Rundtur med Den Norske Turistforeningen (DNT) 12 poeng

Fagutvalget ved institutt for informatikk har bestemt seg for å arrangere en rundtur i et vakkert norsk fjellandskap for alle IFI-studenter. Dette sammenfaller med at Den Norske Turistforeningen har lansert en ny app, sammen med offentlige datasett, slik at det er gode muligheter for å gjøre et godt informert rutevalg. Siden de fleste studenter på IFI har en forkjærlighet for grafer, er det ikke til å unngå at de vil bruke datasettet til å bygge en graf som fanger nok informasjon til å kunne algoritmisk bestemme den beste rundturen.

Grafen som bygges er en rettet graf $G = (V, E)$, der hver node $u \in V$ representerer en lokasjon, og en kant $(u, v) \in E$ representerer en turvei fra u til v. Ingen noder har en kant til seg selv.

(a) Det viser seg at grafen har en ganske spesiell egenskap: *Hver node inngår i makismalt én sykel*. Hver sykel representerer en *rundtur*.

Gi en algoritme som finner alle rundturene i grafen, og oppgi kjøretidskompleksiteten. Du kan kalle på algoritmer kjent fra pensum.

Maksimalt 5 poeng.

Her er det tilstrekkelig å svare at vi finner de sterkt sammenhengende komponentene av G og returnerer alle komponenter som har størrelse større enn én. Dette er i $\mathcal{O}(|V| + |E|)$.

Input: En graf $G = (V, E)$
Output: En mengde av mengder av noder

```
1 Procedure RoundTrip(G)
2     roundtrips ← empty set
3     components ← StronglyConnectedComponents(G)
4     for component ∈ components do
5         if 1 < |component| then
6             add component to roundtrips
7     return roundtrips
```

Det trekkes 1 poeng for å ikke utelukke komponenter som kun inneholder én node.

(b) En av rundturene fra (a) blir valgt som reisemål, hvor t er en node i rundturen. Turen starter ved et *utfartssted*, som er enkelt å komme seg til med kollektivtransport. Et utfartssted er identifisert ved noder som har inngrad 0. Det siste som gjenstår i planleggingen er å velge et egnet utfartssted turen skal starte ved for å nå t.

Datasettet inneholder heldigvis også informasjon om hvor lang tid en gjennomsnittlig turgåer bruker langs en turvei. Dette blir gitt som en vektfunksjon w, slik at for hver kant $(u, v) \in E$, så angir $w(u, v)$ tidsbruk som et positivt heltall. Vi vil bruke vektene til å finne ut hvilket utfartssted som er best egnet for å komme seg til reisemålet.

Input er en rettet og vektet graf $G = (V, E)$ med vektfunksjon w, en mengde med utfartssteder U og en node t i reisemålet. Output er noden $s \in U$ slik at avstanden mellom s og t er så lav som mulig.

Gi en algoritme som effektivt finner utfartsstedet som har lavest avstand til t i reisemålet, og oppgi kjøretidskompleksiteten. Du kan kalle på algoritmer kjent fra pensum.

Maksimalt 7 poeng.

En mulig løsning er å bruke Dijkstra sin algoritme for korteste stier fra hver node $s \in U$, og velge noden som har kortest avstand til t. Dette er i $\mathcal{O}(|U| \cdot (|V| + |E|) \cdot \log(|V|))$. En beskrivelse av en slik løsning kan få opp til 4 poeng.

En bedre løsning er å heller finne korteste avstander (med Dijkstra) *fra t i den reverserte grafen G_r*, altså grafen der alle kanter er snudd. I en slik løsning kan man først beregne avstander fra t til alle noder, og deretter gå gjennom alle noder i U og finne ut hvilken som er nærmest t. Dette er i $\mathcal{O}((|V| + |E|) \cdot \log(|V|))$, altså samme kjøretidskompleksitet som Dijkstra. En slik løsning kan få opp til 7 poeng.

Input: En graf $G = (V, E)$ med vektfunksjon w, en mengde med noder U og en node t
Output: En node $s \in U$ som minimerer avstanden til t

```
Procedure BestStartLocation(G, U, t)
    G_r ← ReverseGraph(G) // Anta at vektfunksjonen også reverseres
    dist ← Dijkstra(G_r, t)
    s ← arbitrary node in U
    for u ∈ U do
        if dist[u] < dist[s] then
            s ← u
    return s
```

En alternativ løsning er å legge til en ny node a i grafen, og legge til en kant (a, s) med $w(a, s) = 0$ for alle noder $s \in U$. Hvis vi gjør Dijkstra med a som startnode, så kan man finne det beste utfartsstedet ved å finne den *andre* noden i den korteste stien fra a til t, som er garantert å være et utfartssted. Den vil være like effektiv som løsningen over og kan også få opp til 7 poeng.