Fire typer grafer poeng

Du ønsker å forbedre kommunikasjonsnettverket i en liten by. Historisk sett har all kommunikasjon mellom husstander i byen skjedd ved hjelp av brevduer. På denne måten kunne alle nå hverandres hus direkte. Men brevduer er upraktiske, så du skal finne bedre måter for husstandene å kommunisere med hverandre på. Du har utforsket følgende muligheter:

- Den nye startupen HamiltonBikes har begynt å bygge nye sykkelstier og kommer til å drifte brevlevering mellom alle hus. De garanterer at man kan ta seg en sykkeltur som går innom alle husene i byen, uten å måtte sykle forbi det samme huset to ganger.
- Husene kan forbindes med nettverkskabel, men med hensyn til kostnaden, vil redundante forbindelser unngås.
- Noen bor så nærme hverandre at naboer kan rope for rask én-til-én kommunikasjon.

Du innser at hver kommunkasjonsform kan formaliseres som en graf der nodene er husstandene og kantene er forbindelsene mellom disse. Du identifiserer fire typer grafer, som hver tilsvarer en kommunikasjonsform:

Type 1 er en urettet komplett graf (brevduene)
Type 2 er en urettet graf med en hamiltonsk sykel (sykkelstien)
Type 3 er et urettet tre (nettverkskabler)
Type 4 er en urettet ikke-sammenhengende graf (roping)

Alle grafer i denne oppgaven har mer enn to noder.

Graftypene er gjentatt der du trenger det, så du trenger ikke å huske dem.

(a) 7 poeng

Din første oppgave er å sjekke om HamiltonBikes virkelig leverer det de lover. De tilbyr å sende deg et kart over sykkelstiene og et forslag på en runde som går forbi hvert hus nøyaktig én gang. Du oversetter problemet til et avgjørelsesproblem:

Type2	
Instans:	En graf G
Spørsmål:	Inneholder G en hamiltonsk sykel?

Du bestemmer deg for å lage en effektiv algoritme som verifiserer **Type2**. Algoritmen tar en graf og et sertifikat som input, og svarer JA hvis sertifikatet bekrefter at grafen er av type 2, NEI ellers. Algoritmen skal kjøre i polynomisk tid.

HamiltonBikes leverer sertifikatet som et array C av noder i G, slik at hver node er med nøyaktig én gang. Skriv pseudokode for verifikatoren `Type2Verifier`.

ALGORITHM: VERIFIKATOR FOR **TYPE2**

Input: En graf G og et array av noder C
Output: Returner JA hvis C viser at G er av type 2, NEI ellers

```
1 Procedure Type2Verifier(G, C)
    // Fyll ut
```

ALGORITHM: VERIFIKATOR FOR **TYPE2**

Input: En graf G og et array av noder C
Output: Returner JA hvis C viser at G er av type 2, NEI ellers

```
1 Procedure Type2Verifier(G, C)
2     (V, E) ← G
3     N ← |C|
4     for i ← 0 to N − 1 do
5         if (C[i], C[(i + 1) mod N]) ∉ E then
6             return NEI
7     return JA
```

Det viktige i denne oppgaven er at man sjekker at etterfølgende par av noder i C svarer til kanter i G. Det er også nødvendig å sjekke at siste og første node i C utgjør en kant i G; dette gjør vi i siste iterasjon av loopen, der $i = N - 1$, hvor vi da sjekker om $(\texttt{C[N - 1]}, \texttt{C[0]})$ er en kant i G (fordi $N \bmod N = 0$).

Veiledende poengfordeling:

- 5 poeng for å sjekke om $e \in E(C) \Rightarrow e \in E(G)$; minus 1 poeng om de glemmer siste kant
- 2 poeng for signatur og generell oversiktlig besvarelse
- Ingen poeng for å finne en generell sykel, f.eks. ved hjelp av BFS/DFS

(b) 3 poeng

Forklar kort (maks 4 setninger) hvorfor svaret på forrige deloppgave viser at avgjørelsesproblemet **Type2** er i NP. Ikke gi en detaljert kjøretidsanalyse av pseudokoden din for `Type2Verifier`.

Dersom svaret ditt på forrige deloppgave har mangler, kan du gjøre nødvendige antagelser uten å få følgefeil.

> `Type2Verifier` kjører i polynomiell tid. Derfor kan JA-instanser av **Type2** verifiseres i polynomiell tid. Dette er kravet for å være i NP.

(c) 3 poeng

Type 1 er en urettet komplett graf (brevduene)
Type 2 er en urettet graf med en hamiltonsk sykel (sykkelstien)
Type 3 er et urettet tre (nettverkskabler)
Type 4 er en urettet ikke-sammenhengende graf (roping)

Alle grafer i denne oppgaven har mer enn to noder.

En fordel med sykkelstiene til HamiltonBikes er redundans: hvis for eksempel alle sykkelstiene rundt et hus må stenges på grunn av byggearbeid, så er det fortsatt mulig å sykle rundt i byen.

Forklar kort (maks 4 setninger) hvorfor grafer av type 2 er 2-sammenhengende.

