Oblig 2

1. oktober 2021

Innlevering

Last opp filene dine på Devilry. Innleveringsfristen er fredag 15. oktober 2021, kl. 23:59.

Vi anbefaler så mange som mulig om å samarbeide i små grupper på *opp til tre*. Dere må selv opprette grupper i Devilry, og levere som en gruppe (altså, ikke last opp individuelt hvis dere jobber som en gruppe).

Filene som skal leveres er:

- Én PDF som skal hete `IN2010-oblig2.pdf`
- Et kjørbart Java- eller Python-program som kan kjøres fra en fil som heter `Oblig2.java` eller `oblig2.py` henholdsvis (programmet kan bestå av flere filer)

Filene `movies.tsv` og `actors.tsv` (som beskrives i neste seksjon) skal *ikke* leveres.

For Java skal `main`-metoden være i en klasse som heter `Oblig2`, og være kjørbart med kommandoen `java Oblig2`. For Python skal programmet være kjørbart med `python3 oblig2.py`.

Hver av oppgavene vil be om en utskrift. Utskriftene skal legges inn i main-blokken til programmet. Før utskriften for en oppgave skal det skrives ut `Oppgave i`, der `i` er nummeret på oppgaven.

PDF-en som skal leveres skal inneholde instruksjoner for hvordan man kjører programmet kommentarer til oppgavene. Den skal inneholde hvilke algoritmer som er benyttet, og eventuelt beskrive feil eller mangler. Dere skal også kommentere på den konkrete kjøretiden for programmet. Løsningsforslaget

bruker ca. 5 sekunder på alle oppgavene tilsammen (på faglærers maskin). Siden det er en stor graf vil små forskjeller kunne gjøre store utslag på kjøretid, så det kan forventes stor variasjon i kjøretid.

Datasettet

> Datasettet vi har generert skal kun brukes for læring, og kan ikke benyttes for andre formål. Du kan lese mer om bruk av IMDB sine datasett her.

Vi skal bygge en ganske stor graf basert på data fra IMDB. Den får rundt hundre tusen noder og fem millioner kanter! På den kan vi utføre mange forskjellige grafalgoritmer. Fordi grafen er stor vil du fort få en følelse for effektiviteten av en algoritme.

I grafen vi skal bygge er hver node en skuespiller, og to skuespillere har en kant mellom seg for hver film de har spilt sammen i. Kantene er merket med en film som har en rating. Dermed får vi en *urettet* graf med *merkede*, *parallelle* og *vektede* kanter.

Vi har generert to TSV-filer som sammen utgjør grafen:

movies.tsv Hver linje består av fire felter, separert med `\t`:

`tt`-id Tittel Rating Antall Stemmer

> Hver film er unikt identifisert ved en `tt`-id, der hver `tt`-id er en streng som begynner med `tt` etterfulgt av 7 siffer. Filmen har en tittel og en rating (et tall mellom 0.0 og 10.0) og et antall stemmer. Ingen av oppgavene vil benytte seg av antall stemmer, så det feltet kan ignoreres.

actors.tsv Hver linje består av

`nm`-id Navn `tt`-id_1 `tt`-id_2 ... `tt`-id_k

> Hver skuespiller er unikt identifisert ved en `nm`-id, der hver `nm`-id er en streng som begynner med `nm` etterfulgt av 7 siffer, og et navn. De k siste kollonnene på hver linje er filemene skuespilleren har spilt i, gitt som `tt`-id-er.

Oppgave 1: Bygg grafen

For å kunne implementere grafalgoritmer trenger vi først en graf å jobbe med. Her må vi gjøre et valg om hvordan vi vil *representere* grafen. Du står fritt til å velge den representasjonen du tenker vil være enklest å jobbe med. Det er ingen restriksjoner på hva du kan bruke fra Java eller Pythons standardbibliotek.

Merk at en skuespiller kan ha spilt i en film som vi ikke har data om. Det vil si at en skuespiller kan ha en `tt`-id som ikke forekommer i movies.tsv. Disse `tt`-id-ene skal ignoreres.

