Innlevering 3

Høst 2025

Om innleveringen

- Innleveringen skal leveres i innleverings- og tilbakemeldingssystemet Devilry innen **31. oktober klokken 23:59**.
- Dere skal *levere i grupper på to eller tre personer*, under forutsetning av at alle i gruppen både står for det som leveres og alene kan gjøre rede for alle delene av arbeidet.
- Gruppene fra andre innlevering er videreført i Devilry. Ta kontakt ved behov for endringer.
- Dersom du tidligere har levert individuelt, kan du opprette gruppe for denne innleveringen.
- For hver oppgave skal innleveringen inneholde:
  - Et Java- eller Python-program for hver oppgave som ber om det.
    - Du kan dele et program i flere filer, men du må sørge for at programmet kan kjøres med én kommando.
  - Én `.pdf`-, `.md`- eller `.txt`-fil som skal inneholde kommandoen som brukes for å kjøre programmet, samt løsninger på oppgaver som skal besvares med pseudokode eller naturlig språk.
- Du skal ikke levere testfiler eller andre ressurser.
- Med mindre noe annet er spesifisert kan dere bruke hva dere vil fra Java eller Python sine standardbibliotek.
- Kommenter **«Klar for retting»** i Devilry hvis du leverer i god tid før fristen, og innleveringen er klar for retting.

Krav til innleveringen

- Alle oppgaver skal være besvart.
- Programmet må kompilere og kunne kjøres med kommandoen dere har oppgitt.
- Det som leveres skal være *lett forståelig, entydig og presist*.

Ressurser for oppgaven:

https://github.uio.no/IN2010/six-degrees-of-imdb-ressursside

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.

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 (publisert på ressurssiden) 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 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 $\texttt{tt-id}_1$ $\texttt{tt-id}_2$ ... $\texttt{tt-id}_k$

> 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 filemene skuespilleren har spilt i, gitt som `tt-id`-er.

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 skrive en prosedyre som teller antall noder og antall kanter og skriver ut resultatet. Output bør se slik ut for de ulike datasettene og for de ulike representasjonene av grafen:

Filmer som kanter		**Filmer som noder**	
Testdatasett	Datasett	Testdatasett	Datasett

`Oppgave 1`
 	
`Nodes: 28`
`Edges: 94`
	`Oppgave 1`
	
	`Nodes: 126196`
	`Edges: 5342530`
		`Oppgave 1`
		
		`Nodes: 36`
		`Edges: 40`
			`Oppgave 1`
			
			`Nodes: 234094`
			`Edges: 775288`

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.

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. Vi definerer størrelsen på en komponent som antall skuespillere i komponenten. 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 2

There are 1 components of size 120030
There are 1 components of size 15
There are 1 components of size 11
There are 1 components of size 10
There are 3 components of size 9
There are 4 components of size 8
There are 8 components of size 7
There are 12 components of size 6
There are 20 components of size 5
There are 46 components of size 4
There are 124 components of size 3
There are 335 components of size 2
There are 4617 components of size 1
```

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 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	nm8076281	Scarlett Johansson	Emma Mackey
nm4689420	nm0000365	Carrie Coon	Julie Delpy
nm0000288	nm2143282	Christian Bale	Lupita Nyong'o
nm0637259	nm0931324	Tuva Novotny	Michael K. Williams

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

```
Oppgave 3

Donald Glover
===[ LA Originals (7.2) ] ===>  Terry Crews
===[ Inland Empire (6.8) ] ===>  Jeremy Irons

Scarlett Johansson
===[ Rough Night (5.2) ] ===>  Kate McKinnon
===[ Barbie (7.1) ] ===>  Emma Mackey
```

```
Carrie Coon
===[ His Three Daughters (8.5) ] ===>  Natasha Lyonne
===[ But I'm a Cheerleader (6.8) ] ===>  Julie Delpy

Christian Bale
===[ The Promise (6.1) ] ===>  Oscar Isaac
===[ Star Wars Episode IX: The Rise of Skywalker (6.4) ] ===>  Lupita Nyong'o

Tuva Novotny
===[ Dear Alice (5.5) ] ===>  Danny Glover
===[ LUV (5.9) ] ===>  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.

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 4

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) ] ===>  Ariana Greenblatt
===[ Barbie (7.1) ] ===>  Emma Mackey
Total weight: 4.5

Carrie Coon
===[ His Three Daughters (8.5) ] ===>  Elizabeth Olsen
===[ Avengers: Age of Ultron (7.3) ] ===>  Julie Delpy
Total weight: 4.2

Christian Bale
===[ The Big Short (7.8) ] ===>  Adepero Oduye
===[ 12 Years a Slave (8.1) ] ===>  Lupita Nyong'o
Total weight: 4.1

Tuva Novotny
===[ Borg McEnroe (6.9) ] ===>  Demetri Goritsas
```

```
===[ Saving Private Ryan (8.6) ] ===>  Paul Giamatti
===[ 12 Years a Slave (8.1) ] ===>  Michael K. Williams
Total weight: 6.4
```

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.

Utforsk videre

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:

- Kan du finne et eksempel der lengden på den korteste stien og den chilleste stien er stor?
- Kan du finne en god alternativ 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)?