakirjasto, jolla voi visualisoida numeerista dataa graafisesti. Grafiikkakirjaston avulla voidaan esittää ainakin seuraavanlaisia kuvaajia: perusviivadiagrammi (1-n viivaa, joiden pisteet määräytyvät datan (x,y) koordinaattien perusteella. Saman käyrän erillisiä pisteitä yhdistää suorat viivat).
Ohjelma pystyy mallintamaan valmiiksi annettua numeerista dataa ja ohjelma lukee datan käyttäjän nimeämästä tiedostosta.
Kuvaajassa näkyy kuvaajan nimi, sekä jokin selite eri käytetyille väreille. Käyttäjä pystyy itse nimeämään myös asteikot sekä mahdolliset akselit. Koordinaattiakselien numerointi on oikealla välillä ja numerot eivät piirrettäessä peitä toisiaan.
Käyttäjällä on mahdollisuus lisätä kuvaajan taakse ns. gridi. Gridi on heikosti erottuvalla katkoviivalla piirretty ruudukko (ruudun koon voi asettaa), jonka avulla kuvaajasta on helpompi lukea datan arvoja. Esim. pylvään korkeutta vastaava arvo on helpompi lukea kuvaajasta gridin avulla.
Projektini on toteutettu keskivaikealla vaikeustasolla, eli sillä pystyy mallintamaan vain viivadiagrammeja. Ohjelma on kuitenkin laajennettavissa mahdollisia muita visualisointitapoja varten.
Käyttäjä pystyy ohjelmassa myös itse säätämään akselien sekä taustan väriä. Viivojen värit ovat satunnaisia, mutta aina eri. Päätin toteuttaa laajemman visuaalisen kustomoinnin, koska se kuulosti mielestäni mielekkäältä ja käyttäjäystävälliseltä vaihtoehdolta. Näiden ominaisuuksien avulla pystyy havainnollistamaan dataa tehokkaammin.

2 Käyttöohje

Ohjelma käynnistetään ajamalla Program-kansiossa sijaitseva main.py -tiedosto.
Käyttäjällä on tiedosto, jossa on kuvaajan viivojen koordinaatit halutussa formaatissa, eli yksi koordinaatti rivillä muotoa 'x,y,ryhmä', jossa ryhmä ID kuvaa aina tiettyä viivaa. Saman viivan kaikki pisteet tulee olla peräkkäin.
Käyttäjä syöttää tiedoston nimen ja sen jälkeen halutun kuvaajan muodon vaihtoehdoista line/pie/column Ohjelma piirtää akselit, kuvaajan, sekä kuvaajan

groupin. Akselien numerointi on halutulla välillä ja se näyttää akseleilla piirrettyjen pisteiden arvot. Ohjelma myös järjestää datan x-koordinaatin mukaan pienemmästä suurempaan.
Ohjelmassa on oletusarvoisesti takana 30 kokoinen gridi, sekä vaalea tausta ja mustat akselit. Käyttäjä saa toolbarin ansiosta otettua halutessaan gridin pois, säätää gridin kokoa sekä asettaa taustavärin. Käyttäjä pystyy myös halutessaan vaihtaa akselien, sekä akselien nimien väriä sekä nimetä akselit uudelleen. Akselien niminä on oletusarvoisesti 'x' sekä 'y'.

3 Ulkoiset kirjastot

Olen käyttänyt ohjelmassani PyQt5 -kirjastosta QtCorea, QtGuita sekä QtWidgetsiä. Sen lisäksi ohjelmassa käytetään Pythonin random-kirjastoa, jotta viivoille saadaan satunnainen väri.

4 Ohjelman rakenne

Ohjelma piirtää kuvaajan ja viivat QPainterin avulla QMainwWindowiin. Päätin toteuttaa ohjelman tällä tavalla, sillä se näytti kurssin esimerkkien mukaan olevan yksinkertainen ja hyvä tapa toteuttaa piirtävä ohjelma.