Deloppgaver:

1. Skriv et Java eller Python-program som leser inputfilene (`movies.tsv` og `actors.tsv`) og bygger grafen. Det kan være lurt å starte med å bare lese en brøkdel av linjene av inputfilene for testing.

2. For å sjekke om resultatet er rimelig skal du nå kunne skrive en prosedyre som teller antall noder og antall kanter. Skriv ut resultatet, som bør se slik ut:

   ```
   Oppgave 1

   Nodes: 108811
   Edges: 4775477
   ```

   Dersom dere får andre tall enn vi har gitt her, så bør det ligge en mulig forklaring på hvorfor i PDF-en. Det kan finnes andre representasjoner av grafen som egner seg for å løse resten av oppgavene, som gjør at dere får andre tall, og det er helt greit.

Oppgave 2: Six Degrees of IMDB

Basert på konseptet six degrees of separation har det blitt laget noen veldig kule applikasjoner, som for eksempel Six Degrees of Wikipedia. Vi skal nå gjøre noe lignende for IMDB (inspirert av Six Degress of Kevin Bacon).

Skriv en prosedyre som gitt to skuespillere finner en korteste sti som forbinder dem. Programmet skal kunne skrive ut stien, og inneholde både nodene (skuespillerene) og kantene som forbinder dem (se eksempelutskrift nedenfor).

Skriv ut korteste stiler for følgende skuespillere (gitt som `nm`-id-er):

`nm`-id$_1$	`nm`-id$_2$	Navn$_1$	Navn$_2$
`nm2255973`	`nm0000460`	Donald Glover	Jeremy Irons
`nm0424060`	`nm0000243`	Scarlett Johansson	Denzel Washington
`nm4689420`	`nm0000365`	Carrie Coon	Julie Delpy
`nm0000288`	`nm0001401`	Christian Bale	Angelina Jolie
`nm0031483`	`nm0931324`	Atle Antonsen	Michael K. Williams

Skriv ut resultatet på søkene, som bør se slik ut:

```
Oppgave 2

Donald Glover
===[ Lennon or McCartney (5.3) ] ===>  Emma Thompson
===[ Beautiful Creatures (6.2) ] ===>  Jeremy Irons

Scarlett Johansson
===[ Don Jon (6.5) ] ===>  Cuba Gooding Jr.
===[ American Gangster (7.8) ] ===>  Denzel Washington

Carrie Coon
===[ Widows (6.9) ] ===>  Jon Bernthal
===[ The Air I Breathe (6.8) ] ===>  Julie Delpy

Christian Bale
===[ Knight of Cups (5.7) ] ===>  Antonio Banderas
===[ Original Sin (6.1) ] ===>  Angelina Jolie

Atle Antonsen
===[ King Curling (6.2) ] ===>  Ane Dahl Torp
===[ 1001 Grams (6.3) ] ===>  Torsten Knippertz
===[ Miracle at St. Anna (6.1) ] ===>  Michael K. Williams
```

Merk at utskriften ikke trenger å nøyaktig slik ut, men det er viktig at det er lett å lese ut hvilke noder og kanter stien går gjennom. Det finnes ofte flere stier av samme lengde; det er ingen krav til hvilken sti som finnes, så lenge det ikke finnes en kortere.

Oppgave 3: Chilleste vei

Nå skal vi vekte grafen, men hva i all verden er poenget med det? Vi har jo allerede en måte å finne den korteste veien mellom to skuespillere. Det gir tross alt ikke mening å snakke om avstanden mellom to skuespillere gjennom hvem de har spilt i film sammen med: enten har de spilt sammen, eller så har de ikke det. Det vi skal forsøke nå er å finne den *chilleste* veien, eller den mest underholdene, givende, fineste, hva enn du vil kalle det. Målet er å finne en sti fra en skuespiller gjennom de *beste* filmene. Altså, hvis du har et par skuespillere du liker, så skal vi prøve å konstruere en liste med filmer du kan se som knytter de to sammen, og samtidig skal denne seerseansen være en fantastisk opplevelse.

