2d-pistejoukon klusterointi k‑keskiarvo -algoritmilla

Tietorakenteet ja algoritmit II  
Ohjaaja: Tomi Suomi

Laatijat:   
Kimmo Kettunen (68220) kmpket@utu.fi  
Timi Suominen (505890) tijusuo@utu.fi

x.x.2014  
Turku

Table of Contents

Tehtävän kuvaus ja analysointi 3

Ratkaisuperiaate 3

Ohjelman ja sen osien kuvaaminen (lähinnä moduulien kuvaus kohdasta 3.5) 3

Testausjärjestely 4

Liitteet: 4

# Tehtävän kuvaus ja analysointi

Tehtävänä oli ryhmitellä 2d-pistejoukon pisteet merkityksellisiin luokkiin (myöhemmin klustereihin). Tarkasteltavaksi algoritmiksi valitsimme yleisesti käytetyn k‑keskiarvot (k-means).

Toteutimme algoritmin kirjastofunktiona, jota kutsutaan syöttämällä sille halutut pisteet, klustereiden lukumäärä ja iteraatioiden maksimimäärä. Funktio palauttaa klustereiden keskipisteet ja liittää syötettyihin pisteisiin sen klusterin numeron, johon se kuuluu.

Tehtävän kuvauksessa selostetaan annettu tehtävä ja ohjelman käyttämät syöttö- ja tulostiedot.

Tämän lisäksi kuvaus kattaa myös muut tehtävänantoon liittyvät seikat, kuten alkuperäiseen tehtävään tehdyt tarkennukset ja muutokset sekä tiedot ohjelmiston rajoituksista ja pohdintaa sen mahdollisista laajennuksista.

Ainoa varsinainen käytön rajoitus on se, että klustereiden määrä pitää olla pienempi, kun syötteenä annettujen pisteiden määrä. Jatkokehitys voisi olla esimerkiksi Piste-luokan muuttaminen rajapinta-luokaksi ja joidenkin toiminnallisuuksien siirtäminen siihen; kuten pisteiden välisen etäisyyden laskeminen ja pisteiden yhdistäminen. Näin algoritmi toimisi myös useampi ulotteisille pisteille ja/tai eri etäisyyden määrittely tavoille.

# Ratkaisuperiaate

K-keskiarvo algoritmissa pisteet klusteroidaan k määrään klustereita pisteiden keskinäisen etäisyyden perusteella. Tämä tapahtuu kahdessa vaiheessa, joita toistetaan kunnes lopputulos ei enää muutu tai on suoritettu haluttu määrä iterointeja. Ensimmäisessä nk. sijoitusvaiheessa pisteet sijoitetaan kukin siihen klusteriin, jonka keskipiste on pistettä lähinnä. Toisessa nk. päivitysvaiheessa, klusterin keskipiste lasketaan uudelleen siihen liitettyjen pisteiden arvojen perusteella.

K-keskiarvot-algoritmin pseudokoodi:

**Syöte**: S (pisteet), k (klustereiden lukumäärä)

**Tuloste**: klusterit

1: Alusta k määrä klusterien keskipisteitä

2: **while** lopetusehto epätosi

3: Sijoita pisteet lähimpään klusteriin

4: Päivitä klustereiden keskipisteet sijoitusten perusteella

5: **end** **while**

Algoritmille annetaan syötetietoina klusteroitavat pisteet, haluttu klustereiden määrä ja iteraatioiden maksimimäärä. Loimme pisteille oman luokan Piste, jolla on attribuutteina x- ja y-koordinaatit sekä klusterin numero, johon se tulee kuulumaan.

Aluksi pisteiden joukosta valitaan satunnaisesti k kappaletta ensimmäisiksi klustereiden keskipisteiksi. Seuraavaksi näitä lähdetään iteroimalla tarkentamaan. Jokaiselle pisteelle lasketaan etäisyys eri klustereihin ja piste sijoitetaan sitä lähinnä olevaan klusteriin. Seuraavaksi lasketaan kuhunkin klusteriin kuuluvien pisteiden perusteella klusterin uusi keskipiste. Koska klusterien koordinaatit ovat (lähes) aina approksimaatioita eikä tarkkaan tulokseen siten päästä, on iteraatiokerrat rajoitettava johonkin mielekkääseen määrään. Iteraatiokerralla tässä tarkoitetaan sitä operaatiota, joka tarkentaa klusteriden sijaintia 2d-pistejoukossa. Iteraation loputtua palautetaan klustereiden keskipisteet ja syötettyihin pisteisiin on tallennettu tieto siitä, mihin klusteriin kukin piste kuuluu.