Ohjelmassa on kaksi osaa, tiedoston lukeva osa ja tietoja säilövät rakenteet sekä GUI-luokka. Ohjelmassa on myös testiluokka, sekä main.py tiedosto joka luo PyQt applikaation ja käynnistää sen.

4.1 read_file_IO

Tiedosto sisältää luokan ReadFile, missä on metodit load_coordinates, load_columns sekä load_piechart. Tässä projektissa vain load_coordinates toimii, sillä ohjelmalla pystyy luomaan vain viivadiagrammeja. Muut metodit ovat helposti lisättävissä, jos haluaa ohjelman luovan erityyppisiä diagrammeja.
Tiedoston lukeva osa lukee tiedostoa rivi kerrallaan ja ottaa talteen jokaiselta riviltä pisteen x- ja y-koordinaatin sekä pisteen group-tunnisteen. Funktio tallentaa luetut tiedot haluttuihin olioihin. X- ja y-koordinaattien perusteella luodaan Coordinates-olio, joka tallennetaan Points-olioon, missä on myös group tunniste.
Linegraph() luokka sisältää sanakirjan, jossa avaimena on group-tunniste ja arvona points-olio. Luokassa on myös metodeja, jonka avulla saadaan viivojen maksimi- ja minimiarvot sekä vaihteluväli.

4.2 GUI

GUI-luokka luo käyttöliittymän ja piirtää taustan, gridin, akselit ja kuvaajan. Sen tärkeimmät metodit ovat:

init
Alustaa QMainWindowin ja luo halutun Lineplot() olion käsittelyä varten. Alustaa tärkeimmät ominaisuudet graafissa.

paintEvent
Alustaa painterin ja kutsuu tärkeimpiä piirtofuktioita.

draw_grid
Piirtää kuvaajan taakse gridin, jos boolean-muuttuja self.gridin arvo on True. Gridi koostuu painter.DrawRect piirretyistä neliöistä, jonka koon käyttäjä saa asettaa.

draw_axis
Piirtää kuvaajaan x- ja y- akselin, sekä niiden selitteet. Piirtää myös vastaavat arvot akselien viivoille sekä nollakohdat. (Origon)

draw_lines
Piirtää viivan koordinaattien ja skaalauksen perusteella kuvaajaan. Luo myös uuden listan, jonka avulla voidaan piirtää niitä vastaavat arvot x- ja y-akselille.

randomize_color
Lisää listaan aina yhtä monta väriä kuinka monta viivaa ollaan piirtämässä. Värin r, g ja b arvot tulevat random toiminnon avulla.

set_toolBar
Luo uuden toolBarin, joka sisältää seuraavat ominaisuudet:
Grid: Antaa käyttäjän ottaa gridin pois klikkauksella
GridSize: Antaa käyttäjän asettaa gridin ruudulle uuden koon
Labels: Antaa käyttäjän asettaa x- ja y-akseleille uudet nimet
ShowNumbers: Antaa käyttäjän ottaa akselien numerot pois näkyvistä klikkauksella
LineWidth: Antaa käyttäjän asettaa viivojen paksuuden
BackgroundColor: Antaa käyttäjän asettaa ikkunan taustavärin
AxisColor: Antaa käyttäjän asettaa akselien värin
Exit: Sulkee ohjelman

5 Algoritmit

Viivadiagrammin piirtämisessä täytyi viivat skaalata oikein kuvaajaan. PyQt QPainterin 0,0 kohta on vasemmassa yläkulmassa, joten sitä piti muuttaa. Draw_lines -funktiossa on skaalattu jokainen piste arvolla a joka on

$$a = (piste - minimipiste)/vaihteluväli$$

Lisäksi a kerrotaan niin, että kuvaajaan jää vasempaan reunaan n. 5% marginaali, oikeaan reunaan 10%, ylös 20% (jotta myös toolBar mahtuisi) sekä alas 5%.

Akselit ovat skaalattu kuvaajan reunoille niin, että vasempaan reunaan jää noin 5% ja alas myös 5%.

