Kollektivtilbudet i Oslo — 15 poeng

I det siste bystyremøtet ble det sluttet flere vedtak vedrørende kollektivtilbudet i Oslo. La kollektivtilbudet være representert som en sammenhengende, urettet og vektet graf $G = (V, E)$, der $u \in V$ er en holdeplass, en kant $\{u, v\} \in E$ er en *strekning* og vektfunksjonen w angir reisetid (i minutter) for en strekning.

(a) Grafen er *sammenhengende* og *urettet*. Forklar kort hva hver av egenskapene vil si for en som reiser kollektivt i Oslo.

At grafen er sammenhengende vil si at en reisende kan reise mellom to vilkårlige holdeplasser ved å bruke kollektivtilbudet.

At grafen er urettet vil si at en som har reist en strekning alltid kan reise den samme strekningen tilbake.

- Inntil 1 poeng.

(b) Bystyret har vedtatt at det skal bygges nye holdeplasser langs alle *strekninger* som overstiger k minutter. Under bygningsarbeidet vil det ikke være mulig å reise langs noen av disse strekningene. Skriv en algoritme som tar grafen som representerer det nåværende kollektivtilbudet, samt en grense k, og sjekker om det vil være mulig å reise mellom alle par av holdeplasser under bygningsarbeidet.

Her kan vi gjøre et dybde-først søk, der kanter med vekt som overstiger k ignoreres, og sjekke om alle noder nås gjennom én traversering.

```
1  Procedure CheckConnected(G, k)
2      stack ← stack initialized with an arbitrary vertex u ∈ V
3      visited ← empty set
4
5      while stack is not empty do
6          u ← stack.pop()
7          if u not in visited then
8              add u to visited
9              for (u, v) in E do
10                 if w(u, v) < k then
11                     add v to stack
12
13     return visited = V
```

- Inntil 4 poeng for en løsning i $\mathcal{O}(|E|)$.
- Inntil 1 poeng dersom kanter med vekt som overstiger k benyttes.

(c) Det viser seg at algoritmen fra deloppgave (b) svarer **`false`** for k-en bystyret hadde i tankene. Gi en algoritme som bystyret kan bruke til å finne den minste verdien for k som algoritmen fra deloppgave (b) svarer **`true`** for.

En naturlig løsning er å beregne det minimale spenntreet for grafen, og finne verdien til den kanten med høyest vekt.

```
1  Procedure FindMinimalK(G)
2      T ← Prim(G)
3      k ← 0
4      for (u, v) in edges of T do
5          k ← max(k, w(u, v))
6      return k
```

- Inntil 4 poeng for en løsning i $\mathcal{O}(|E| \cdot \log(|V|))$.

(d) Det siste året har det vært to hendelser der det har vært full stopp ved en holdeplass. Det ledet til at reisende ikke kunne nå destinasjonen sin. Bystyret har vedtatt en garanti: dersom det blir full stopp ved en holdeplass, skal det fremdeles være mulig for reisende å nå sin destinasjon via en alternativ rute.

Garantien gjelder for alle holdeplasser.

Forklar hvordan vi kan bruke en algoritme kjent fra pensum til å sjekke om kollektivtilbudet leverer denne garantien.

> Her ønsker vi å sjekke om grafen er 2-sammenhengende; altså skal grafen forbli sammenhengende hvis vi fjerner en vilkårlig node (sammen med dens tilhørende kanter). Fra pensum kjenner vi en algoritme som finner alle separasjonsnoder i lineær tid, og algoritmen `IsBiconnected` sjekker at det ikke finnes noen separasjonsnoder i grafen. Dette er det samme som å sjekke at garantien holdes for alle holdeplasser.
>
> - Inntil 3 poeng for en løsning i $\mathcal{O}(|V| + |E|)$.

(e) Møtet til bystyret varte for lenge, så de bestilte pizza, og etter det fløt det over med nye kreative idéer. Den siste i denne rekken er følgende: for å avlaste de mest trafikkerte rutene, vil de gi *negative billettpriser* på de minst trafikkerte strekningene.

Før de setter i gang med prosjektet må de forsikre seg om at det ikke er mulig å opptjene seg vilkårlig høye summer i negative billettpriser ved å reise rundt og rundt.

Billettprisen for en strekning er gitt av en vektfunksjon c. En strekning kan være svært trafikkert i én retning, men lite trafikkert i den andre, derfor kan $c(u, v)$ være ulik $c(v, u)$ for en strekning $\{u, v\} \in E$.

Forklar hvordan vi kan bruke en algoritme kjent fra pensum til å sjekke om det finnes måter å opptjene vilkårlig høye summer i negative billettpriser.

> Her ønsker vi å finne negative sykler i en graf. Algoritmen `BellmanFord` finner korteste stier i en rettet graf *og* identifiserer negative sykler. Dermed kan vi kalle på `BellmanFord` med grafen G fra en vilkårlig node $u \in V$ og vektfunksjonen c og returnere **`false`** dersom den ikke rapporterer at det finnes negative sykler.
>
> Merk at `BellmanFord` finner avstanden fra én node til alle andre noder, derfor er input både grafen og en startnode. Siden grafen er urettet spiller det ingen rolle hvilken node vi starter fra.
>
> - Inntil 3 poeng.
> - Trekk 1 poeng dersom det ikke er spesifisert hvilken node algoritmen skal starte fra.