IN2010 Plenum

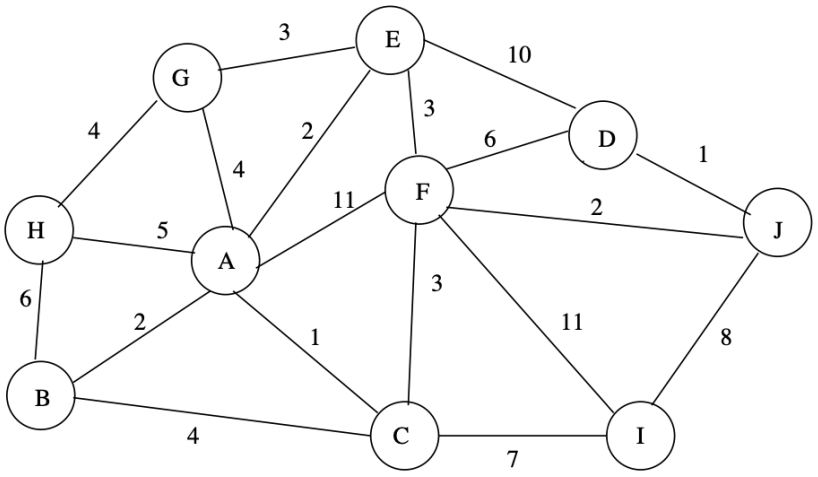
Gruppe 4

5b Kruskal og Prim (vekt 5%)

Gitt følgende graf:

(Fortsettes på side 5.)

[redacted]

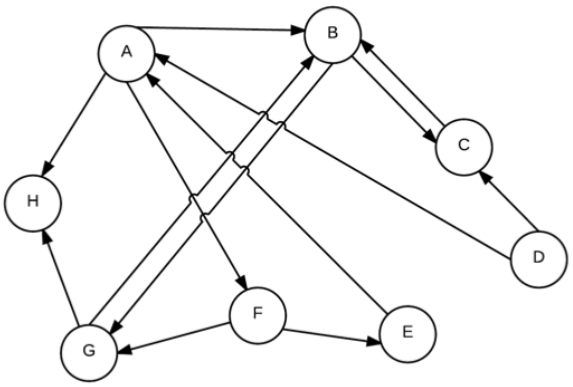


1. Finn et minimum spenntre for grafen ved hjelp av *Kruskal*s og *Prim*s algoritmer. Vis tydelig resultatet etter hvert trinn av algoritmene ved å liste kantene i den rekkefølgen de er valgt.
2. Er dette treet unikt?
3. Når kan vi garantere at treet er unikt?

Oppgave 3 Grafer (vekt 14%)

3a Sterkt sammenhengede komponenter (SCCs) (vekt 7%)

Gitt en rettet graf med noder $A, \ldots, H$ som vist i Figur 2.



Figur 2: En rettet graf

1. Hvilke sterkt sammenhengende komponenter (Strongly Connected Components (SCCs)) har grafen i Figur 2? Du skal illustrere hvordan SCCs er funnet algoritmisk ved å vise trinnene i algoritmen.

2. Gitt en vilkårlig rettet graf G, la G' være en rettet graf der hver node i G' representerer en SCC av G. For nodene u og v i G' finnes det en kant (u, v) hvis det finnes en kant i G som forbinder SCCene som tilsvarer u og v. Kan G' sorteres topologisk? Begrunn kort.

3b Korteste vei (vekt 7%)

Korteste vei-en til alle-problemet (single-source shortest path problem) handler om å finne korteste vei fra en startnode til alle andre noder i en graf. Det kan være mer enn en vei som har den minste kosten. For eksempel, for grafen G i Figur 3 med startnode A, har vi 3 stier fra A til E med kost 4 ($\langle A, C, E \rangle$, $\langle A, D, C, E \rangle$ og $\langle A, B, E \rangle$).

Implementer en algoritme som i tillegg til å finne korteste vei fra en gitt startnode s til alle andre noder w også beregner antall korteste veier fra s til w. La `int num_paths` være den variabelen i klassen `Vertex` som inneholder denne informasjonen. Du kan anta at grafen ikke har negative kanter.

