

IN2010

Innleveringsoppgave 3

Innlevering

Last opp filene dine på [Devilry](#). Dere står fritt til 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, markdown- eller tekst-fil som henholdsvis skal hete
 - IN2010-innleveringsoppgave3.pdf,
 - IN2010-innleveringsoppgave3.md eller
 - IN2010-innleveringsoppgave3.txt.

Andre formater aksepteres ikke.

- Et kjørbart Java- eller Python-program som kan kjøres fra en fil som heter henholdsvis `Innleveringsoppgave3.java` eller `innleveringsoppgave3.py` (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 `Innleveringsoppgave3`, og være kjørbart med kommandoen `java Innleveringsoppgave3`. For Python skal programmet være kjørbart med `python3 innleveringsoppgave3.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 (eller tekstfilen) som skal leveres skal inneholde instruksjoner for hvordan man kjører programmet og eventuelle 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. 3 sekunder på alle oppgavene tilsammen (på faglærers maskin). Siden det er en stor graf vil små forskjeller kunne gjøre store utslag i kjøretid. Vi foretrekker raske løsninger fremfor trege, men stiller ingen strenge krav til 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*, *parallele* og *vektede* kanter.

Alternativt kan vi se på grafen som en *urettet* og *vektet* graf med *to forskjellige nodetyper*, en type for skuespillere og en type for filmer. I en slik graf vil hver skuespiller kun være koblet til filmer, og hver film er kun koblet til skuespillere.

Vi har generert to [TSV](#)-filer som sammen utgjør grafen:

[movies.tsv](#) Hver linje består av fire felt, 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 eller 8 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-id₁ tt-id₂ ... 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 eller 8 siffer, og et navn. De k siste kolonnene på hver linje er filmene 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 `marvel_movies.tsv` og `marvel_actors.tsv` som er vesentlig mindre 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: 119205

Edges: 5068918

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 Degrees 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 stier 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

=== [The Lazarus Effect (5.2)] ===> Olivia Wilde

=== [The Words (7.0)] ===> Jeremy Irons

Scarlett Johansson

=== [Woody Allen: A Documentary (7.7)] ===> Antonio Banderas

=== [Philadelphia (7.7)] ===> Denzel Washington

Carrie Coon

=== [Avengers: Infinity War (8.4)] ===> Scarlett Johansson

=== [Avengers: Age of Ultron (7.3)] ===> Julie Delpy

Christian Bale

=== [Thor: Love and Thunder (6.4)] ===> Matt Damon

=== [The Good Shepherd (6.7)] ===> Angelina Jolie

Atle Antonsen

=== [The Trip (6.9)] ===> Noomi Rapace

```
=== [ Unlocked (6.3) ] ==> John Malkovich
=== [ Arkansas (6.0) ] ==> Michael K. Williams
```

Merk at utskriften ikke trenger å se 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 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) ] ===> Scarlett Johansson
===[ 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 (7.1) ] ===> Stellan Skarsgård
===[ Good Will Hunting (8.3) ] ===> Casey Affleck
===[ Gone Baby Gone (7.6) ] ===> Michael K. Williams
Total weight: 7.0
```

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. I denne oppgaven skal du finne hvor mange komponenter det er av forskjellige størrelser. Programmet skal skrive ut hvor mange komponenter det finnes av ulik størrelse, for eksempel slik:

Oppgave 4

```
There are 1 components of size 113102
There are 1 components of size 11
There are 2 components of size 10
There are 4 components of size 9
There are 2 components of size 8
There are 6 components of size 7
There are 9 components of size 6
There are 19 components of size 5
There are 43 components of size 4
```

There are 117 components of size 3
There are 320 components of size 2
There are 4666 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)?