Lukija johdatellaan tehtävän ratkaisuun esittelemällä yhteenveto ratkaisun kulmakivistä ja peruskäsitteistöstä (ts. tehtävän kannalta oleelliset asiat selostetaan tiivistetysti käyttäen apuna esim. kuvia ja kaavioita).

Dokumentoitavia pääpiirteitä ovat mm. syöttö- ja tulostietojen käyttö, yleiset ratkaisuperiaatteet ja -kaavat, sovelletut algoritmit yms. Sanallisessa selostuksessa on vältettävä tarpeettomia yksityiskohtia kuten laitteisto-, ympäristö- tai ohjelmointikielikohtaisia termejä. Tavoitteena on laatia teksti siten, että sen läpi lukemalla käy ilmi, onko työn tekijä ymmärtänyt tehtävänannon oikein ja ratkaissut sen mielekkäästi.

# Ohjelman ja sen osien kuvaaminen

## Luokkien kuvaus

### class Piste

**Vakiot ja attribuutit:**

double X

Pisteen x-koordinaatti.

double Y

Pisteen y-koordinaatti.

int Group

Klusterin numero

### class Kmeans2d

**Julkiset metodit:**

GetKmeans2d(List<Piste> pisteet, int k, int maxIter)

Parametri pisteet sisältää listauksen kaikista pisteistä. Klustereiden kokonaismäärä k puolestaan on ennalta määrätty määrä klustereita. Iteraatioiden maksimimäärä maxIter puolestaan on ohjelmallinen rajoite laskentakierroksille.

Metodi palauttaa suorituksen päätyttyä keskipisteiden tarkennettuine sijaintineen.

Virhetilanteissa heitetään exception eli poikkeus. Poikkeus heitetään tilanteissa, joissa pisteitä on vähemmän kuin k:kpl.

Sivuvaikutuksena pisteet liitetään klustereihin.

**Yksityiset metodit:**

LiitaLahimpaanKeskukseen(List<Piste> pisteet, List<Piste> keskipisteet)

Päivittää kunkin syötteenä saadun pisteen (pisteet) kuulumaan lähimpään klusteriin sen etäisyyden perusteella eri klustereiden keskipisteistä (keskipisteet). Lähin klusteri etsitään LahinKeskus()-funktion avulla.

Palauttaa true, jos yksikin piste on vaihtanut klusteria, false muulloin.

Sivuvaikutuksena pisteen Group-attribuutti muuttuu, jos sen uusi klusteri on eri kuin edellinen.

LahinKeskus(Piste vertailupiste, List<Piste> keskipisteet)

Palauttaa attribuuttina annetun pisteen (vertailupiste) lahinna olevan klusterin numeron.

Etäisyys lasketaan käyttäen apuna EtaisyydenNelio(Piste a, Piste b) -funktiota.

EtaisyydenNelio(Piste a, Piste b)

Palauttaa kahden parametreina saadun pisteen (a ja b) välisen euklidisen etäisyyden.

PaivitaKeskipisteenSijainti(int k, List<Piste> pisteet)

Metodi laskee klustereiden uusien keskipisteiden sijainnit. Parametreina annetaan klusterien lukumäärä k kaikki klusteroitavat pisteet (pisteet).

Metodi laskee uuden keskipisteen sijainnin laskemalla ensiksi klusteriin liitettyjen pisteiden kehyksen, eli kaikkien pisteiden koordinaattien yhteenlasketun leveyden ja korkeuden. Lopuksi tämän kehyksen keskipiste lasketaan ja siitä tulee uusi klusterin keskipiste.

**Moduulien kuvaus**  Jokainen moduuli kuvataan omassa aliluvussaan selkeästi ja paneutumatta teknisiin yksityiskohtiin (ts. kuvaus ei sisällä suoraa ohjelmakoodia, vaan toteutukselliset seikat dokumentoidaan ohjelmalistaukseen). Moduulin kuvaus jakautuu julkiseen ja implementointiin liittyvään rajapintaan, jotka ryhmitellään tarkoituksenmukaisella tavalla. Esimerkiksi metodit avaaTiedosto(), lueTiedostosta(), kirjoitaTiedostoon() ja suljeTiedosto() voidaan dokumentoida yhdessä tiedostonkäsittelyyn liittyvinä piirteinä. Samoin voidaan menetellä esimerkiksi tietorakenteiden käsittelyn dokumentoinnissa.