Oppgave 5 Korteste enkle sykel (15%)

I denne oppgaven skal du skrive pseudokode for to effektive algoritmer som gitt en rettet graf G med positive vekter returnerer vekten av den korteste enkle sykelen i G. Dersom G ikke inneholder noen sykel, retunerer du vekt ∞.

5a (4%)
Forklar hvorfor dybde først-søk ikke egner seg her.

5b (6%)
Skriv en algoritme der du bruker Dijkstra for å finne den korteste enkle sykelen. Hvis du bruker Dijkstra slik som den er (umodifisert), trenger du ikke å gjengi koden, du kan da anta at du har en metode med signatur void dijkstra (Vertex s) . Modifiserer du Dijkstra, må du vise den modifiserte Dijkstras algoritme.

Hva er kompleksiteten til algoritmen din?

5c (5%)
Skriv en algoritme der du løser problemet ved bruk av Floyd i din algoritme.

10 % **9** La u og v være to forskjellige noder i en vektet, rettet graf G. La p være den korteste stien fra u til v, og la p' være den korteste av alle de stiene fra u til v som ikke inneholder negative sykler.

Betrakt følgende påstander:

1. Stien p vil aldri inneholde en negativ sykel.
2. Hvis G inneholder en negativ sykel, finnes ikke p.
3. Hvis G inneholder en negativ sykel, kan vi likevel finne p'.

Forklar om utsagnene stemmer eller ikke, ev. med hvilke unntak, antagelser eller forbehold.

Svar relativt grundig (f.eks. ca. 100–200 ord).

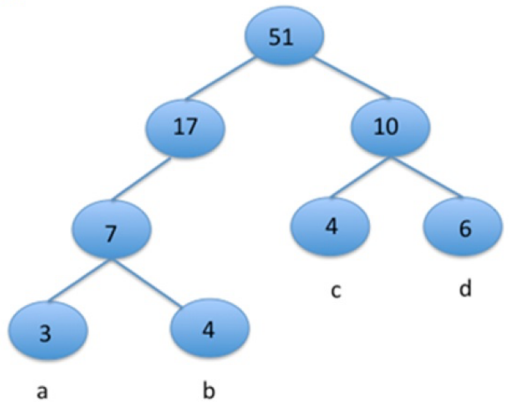
Du kan anta at p og p' er unike; ingen andre stier av samme type er like korte.

Oppgave 3a: Huffman

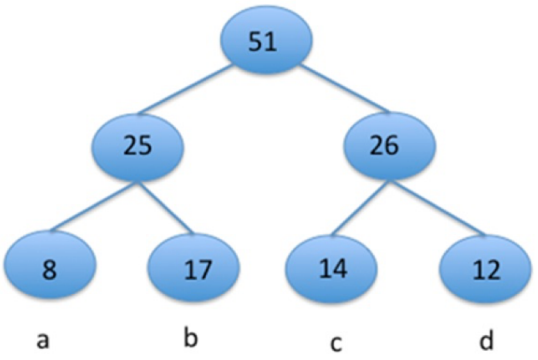
Gitt de to trærne A og B under. For hvert tre, er treet et gyldig Huffman-tre? Hvis det ikke er gyldig, forklar hvorfor og endre det slik at det blir gyldig. Du kan anta at vektingen til løvnodene er riktig.

I denne oppgaven skal du svare med digital håndtegning. Bruk eget skisseark (utdelt). Se instruksjon for utfylling av skisseark på pult.

A:



B:



Maks poeng: 8

Oppgave 3b: Huffman

Gitt tekststrengen “AADDCCCEEEEE”, vis Huffman-treet og -koden for hvert tegn i strengen.

I denne oppgaven skal du svare med digital håndtegning. Bruk eget skisseark (utdelt). Se instruksjon for utfylling av skisseark på pult.