Lineplot-luokassa on metodeja, jotka etsivät viivadiagrammin maksimi- ja minimikohdat. Nämä on toteutettu etsimällä jokaisen groupin maksimeista/minimeistä maksimi/minimi.

6 Tietorakenteet

Ohjelman tietorakenteet ovat kokonaisuudessaan melko yksinkertaisia, eikä siinä käytetä muuttumattomia tietorakenteita.
Points-olio säilöö pisteet Python listana, Linegraph käyttää sanakirjaa.
Nämä tietorakenteet mahdollistavat datan helpon säilytyksen ja siirrettävyyden tiedostosta toiseen helposti. Sanakirja valikoitui parhaaksi ratkaisuksi säilöä viivoja sen muokattavuuden ja helppokäyttöisyyden ansiosta. Myöhemmin GUI-luokkaa tehdessä ratkaisu osoittautui varsin hyväksi ja oikeaksi.

7 Tiedostot

Ohjelmalle annetaan tiedosto, jossa haluttu data sijaitsee. Tiedoston tulee olla muotoa .txt, jotta ohjelma avaa sen. Tiedoston, jossa viivadiagrammille annettu data on, tulee sisältää data tietyssä muodossa. Datan sisältää viivadiagrammin eri pisteet eri riveillä, niin että yhdellä rivillä on yhden koordinaatin tiedot sekä tieto siitä, mihin viivaan piste kuuluu. Rivin tiedot on eroteltu pilkuilla, jossa ensimmäinen osa kertoo x-koordinaatin, toinen osa y-koordinaatin ja kolmas osa ryhmän eli viivan johon piste kuuluu. Samassa tiedostossa voi olla usean eri viivan tietoja, mutta saman viivan kaikki pisteet tulee olla peräkkäin.
Projektia suunniteltaessa päädyin tähän ratkaisuun, sillä se on yksinkertainen ja helposti luettava formaatti. Lisäksi .txt-tiedosto on helppo luoda ja sitä on helppo muokata.

Alla esimerkki tiedoston rivistä, joka on oikeaa formaattia ohjelmalle.

```
1,1,esim
```

8 Testaus

Ohjelmassa on test-luokka joka painottuu lähinnä tiedostoa lukevaan osaan ja tietorakenteisiin. Ohjelmasta löytyy erillinen test.py tiedosto, joka hoitaa yksikkötestauksen. Testausta pystyi suorittamaan luomalla eri määrän erilaisia testitiedostoja joissa oli erilaista dataa, osa virheellistä ja osa kelpaavaa. Ohjelma tuntui kelpuuttavan kaikki yksikkötestit.
Ohjelmaa pysty myös testaamaan visuaalisesti, katsomalla piirtääkö se oikein. Se olikin hyvä tapa testata GUI-luokkaa ja nähdä missä on vielä virheitä. Esimerkiksi viivojen ja akseleiden skaalauksia tehdessä tämä oli varsin toimiva tapa testata oikeita skaalausarvoja.

9 Ohjelman tunnetut puutteet ja viat

Ohjelma on hidas reagoimaan klikkauksiin ja näytön muokkaamiseen. Tämän olisi voinut välttää tekemällä jotain tehokkaampia ratkaisuja. Gridi ei ole synkronoitu akseleihin ja kuvaajiin, vaan on itsenäinen näistä molemmista. Tämä olisi ollut muokattavissa käyttämällä erilaista skaalausta, tai luomalla gridi viivojen avulla. Yksittäisen viivan paksuutta ei saa muokattua, mikä olisi ollut hyödyllistä, jos haluaisi korostaa tiettyä kuvaajaa. Myöskään tietyn viivan väriä ei saa itse muokattua. Minimiarvojen asettelu on hieman epäselkeä, sillä en oikein löytänyt hyvää tapaa asetella niitä.