Hvis grafen inneholder en hamiltonsk sykel kan alle noder nå alle andre på minst to måter. Dette er ved å følge sykelen «den ene eller den andre» veien. Her bruker vi at grafen er urettet.

(d) 8 poeng

Type 1 er en urettet komplett graf (brevduene)
Type 2 er en urettet graf med en hamiltonsk sykel (sykkelstien)
Type 3 er et urettet tre (nettverkskabler)
Type 4 er en urettet ikke-sammenhengende graf (roping)

Alle grafer i denne oppgaven har mer enn to noder.

Bystyret skjønner ikke grafteori, så de vil gjerne at du viser hvordan man skulle kunne komme seg rundt i byen dersom det gjøres byggearbeid. Du bestemmer deg for å lage et program som skriver ut to distinkte stier fra et gitt hus til et annet.

Du tar altså utgangspunkt i en graf $G = (V, E)$ som er av type 2, og to noder s og t i grafen.

Skriv pseudokode for en prosedyre `TwoPaths`, som skriver ut to distinkte stier fra s til t.

Du er gitt et sertifikat C, som beskrevet i deloppgave (a).

Ikke tenk på formatet på utskriften; det viktige er at nodene skrives ut i riktig rekkefølge.

ALGORITHM: SKRIV UT TO DISTINKTE STIER FRA s TIL t

Input: En graf G, to noder s og t og et sertifikat C
Output: Skriver ut to distinkte stier fra s til t

```
1 Procedure TwoPaths(G, C, s, t)
  | // Fyll ut
```

Siden C er et sertifikat for at G inneholder en hamiltonsk sykel, så utgjør C en sykel som inneholder alle noder i G. Det vil si at vi har en sykel som nødvendigvis inneholder både s og t. Alt vi trenger å gjøre er å finne s og så følge sykelen i begge retninger til vi finner t, hvor hver retning svarer til en distinkt sti mellom s og t.

```
1  Procedure TwoPaths(G, C, s, t)
2  |  N ← |C|
3  |  si ← 0
4  |  while C[si] ≠ s do
5  |  |  si ← si + 1
6  |  i ← si
7  |  while C[i] ≠ t do
8  |  |  print(C[i])
9  |  |  i ← (i + 1) mod N
10 |  print(C[i])
11 |  i ← si
12 |  while C[i] ≠ t do
13 |  |  print(C[i])
14 |  |  i ← (i − 1) mod N
15 |  print(C[i])
```

Veiledene poengfordeling:

- 2 poeng for sti en vei
- 2 poeng for sti motsatt vei
- 2 poeng for å huske modulo-operator eller lignende for å få til syklisk iterasjon
- 2 poeng for riktig oppsett (signatur/ryddig svar)
- Ingen poeng for å finne alle stier i G ved hjelp av DFS
- 1-2 poeng hvis de finner 2 mulige distinkte stier og stopper etter det.

(e) 3 poeng

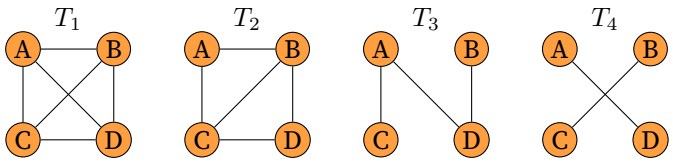
Type 1 er en urettet komplett graf (brevduene)
Type 2 er en urettet graf med en hamiltonsk sykel (sykkelstien)
Type 3 er et urettet tre (nettverkskabler)
Type 4 er en urettet ikke-sammenhengende graf (roping)

Alle grafer i denne oppgaven har mer enn to noder.

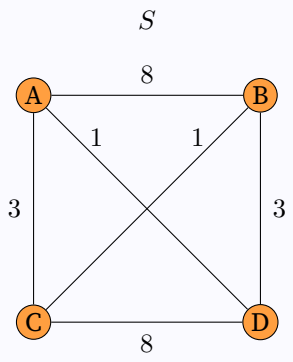
Vi har lyst til å få oversikt over alle kommunikasjonsmåter i byen. Derfor gir vi forskjellig vekt til de ulike kanttypene, basert på kostnad. Deretter slår vi disse sammen til en graf S. Der det er flere kanter beholder vi bare den med lavest vekt.

Her er fire eksempelgrafer, en av hver type, hvor

- kanter i T_1 har vekt 10
- kanter i T_2 har vekt 8
- kanter i T_3 har vekt 3
- kanter i T_4 har vekt 1



Fyll ut kantvektene i grafen S.



(f) 4 poeng

Type 1 er en urettet komplett graf (brevduene)
Type 2 er en urettet graf med en hamiltonsk sykel (sykkelstien)
Type 3 er et urettet tre (nettverkskabler)
Type 4 er en urettet ikke-sammenhengende graf (roping)

Alle grafer i denne oppgaven har mer enn to noder.

Du konstruerer den sammenslåtte grafen G på samme måte som i forrige deloppgave utifra de følgende grafene:

- G_1 av type 1, hvor alle kanter har vekt 10,
- G_2 av type 2, hvor alle kanter har vekt 8,
- G_3 av type 3, hvor alle kanter har vekt 3,
- G_4 av type 4, hvor alle kanter har vekt 1.