Fire typer grafer

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)

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 4: 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
```

(b)

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.

(c)

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.

(d)

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 5: 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
```

(e)

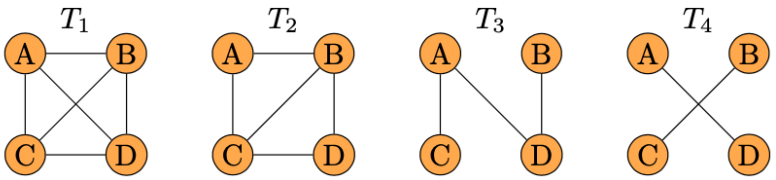
Type 1 er en urettet komplett graf (brevduene)
Type 2 er en urettet graf med en hamiltonsk sykel (sykkelstien)
Type 3 er et urettet tre (nettverkskabler)
Type 4 er en urettet ikke-sammenhengende graf (roping)

Alle grafer i denne oppgaven har mer enn to noder.

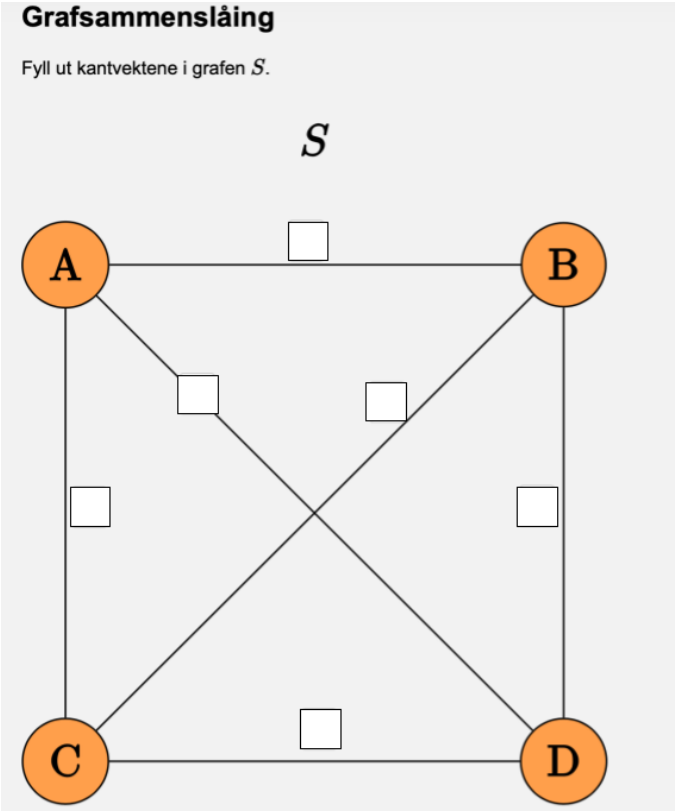
Vi har lyst til å få oversikt over alle kommunikasjonsmåter i byen. Derfor gir vi forskjellig vekt til de ulike kanttypene, basert på kostnad. Deretter slår vi disse sammen til en graf S. Der det er flere kanter beholder vi bare den med lavest vekt.

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)

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?

(g)

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?

(h)

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.

Korteste stier II

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

Node	Avstand
A	
B	
C	
D	
E	
F	
G	
H	

(5 p) 5. Du skal konstruere en Huffman-kode for tegnene a–d, med frekvenser som angitt nedenfor.

Tegn	a	b	c	d
Frekvens	4	1	6	2

Hva blir kodeordet for b?

Merk: Venstre barn plukkes ut først, og kanten til venstre barn får verdien 0.

15. Du skal lage en algoritme som hjelper politiet med å sette opp veisperringer. Som input får du en urettet graf med følgende egenskaper:
– Kantene representerer veier og nodene representerer veikryss
– Hver kant har en lengde som representerer veistrekningen
– En bestemt node representerer åstedet for en forbrytelse
– Et sett med noder angir havner og grenseoverganger, såkalte «kantnoder»
Du får også opplyst maksimal hastighet forbryterne kan ha beveget seg i fra åstedet, samt hvor lang tid som har gått siden forbrytelsen ble begått. Du kan anta at forbryterne ikke kan ha nådd frem til noen av kantnodene ennå. Algoritmen skal fortelle politiet hvor (dvs. på tvers av hvilke kanter) de skal sette opp veisperringer, slik at (i) forbryterne ikke kan komme seg til en kantnode (ved å traversere kanter) uten å passere en veisperring, og (ii) antallet veisperringer er så lite som mulig. Beskriv svært kort en algoritme som effektivt løser problemet.