10 3 parasta ja 3 heikointa kohtaa

10.1 Parhaat

Viivojen piirto

Viivat piirtyvät ikkunaan täydellisesti skaalattuina ja muokkatuvat ikkunan pituuden ja leveyden mukaan. Tämä oli henkilökohtaisesti yksi hankalimmista kohdista tehdä, mutta myös erittäin palkitseva kun oikean kaavan sai ratkaistua.

Visuaalinen kustomointi

Ikkuna on visuaalisesti miellyttävä ja helppokäyttöinen. Ohjelmassa käyttäjä pystyy muokkaamaan taustaväriä, akseleiden väriä, viivojen paksuutta sekä nimeämään akselit uudestaan. Ikkunan kokoa muokkaamalla myös kuvaajan koko muokkautuu, eikä ole sidottu vain yhteen kokoon. Käyttäjä pystyy myös halutessaan ottamaan akseleissa näkyvät numerot pois, joka on hyödyllinen toiminto varsinkin jos käsitellään suurta määrää dataa.

Tietorakenteet

Tiedostosta luettu tieto on säilötty selkeästi ja helposti käsiteltävässä muodossa. Tietorakenteita käsittelevissä luokissa on selkeät ja tärkeät metodit maksimi- ja minimikohtien löytämiseen sekä vaihteluväliin.

10.2 Heikoimmat

Hitaus

Ohjelma ei luultavastikaan toimi parhaiten mahdollisesti, sillä se on melko hidas ainakin itse sitä testatessa. Tämä olisi ollut ehkä korjattavissa, mutta huomasin sen melko myöhäisessä vaiheessa ja en löytänyt siihen oikein kunnolla ratkaisua.

Yhden viivan muokkaaminen

Vaikka suurin osa visuaalisista muokkausmahdollisuuksista ovat erittäin hyviä, ohjelmalla ei pysty tekemään tiettyjä hyödyllisiä muokkauksia. Esimerkiksi yhden viivan muokkaaminen monen viivan diagrammissa. Joissakin diagrammeissa olisi hyödyllistä vaihtaa viivan väriä tai sen paksuutta sen korostamiseksi, mutta tässä ohjelmassa se ei onnistu. Ongelman olisi voinut korjata esimerkiksi draw_lines -funktiossa.

Gridi

Gridi toimii mielestäni hyvin ja helpottaa datan lukemista. Sen kuitenkin pitäisi

olla yhteydessä akseleihin ja kuvaajaan, toisin kuin ohjelmassani. Tämän olisi voinut korjata käyttämällä erilaista skaalausta. Päätin kuitenkin panostaa projektissani muihin asioihin sen sijaan että olisin korjannut asian.

11 Poikkeamat suunnitelmasta

Pysyin suunnitelmassani melko hyvin. Tein asioita suunnittelemassani järjestyksessä, vaikka välillä toiminkin melko impulsiivisesti ja muokkailin asioita mitä tuli satunnaisesti mieleeni. Lisäsin vain asioita suunnitelmaani, kuten esimerkiksi Linegraph() luokan. Koin sen hyödylliseksi projektia tehdessä ja se osoittautui hyväksi valinnaksi. Käytin ehkä hieman enemmän aikaa GUI-luokan tekemiseen, kuin mitä olin arvioinut.

12 Toteutunut työjärjestys ja aikataulu

Suunnitelman mukaan, aloitin tiedostoa lukevasta osasta ja pisteitä, viivoja ja diagrammia säilövistä luokista ja lopetin GUI-luokkaan. Välissä tein testejä ja etsin paljon tietoa aiheesta. Arvioinkin aluksi, että GUI-luokan tekeminen tulee olemaan haastavaa, mutta se olikin vielä haastavampaa kuin ajattelin, joten siihen meni reilusti eniten aikaa ja jätin sen tekemisen liian myöhäiseksi. Olisin voinut päivittää gittiä enemmän, jotta todellinen aikataulu olisi myös itselleni selkeämpi, mutta työkalu on minulle aika uusi. Tässä kuitenkin commithistorian perusteella luotu aikataulu:

