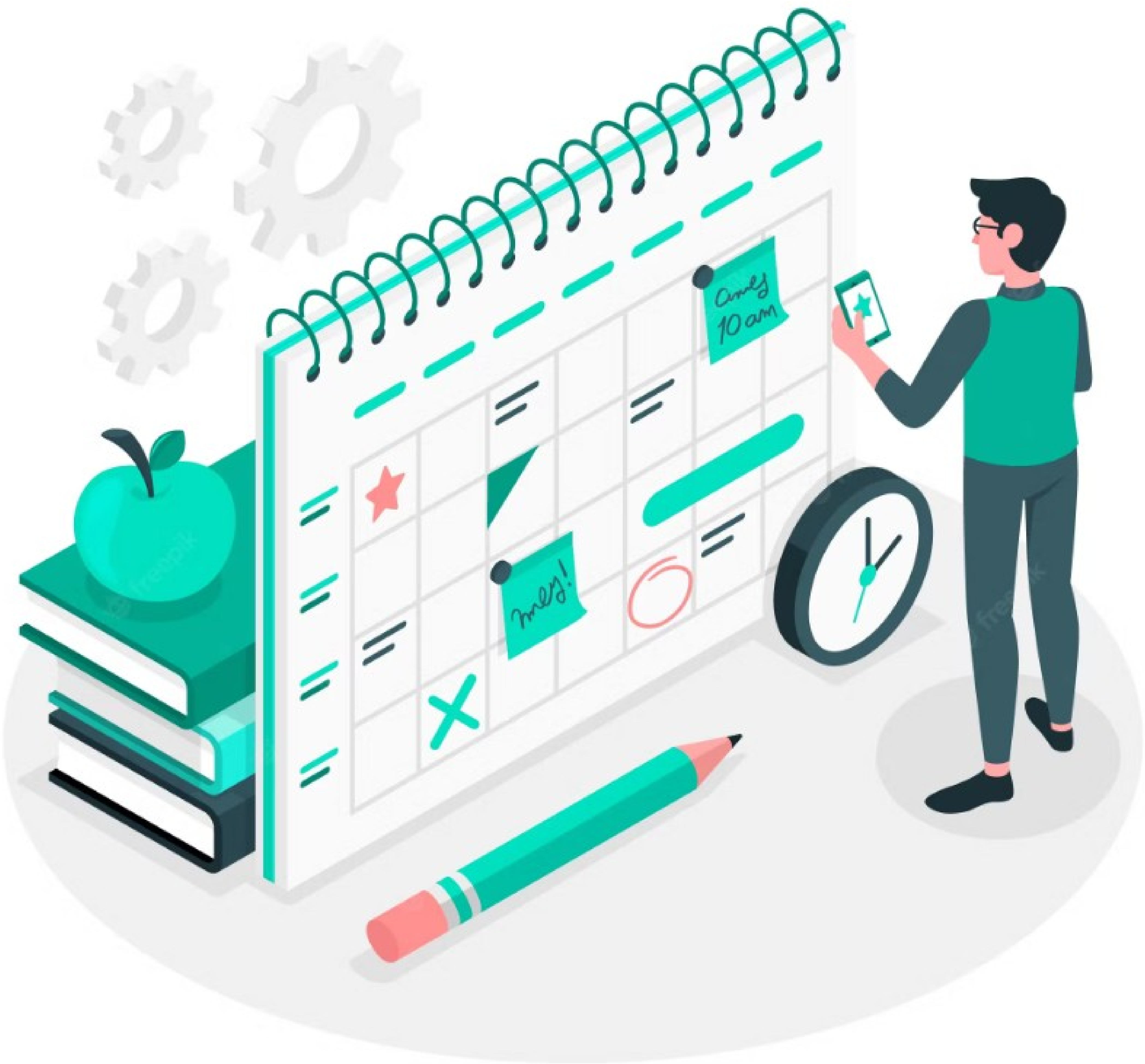
IN2010 - Gruppe 5

Repitisjonstime - 2-Sammenhengende grafer og sterkt sammenhengende komponenter

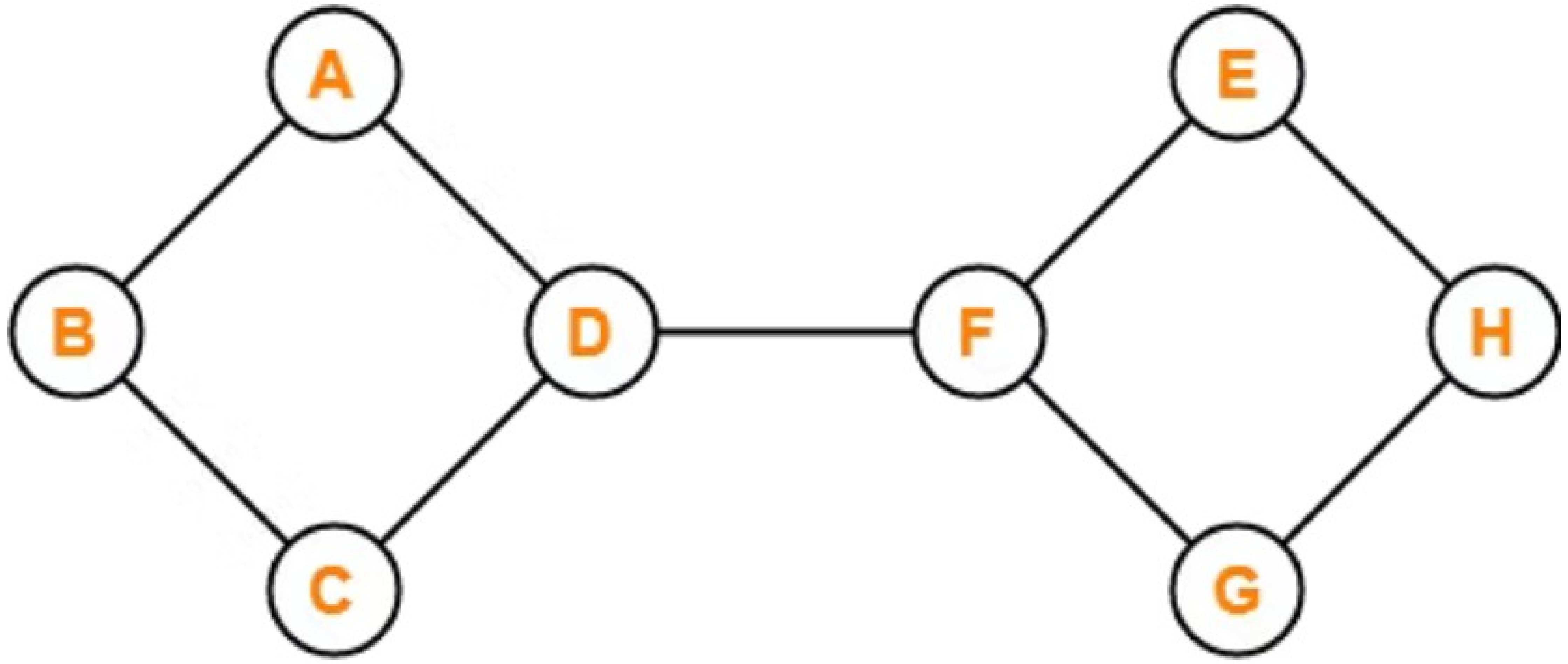


Dagens Plan

- Sammenhengende Grafer
