Whops!-logger 10 poeng

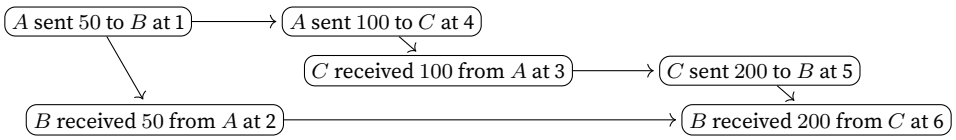
Tjenesten Whops! lar brukere overføre penger mellom hverandre gjennom en app. De prøver i så stor grad som mulig å unngå å lagre informasjon om brukerene sine sentralt, og lar heller informasjonen lagres på brukeren sin egen mobiltelefon.

En gang i blant har de behov for å anskaffe en komplett logg av overføringer for en periode for å kunne feilsøke systemene sine. Da henter de inn *lokale* logger fra noen utvalgte mobiltelefoner i systemet. En lokal logg er en liste av hendelser, der hver hendelse beskriver enten at et beløp er sent eller mottatt. I tillegg inneholder hver hendelse et tidsstempel som forteller når beløpet ble sendt eller motatt. For enkelhets skyld er tidsstemplene representert som positive heltall. Her er et lite eksempel på tre logger fra mobiltelefoner A, B og C.

A	B	C
A sent 50 to B at 1	B received 50 from A at 2	C received 100 from A at 3
A sent 100 to C at 4	B received 200 from C at 6	C sent 200 to B at 5

Dessverre er tidsstemplene fra mobiltelefonene upålitelige. De observerer ofte at et mottak av et beløp forekommer *før* beløpet ble sendt, i følge tidsstemplene. I eksempelet ovenfor mottar C et beløp på 100 *før* A har sendt beløpet til C.

For å approksimere en riktig logg lar de hver hendelse fra de lokale loggene være noder i en rettet graf $G = (V, E)$. Dersom to hendelser u og v kommer direkte etter hverandre i en lokal logg, så finnes det en rettet kant (u, v) i grafen. For hver hendelse u som representerer at et beløp er sendt, så finnes det en rettet kant (u, v) der v er hendelsen for det korresponderende mottaket av det beløpet. Her er grafen for eksempelet ovenfor:



Den ønskede komplette loggen for dette eksempelet skal se slik ut:

A sent 50 to B at 1
B received 50 from A at 2
A sent 100 to C at 4
C received 100 from A at 3
C sent 200 to B at 5
B received 200 from C at 6

Hver node v kan aksessere tidsstempelet fra loggen ved $v.\texttt{ts}$. Skriv en algoritme som tar rettet graf $G = (V, E)$ som input, og skriver ut alle noder i V i en rekkefølge slik at:

- Dersom det finnes en kant fra u til v så skal u skrives ut før v.
- u skal skrives ut før v hvis $u.\texttt{ts} < v.\texttt{ts}$, så lenge det ikke bryter med det første kravet.

Denne oppgaven kan løses med en litt modifisert topologisk sortering. I en standard topologisk sortering av nodene plasserer alle noder med inngrad 0 på en stack. For hver node u som besøkes, så fjerner vi alle kanter (u, v), og plasserer v på stacken dersom den får inngrad 0 som en konsekvens av å ha besøkt u. Dette er tilstrekkelig for å garantere at dersom det er en sti fra u til v, så skrives u ut før v. Merk at det ikke spiller noen rolle for den topologiske sorteringen om vi bruker en stack, en kø, eller en annen datastruktur med konstant tid på innsetting og uttak.

Det andre kravet sier at dersom det *ikke* er en sti fra u til v, så skal u skrives ut før v hvis $u.\texttt{ts} < v.\texttt{ts}$; med andre ord vil vi *sortere* nodene etter tidsstempelet, enn så lenge det ikke bryter med det første kravet. Som allerede nevnt, er det ikke viktig hva slags datastruktur vi bruker for å lagre nodene som er klare for å bli besøkt for at den topologiske sorteringen skal være korrekt. Dermed kan vi bruke en *heap* som ordnes etter tidsstemplene, og dermed garantere at vi hele tiden besøker den noden med det minste tidsstempelet, uten å bryte med den topologiske ordningen.

Input: En rettet graf $G = (V, E)$
Output: Skriv ut nodene i V slik kantene i grafen respekteres, og tidsstemplene minimeres.

```
1  Procedure TopSortLogs(G = (V, E))
2      H ← empty heap, ordered by ts
3      for v ∈ V do
4          if indegree of v is 0 then
5              add v to H
6      while H is not empty do
7          u ← pop smallest from H
8          print(u)
9          for (u, v) ∈ E do
10             remove u from the incoming edges of v
11             if indegree of v is 0 then
12                 add v to H
```

En umodifisert topologisk sortering gir opp til 5 poeng. Sortering etter tidsstemplene, uten å ta høyde for kantene i grafen gis opp til 2 trøstepoeng.

En modifisert topologisk sortering, som løser problemet i kvadratisk tid gis opp til 8 poeng. For full uttelling må algoritmen kjøre i $O((|V| + |E|) \log |V|)$.