Aikataulu
16.3.2020 - Aloitin tekemään projektia ja loin ohjelman keskeisimmät luokat sekä git-repositoryn.
2.4.2020 - Read_File -luokka oli jo lähes valmis (ainakin niin luulin), samoin tietorakenteita käsittelevät luokat ja aloin tekemään GUI-luokkaa.
22.4.2020 - Ohjelmaan oli tehty paljon paranteluja ja korjauksia, akselien ja gridin piirtäminen onnistui jo hyvin.
5.5.2020 - Projekti jo viimeistelyjä vaille valmis, viivan piirto ominaisuus toimii osittain ja ohjelma kaipaa vain loppusilausta.
8.2.2020 - Projektin deadline, projekti toimii mielestäni todella hyvin ja tekee sen mitä pitääkin. Projektidokumentti valmis.

Ajankäyttö (arvio / toteuma tunteina)
Suunnittelu (5-10 / 10)
Rakenteiden ja metodien tekemistä ja tekstitiedoston käsittelevä osa (10-20 / 20)
Käyttöliittymän luominen, GUI (20-30 / 30)
Testausta ja ohjelman hiomista loppuun (20 / 20)
Loppuraportti ja loppudemo (5-10 / 10)
Yhteensä: n. 90h

Oletin, että suoriutuisin projektista ajankäytön arvion alarajoilla, mutta kaikki menivät ylärajoille. Se on ihan hyvä, sillä panostin projektiin todella ja tein asiat huolella. Tosin olisin voinut säästää aikaa tekemällä vielä perusteellisemman suunnitelman miten aion asiat tehdä ja aikatauluttamalla henkilökohtaista elämääni paremmin.

13 Arvio lopputuloksesta

Omasta mielestäni ohjelmani täyttää tehtävänannon hyvin. Onnistuin tekemään toimivan ohjelman, jolla pystyy visualisoimaan numeerista dataa viivadiagrammin muodossa käyttäen PyQt-kirjastoa. Samalla sain ohjelmasta yksilöidyn, sillä siinä on laajempi visuaalinen muokattavuus. Pysyin suunnitelmassani hyvin pieniä muutoksia lukuunottamatta ja palautin työn ajoissa. Toisaalta ohjelma olisi voinut olla laajemmin kustomoitavissa ja se voisi toimia tehokkaammin. Myöhemmin sitä tehdessäni tajusin, että sen laajentaminen tekemään muunlaisia kuvaajia olisi ollut myös erittäin tehtävissä, joten vaikea taso ei ollutkaan niin kaukana saavuttaa.

14 Viitteet

http://zetcode.com/gui/pyqt5/introduction/
https://www.qt.io
A+ oppimateriaali ja harjoitustehtävät
https://docs.python.org/3/tutorial/index.html
https://www.wikipedia.org/
https://www.stackoverflow.com/

