Vedlegg Oblig 2 IN2010

Instruksjoner:

Programmet kan kjøres med kommandoen «python oblig2.py». (evt python3 oblig2.py)

Kjøretid for programmet:

Omtrent 12 sekunder på en av gruppemedlemmenes laptop «brukt siden starten av videregående.»

Oppgave 1:

Grafen vår er representert ved hjelp av to hashmaps (dicts), som respektivt mapper tt-ider til en mengde nm-ider, og nm-ider til en mengde tt-ider.

Både antall noder og kanter er korrekt, så grafen ser solid ut.

Oppgave 2:

Besvarer oppgaven ved bruk av bredde først søk, for å finne korteste sti. Sjekker naboene til startnoden. Besøkte noder legges i en mengde, og naboer som ikke enda er sjekket legges i et objekt av klassen Ko (fifo). Besøkte noder legges også i HashMapen sti, hvor nøkkelverdi er noden og innholdsverdi er mor-noden. Går gjennom Ko-objektet til det ikke er flere noder å sjekke, eller til sluttnoden er funnet. Da forkorter vi den stien som ble funnet ved å begynne i sluttnoden og følge mor-nodene tilbake til startnoden.

Oppgave 3:

For å løse denne oppgaven har vi implementert en modifisert versjon av Dijikstras algoritme.

Hver gang vi finner en “chillere” vei til en annen node vil denne måten legges som veien vi bruker i en dictionary som har oversikt over denne veien og veien blir lagt til prioritets-køen vår. Dette skjer om og om igjen helt til noden vi skal gå utfra er den vi leter etter – da returnerer vi bare dictionary-en med veien vår og går gjennom den for å skrive ut resultatet.

Oppgave 4:

I denne oppgaven har vi implementert en variant av BFS.

Hver gang vi ikke lenger finner en ny nabonode, velger vi en tilfeldig ubesøkt node og starter BFS på nytt.