Nå har du lyst til å finne den billigste måten å forbinde husene i byen. Derfor har du bestemt deg for å bruke Kruskals algoritme på den sammenslåtte grafen G.

Besvar spørsmålene nedenfor, og begrunn svaret *kort* (maks 4 setninger).

- Finnes det nøyaktig én billigste måte å forbinde husene i byen?
- Er det mulig for alle i byen til å kommunisere bare ved bruk av roping? Med andre ord, kunne vi ha bygget et spenntre for G, med kun kanter fra G_4?
- Din kollega sier at hvis G_4 ikke har noen kanter, så er det minimale spenntreet for G nøyaktig G_3. Stemmer dette?
- Hva er vekten på den *siste* kanten Kruskals algoritme velger når den bygger spenntreet?

- Nei, det er ikke garantert fordi flere kantar kan ha samme vekt. Et unikt minimalt spenntre er bare garantert når ingen kanter har samme vekt.
- Nei, fordi G_4 ikke er sammenhengende. Poenget med spenntreet er at det skal spenne ut grafen, vi må derfor bruke andre kanter enn kun de fra G_4.
- Ja. Dette er fordi G_3 er av type 3 og derfor er et tre. Hvis G_4 ikke inneholder noen kanter, vil alle de billigste kantene være i G_3 så spenntreet er derfor også minimalt.
- 3. Kruskal terminerer når siste kant er valgt, og det er den billigste kanten som forbinder de to siste spenntrærne (T_1 og T_2). Denne kanten kan ikke være i G_4 som forklart i punkt 2 over. Siden G_3 er et spenntre, som forklart i punkt 3, må det finnes en kant som forbinder enhver ikke-triviell todeling av nodene. Dette er fordi hvis det var mulig å lage T_1 og T_2 uten at det fantes en slik kant i G_3 så hadde ikke G_3 vært et spenntre.

Kun halv uttelling hvis studenten forklarer ut fra grafene fra forrige deloppgave istedenfor for en generell graf. Manglende begrunnelse gir ikke uttelling.

(g) 2 poeng

Type 1 er en urettet komplett graf (brevduene)
Type 2 er en urettet graf med en hamiltonsk sykel (sykkelstien)
Type 3 er et urettet tre (nettverkskabler)
Type 4 er en urettet ikke-sammenhengende graf (roping)

Alle grafer i denne oppgaven har mer enn to noder.

La $G = (V, E)$ være en graf og la s og t være noder i V. Vi skal nå finne lengden til den korteste stien fra s til t.

- Hvilken algoritme løser problemet mest effektivt hvis du vet at G er av type 1 og alle kantene har vekt 10?
- Hvilken algoritme løser problemet mest effektivt hvis du vet at G er av type 2 og alle kantene har vekt 8?

- Hvilken algoritme løser problemet mest effektivt hvis du vet at G er av type 1 og alle kantene har vekt 10?
  - DFS
  - BFS
  - Dijkstra
  - Prim
  - Bellman Ford
  - Topologisk sortering
  - En algoritme som bare returnerer 10
- Hvilken algoritme løser problemet mest effektivt hvis du vet at G er av type 2 og alle kantene har vekt 8?
  - DFS
  - BFS
  - Dijkstra
  - Prim
  - Bellman Ford
  - Topologisk sortering
  - En algoritme som bare returnerer 8

Merk at det er en liten svakhet i oppgaven her, hvor vi burde spesifisert at $s \neq t$. Det er allikevel ikke direkte feil, fordi en sti ikke kan inneholde samme node flere ganger; altså er det å gå fra s til s ikke en sti.

(h) 8 poeng

Nå skal du finne den korteste kommunikasjonsveien fra posten (A) til alle hus i byen, formalisert som noder i en mengde V. Du har representert alle kommunikasjonstyper i de følgende kantmengdene:

- E_1 inneholder en kant mellom hvert par av noder i V.
  Alle kanter i E_1 har vekt 10.
- $E_2 = \{\{A, B\}, \{B, C\}, \{C, D\}, \{D, E\}, \{E, F\}, \{F, G\}, \{G, H\}, \{E, H\}, \{F, H\}, \{F, A\}\}$
  Alle kanter i E_2 har vekt 8.
- $E_3 = \{\{A, C\}, \{B, C\}, \{C, D\}, \{D, E\}, \{E, F\}, \{F, G\}, \{G, H\}\}$
  Alle kanter i E_3 har vekt 3.
- $E_4 = \{\{A, D\}, \{C, D\}, \{B, D\}, \{F, G\}\}$
  Alle kanter i E_4 har vekt 1.
- $V = \{A, B, C, D, E, F, G, H\}$

Finn de korteste stiene fra A til alle andre noder. Du kan bruke kanter fra alle fire kantmengdene. Husk at det kan være flere kommunikasjonsveier mellom hus, altså at det er parallelle kanter mellom nodene.

Fyll ut tabellen med korrekt minimalavstand fra A for hver node i V.

A	0
B	2
C	2
D	1
E	4
F	7
G	8
H	10