15 Liitteet

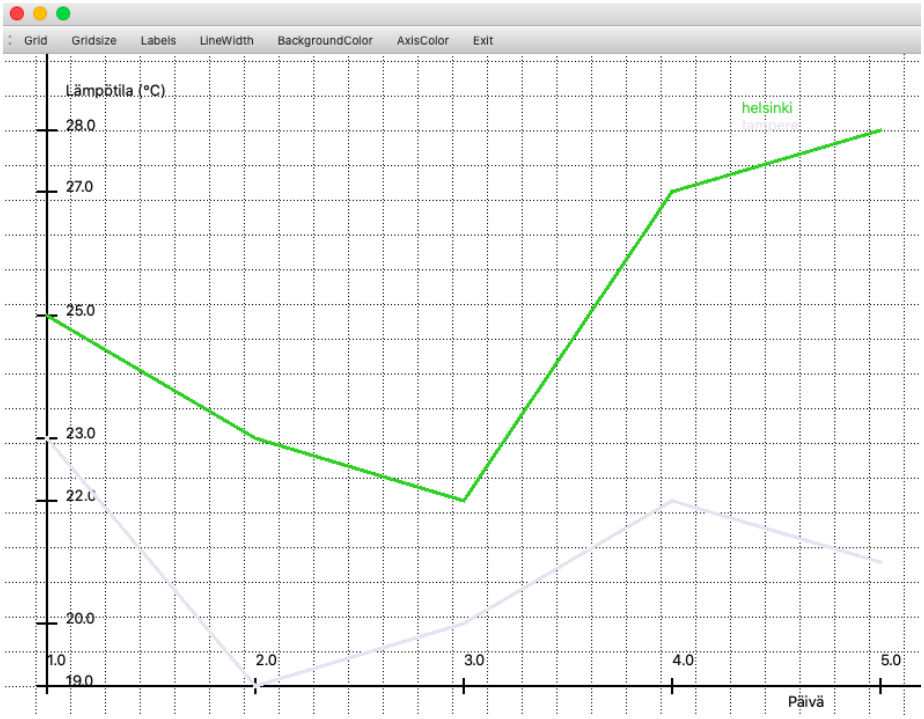
```
1,25,helsinki
2,23,helsinki
3,22,helsinki
4,27,helsinki
5,28,helsinki
1,23,tampere
2,19,tampere
3,20,tampere
4,22,tampere
5,21,tampere
```

Esimerkki ohjelmaan syötettävästä line-tyyppisestä -txt tiedostosta. Tiedosto sisältää dataa kahden eri kaupungin lämpötiloista eri päivinä.

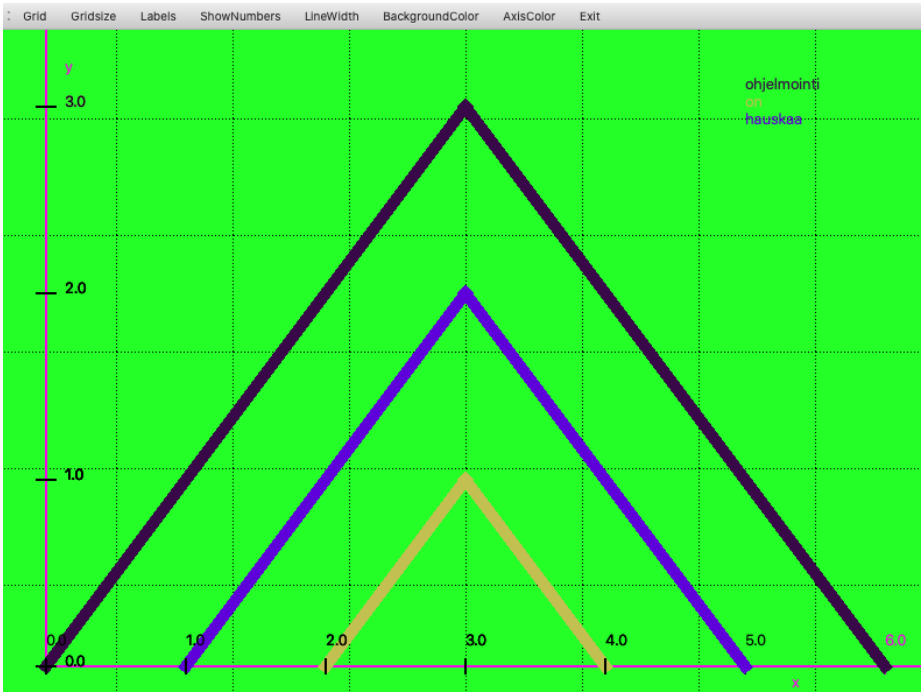
```
bash-3.2$ python3 main.py
Insert filename:
esim3.txt
Insert graph type (line/pie/column):
line
```

Ohjelma ajetaan komentoriviltä näin.

Tiedostossa oleva data piirtää alla olevan graafin:



Graafissa olen muokannut x- ja y-akselien selitteet oikein.



Toinen esimerkki jossa olen hyödyntänyt ohjelman visuaalista kustomointia.