Innlevering 2

Høst 2026

Om innleveringen

- Innleveringen skal leveres i innleverings- og tilbakemeldingssystemet Devilry innen **12. oktober klokken 23:59**.
- Dere skal helst *levere i grupper* på inntil 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.
- Du kan levere individuelt dersom du ønsker (du trenger ikke fylle ut et skjema eller lignende).
- Du skal bruke programmet `in2010-testrunner` som du finner her: `https://github.uio.no/IN2010/in2010-testrunner/`
- `in2010-testrunner` hjelper deg med å:
  - generere kodeskjelett,
  - kjøre koden mot testfiler, og
  - lage en zip-fil som skal leveres i Devilry.
- Med mindre noe annet er spesifisert kan dere bruke hva dere vil fra Java eller Python sine standardbibliotek, men ingen andre biblioteker.
- Kommenter **«Klar for retting»** i Devilry hvis du leverer i god tid før fristen, og innleveringen er klar for retting.

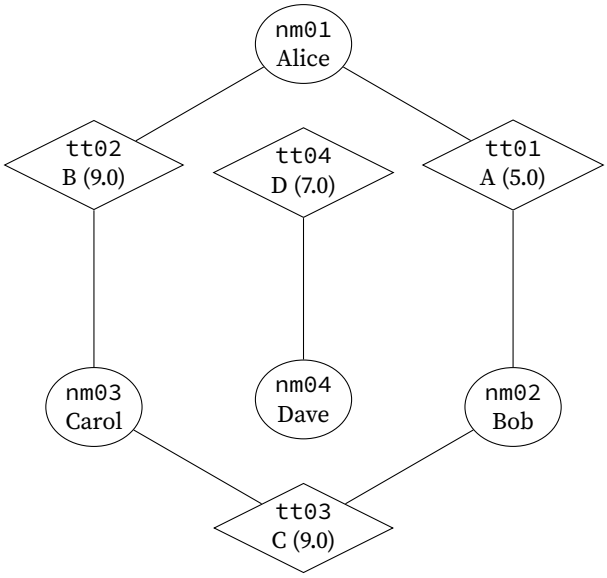
Krav til innleveringen

- Alle oppgaver skal være besvart.
- Programmene må kompilere og kunne kjøres med `in2010-testrunner`.
- Det som leveres skal være *lett forståelig, entydig og presist*.

Datasettet

Vi skal bygge en ganske stor graf basert på data fra IMDB. Den får hundretusenvis av noder og kanter! Fordi grafen er stor vil du fort få en følelse for effektiviteten av en algoritme. Merk at vi også har laget mindre datasett for å gjøre det enklere å feilsøke under utvikling.

Grafen vi skal bygge er 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. Under ser du et lite eksempel en graf som består av fire filmer og fire skuespillere:



Skuespillere har et navn (for eksempel *Alice*) og en `nm`-ID (for eksempel `nm01`). Filmer har en tittel (for eksempel *A*), en `tt`-ID (for eksempel `tt01`) og en rating (for eksempel 5.0). Input og output for programmet er mer presist beskrevet i slutten av oppgavesettet.

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 eksempelgrafen over skal resultatet være:

```
There are 1 components of size 1
There are 1 components of size 3
```

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 skrive ut stien. For eksempelgrafen over er resultatet av den korteste stien mellom `nm01` og `nm02`:

```
nm01        tt01        nm02
```

Og resultatet av den korteste stien mellom `nm01` og `nm03` er:

```
nm01        tt02        nm03
```

Det finnes ofte flere stier av samme lengde; det er ingen krav til hvilken sti som skrives ut, 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 de samme to eksemplene som ovenfor kan vi få en utskrift som:

```
nm01	tt02	nm03	tt03	nm02
nm01	tt02	nm03
```

Input

Grafen skal konstrueres fra input som leses fra `stdin`. Input beskriver grafen som skal bygges, etterfulgt av to ulike typer *spørringer*.

- Én linje med et tall M som representerer antall filmer.
  - Hver av de neste M linjene beskriver en film og består av tre felter, separert med `\t`:

    `tt`-ID Tittel Rating

    Hver film er unikt identifisert ved en `tt`-ID, der hver `tt`-ID er en streng som begynner med `tt` etterfulgt flere siffer. Filmen har en tittel og en rating (et tall mellom 0.0 og 10.0).
- Én linje med et tall A som representerer antall skuespillere.
  - Hver av de neste A linjene beskriver en skuespiller og består av to felter, separert med `\t`:

    `nm`-ID Navn

    Hver skuespiller er unikt identifisert ved en `nm`-ID, der hver `nm`-ID er en streng som begynner med `nm` etterfulgt av flere siffer. Skuespilleren har et navn.
- Én linje med et tall E som representerer antall kanter.
  - Hver av de neste E linjene beskriver at en skuespiller spilte i en film og består av to felter, separert med `\t`:

    `tt`-ID `nm`-ID

    Der `tt`-ID gir filmen som skuespilleren med `nm`-ID spilte i.
- Én linje med et tall Q_s som representerer antall spørringer om korteste stier.
  - Hver av de neste Q_s linjene beskriver en spørring om den korteste stien mellom to skuespillere og består av to felter, separert med `\t`:

    `nm`-ID$_1$ `nm`-ID$_2$
- Én linje med et tall Q_c som representerer antall spørringer om chilleste stier.
  - Hver av de neste Q_s linjene beskriver en spørring om den chilleste stien mellom to skuespillere og består av to felter, separert med `\t`:

    `nm`-ID$_1$ `nm`-ID$_2$

Eksempel-input

Eksempel-inputet beskriver grafen som vises øverst i oppgavesettet, og spørringene er de som er brukt som eksempel i de relevante seksjonene.

```
4
tt01 A   5.0
tt02 B   9.0
tt03 C   9.0
tt04 D   7.0
4
nm01 Alice
nm02 Bob
nm03 Carol
nm04 Dave
7
tt01 nm01
tt01 nm02
tt02 nm01
tt02 nm03
tt03 nm02
tt03 nm03
tt04 nm04
2
nm01 nm02
nm01 nm03
2
nm01 nm02
nm01 nm03
```

Output

Programmet skal først skrive ut:

- Et tall C som svarer til antall forskjellige komponentstørrelser.
  - Hver av de neste C linjene skal inneholde teksten:

    `There are X components of size Y`

    Der X er antall komponenter av størrelse Y.
- Hver av de neste Q_s linjene (der Q_s kommer fra input) skal inneholde en korteste sti. Stien skrives ut som en liste av `nm`- og `tt`-IDer separert med `\t`.
- Hver av de neste Q_c linjene (der Q_c kommer fra input) skal inneholde en chilleste sti. Stien skrives ut som en liste av `nm`- og `tt`-IDer separert med `\t`.

Eksempel-output

Følgende er korrekt output for eksempel-inputet:

```
2
There are 1 components of size 1
There are 1 components of size 3
nm01	tt01	nm02
nm01	tt02	nm03
nm01	tt02	nm03	tt03	nm02
nm01	tt02	nm03
```

Utforsk videre (frivillig)

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)?