La oss ta utgangspunkt i en enkel vektfunksjon. Vi vet at 10 er den høyeste mulige ratingen, så dersom vi tar $10 - r$ der r er en ratingen til en gitt film, vet vi at dette tallet ikke vil bli negativt. Det vil for eksempel si at en sti som går gjennom to filmer med $r = 9$ vil prioriteres over én film med $r = 7$ (fordi $2 \cdot (10 - 9) = 2$ og $10 - 7 = 3$, og $2 < 3$).

Hvis vi bruker samme eksempler som ovenfor kan vi få en utskrift som:

```
Oppgave 3

Donald Glover
===[ The Martian (8.0) ] ===>  Enzo Cilenti
===[ The Man Who Knew Infinity (7.2) ] ===>  Jeremy Irons
Total weight: 4.8

Scarlett Johansson
===[ Avengers: Infinity War (8.4) ] ===>  Josh Brolin
===[ American Gangster (7.8) ] ===>  Denzel Washington
Total weight: 3.8

Carrie Coon
===[ Avengers: Infinity War (8.4) ] ===>  Elizabeth Olsen
===[ Avengers: Age of Ultron (7.3) ] ===>  Julie Delpy
Total weight: 4.3

Christian Bale
===[ The Dark Knight Rises (8.4) ] ===>  Liam Neeson
===[ For the Love of Spock (7.6) ] ===>  Angelina Jolie
```

```
Total weight: 4.0

Atle Antonsen
===[ In Order of Disappearance (7.2) ] ===>  Stellan Skarsgård
===[ Good Will Hunting (8.3) ] ===>  Casey Affleck
===[ Gone Baby Gone (7.6) ] ===>  Michael K. Williams
Total weight: 6.9
```

I likhet med forrige oppgave er det ingen strenge krav til utskrift, så lenge det er lett å lese ut hvilke noder og kanter stien går gjennom, samt den totale vekten på stien.

Oppgave 4: Komponenter

En urettet (og ikke-tom) graf består av én eller flere komponenter. To noder tilhører samme komponent hvis og bare hvis det finnes en sti mellom dem.

Det vi vil undersøke i denne oppgaven er hvor store de komponentene i IMDB-grafen er. Fra de forrige oppgavene vet vi allerede at det finnes en vei mellom alle skuespillerene vi har testet; intuitivt kommer denne komponenten til å bli veldig stor!

I denne oppgaven skal du finne hvor mange komponenter det er av forskjellige størrelser. Merk at forelesningene ikke dekker hvordan man finner sammenhengende komponenter i en *urettet* graf, og dette er fordi det er et *enklere* problem enn å gjøre tilsvarende i en *rettet* graf.

Programmet skal skrive ut hvor mange komponenter det finnes av ulik størrelse, for eksempel slik:

```
Oppgave 4

There are 1 components of size 103175
There are 1 components of size 19
There are 1 components of size 10
There are 3 components of size 9
There are 1 components of size 8
There are 5 components of size 7
There are 8 components of size 6
There are 14 components of size 5
There are 40 components of size 4
There are 112 components of size 3
```

```
There are 297 components of size 2
There are 4329 components of size 1
```

Oppgave 5: Frivillig bonusoppgave!

Dette er et stort datasett som kan besvare mange spørsmål man kan ha om filmer og skuespillere. Kan du finne svaret på et spørsmål du selv finner interessant? Eksempler kan være (som vi selv ikke har funnet svar på):

- Kan du finne et eksempel der lengden på den korteste stien og den chilleste stien er stor?
- Kan du finne en bedre vektfunksjon for å finne chilleste vei? Kanskje kan man benytte seg av antall stemmer?
- Hvor mange filmer må man inkludere for at den største komponenten fremdeles er bevart (eventuelt, hvor mange filmer kan fjernes)?