Moduulin rajapintojen määrittely perustuu sen proseduurien ja funktioiden (joita kutsutaan yhteisnimellä rutiini tai metodi) rajapintoihin. Rutiineista (myös mahdollisista luonti- ja alustusrutiineista) dokumentoidaan niiden *otsake* (eli rutiinin nimi, parametrien nimet ja tyypit sekä mahdollisen paluutuloksen tyyppi), ytimekäs *sanallinen kuvaus* siitä mitä rutiini tekee, *parametrien rajoitukset*, *paluutuloksen tulkinta*, *virhetilanteisiin reagointi* (esim. poikkeusten nosto), ja *syntyneet muutokset* (ts. sivuvaikutukset) tietorakenteisiin, syöttöön ja tulostukseen. Tämän lisäksi kuvataan attribuuttien ja vakioiden tarkoitus luokassa.

Rutiinit jotka liittyvät mallin ohjaukseen (mm. juurimoduulissa oleva pääohjelma) tai ovat tehtävän ratkaisun kannalta oleellisia voidaan tarvittaessa kuvata algoritmisesti esimerkiksi pseudokoodilla.

Käyttöliittymän ohjelmallinen toteutus dokumentoidaan kuten muutkin ohjelmiston osat. Sen sijaan käyttöliittymän toimintaa käsitellään lähinnä työselostuksen liitteenä olevassa käyttöohjeessa sekä mahdollisesti testien kuvauksessa.

# Testausjärjestely

Testauksen tarkoituksena on varmistaa *empiirisesti*, että ohjelma toimii oikein. Testausjärjestelyn ja testauksessa tehtyjen havaintojen kuvaaminen on olennainen osa harjoitustyön tekemistä; testaamista ei saa koskaan jättää työn ohjaajan tehtäväksi. Testausta voidaan tehdä kahdella tasolla:

* moduulien erillinen testaus: varmistetaan että moduulin sisäinen toteutus vastaa liittymän kuvausta
* ohjelmiston verifiointi: varmistetaan että annettu ongelma ratkaistaan oikein

Moduulien erillinen testaus huolehtii ohjelmiston erillisten rakennekomponenttien eheydestä, jolloin ohjelmiston verifioinnin osaksi jää komponenttien välisten liitosten toimivuus. Testausta ei kannata miettiä jälkikäteen, vaan sitä on suunniteltava jo ohjelmistoa kehitettäessä. Testausta varten kehitetyt osat on syytä jättää myös valmiiseen ohjelmistoon, sillä ne ovat ensiarvoisen tärkeitä ohjelmiston jatkokehityksessä.

Ohjelmiston verifiointia varten tarvitaan tarkasti mietittyjä testiaineistoja, minkä vuoksi testien laatijalla on oltava selvä kuva ongelmasta ja sen ratkaisusta. Tämän tietämyksen pohjalta hän voi suunnitella ytimekkään *testistrategian*, joka sisältää joukon erikoistapauksia ja muutamia perustapauksia. Jotta testaus olisi hallittavissa, testiaineiston tapausten on kohdistuttava niihin ohjelmiston osiin, joissa virheiden esiintyminen on todennäköisintä — kuitenkaan unohtamatta aineiston monipuolisuutta. Yleisenä ohjeena voidaan pitää sitä, että yksi testiaineisto testaa yhden ohjelmisto-osan toimintaa (esim. omat ajot virhesyötteistä toipumisen testaamiseksi, erikoistapauksiin ja normaalisyötteisiin). Testiohjelmien tulosten on oltava helposti tarkastettavissa (ts. testiohjelman pitää *itse* analysoida mahdollimman pitkälle tulosten oikeellisuus). Huolellisesti tehty testiajo (ja -ohjelma) on myös havainnollinen *esimerkki* siitä, miten ohjelmisto toimii kokonaisuutena.

Testiohjelman dokumentointia tärkeämpää on testistrategian selkeä kuvaus. Testiajojen tulostukset liitetään sellaisenaan mukaan dokumenttiin. Tämän lisäksi työselostuksessa esitellään, millaisilla syötteillä ohjelmaa on testattu sekä miten ohjelma niillä toimii.

# Liitteet:

* + alkuperäinen tehtävänanto
  + yksityiskohtaisesti kommentoitu ohjelmalistaus
  + käyttöohje