Whops!-oppgjør 10 poeng

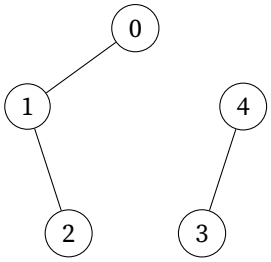
Ansatte på Institutt for informatikk har vært på instituttseminar hvor det har sklidd helt ut med utlån av små beløp mellom hverandre. De skal alle gjøre opp for seg ved å overføre penger via tjenesten Whops!. Dessverre har det oppstått et virvar av interne stridigheter mellom de ansatte, hvor mange har endt opp med å blokkere hverandre på Whops!, som gjør det vanskelig å få oppgjøret til å gå opp. Administrasjonen ser fortvilet på situasjonen, og skjønner at dette problemet vil oppstå igjen i årene fremover, hvor det også skal ansettes vanvittig mange. De har rutiner på plass for å få oversikt over hvor mye hver ansatt har lagt ut eller skylder totalt. Nå ber de om hjelp fra studentene i IN2010 (som på dette tidspunktet har blitt eksperter på algoritmer og datastrukturer) til å finne en generell måte å sjekke om oppgjøret kan gjennomføres gjennom Whops! eller ikke, som også vil skalere etterhvert som Institutt for Informatikk får ubegripelig mange ansatte.

Du setter flittig i gang, og formaliserer problemet som et grafproblem. Du lar $G = (V, E)$, der hver ansatt være representert ved en node $v \in V$, og lar det være en (urettet) kant $\{u, v\} \in E$ mellom to ansatte dersom de ikke har blokkert hverandre på Whops!. For hver ansatt $v \in V$ kan du slå opp i en tabell `T`, der `T[v]` er et heltall som angir hvor mye v skylder eller har lagt ut totalt. Dersom `T[v]` er negativ, betyr det at v skylder penger, men dersom `T[v]` er positiv betyr det at v har lagt ut penger. Du vet at summen av alle tallene i `T` er nøyaktig 0. Merk at det betyr at dersom alle kunne utveksle penger mellom hverandre gjennom Whops! (muligens via andre ansatte) ville oppgjøret alltid gått opp.

Du skal gi en algoritme som tar G og `T` som input og svarer **`true`** hvis oppgjøret kan gjøres gjennom Whops!, og **`false`** dersom det ikke er mulig.

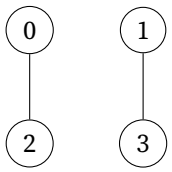
Her er to eksempler på hvordan en G og `T` kan se ut. For eksempelet med G_1 og T_1 kan oppgjøret gjøres gjennom Whops!, men for G_2 og T_2 er det ikke mulig å gjøre oppgjøret gjennom Whops!.

G_1



v	$\mathsf{T}_1[v]$
0	500
1	−200
2	−300
3	−50
4	50

G_2



v	$\mathsf{T}_2[v]$
0	300
1	−250
2	0
3	−50

Innsikten som trengs for å løse denne oppgaven er at dersom summen (med hensyn til T) i hver av komponentene av grafen G er 0, så går oppgjøret opp med Whops!. Vi kan samle komponenten som tilhører en node v, ved å gjøre et dybde-først (eller bredde-først) fra v. Ved et fullt dybde-først søk samler vi komponentene for hver node $v \in V$. Til slutt kan vi gå gjennom hver komponent, og sjekke at summen av `T[`v`]` for alle v i komponenten er 0.

I denne oppgaven er det viktigere at studenten klarer å kommunisere idéen bak algoritmen godt, enn å skrive pseudokode som veldig enkelt lar seg oversette til et konkret programmeringsspråk. Det er helt greit om studenten gjør antagelser om representasjonen av grafen (for eksempel en nabomatrise, nabolister eller en mer objekt-orientert stil). Løsningsforslaget nedenfor er ganske detaljert, men binder seg ikke til noen spesifikk representasjon av grafen.

Input: En graf $G = (V, E)$ og en tabell av heltall T
Output: **true** hvis oppgjøret går opp i Whops!, **false** ellers.

```
1  Procedure ComponentOf(s, G)
2      component ← empty set
3      stack ← [s]
4      while stack is not empty do
5          v ← pop stack
6          add v to component
7          for (v, u) ∈ E do
8              if u ∉ component then
9                  add u to stack
10     return component
11
12 Procedure ComponentsOf(G)
13     components ← empty set
14     visited ← empty set
15     for v ∈ V do
16         if v ∉ visited then
17             C ← ComponentOf(v, G)
18             add C to components
19             add every c ∈ C to visited
20     return components
21
22 Procedure Settleable(G, T)
23     for C in ComponentsOf(G) do
24         if Σ_{v∈C} T[v] ≠ 0 then
25             return false
26     return true
```

En setning som tilsvarer «Sjekk at alle komponenter summerer til 0 med hensyn til T » gir opp til 5 poeng. De resterende 5 poengene fordeles etter hvor gode og presise beskrivelser studenten gir av hvordan dette skal beregnes (altså, hvordan DFS kan brukes for å beregne komponentene, og hvordan vi sjekker summene med hensyn på T).