- Seperasjonsnoder
- Pseudokode
- Sterkt Sammenhengende komponenter
- Kosaraju
- Oppgaver i fellesskap
- Q&A/Andre tema?
- Kattis oppgave

Sammenhengende grafer

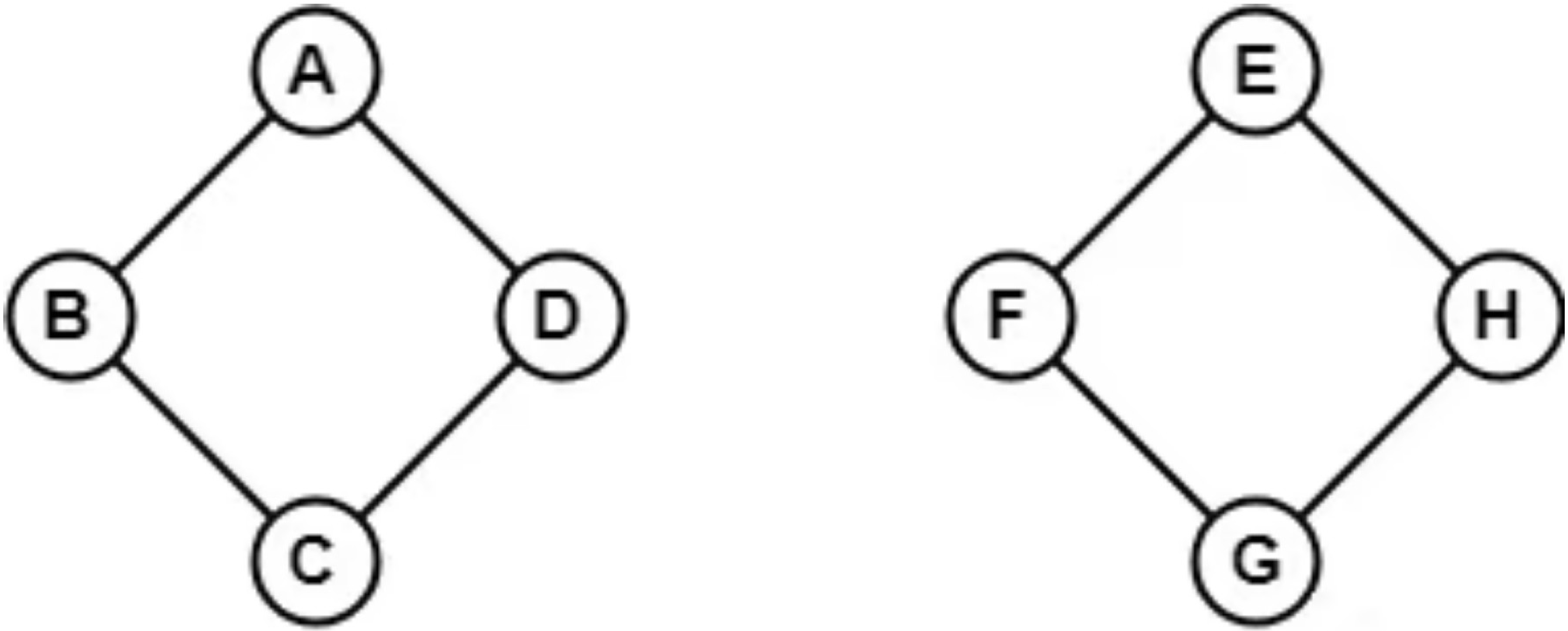
1



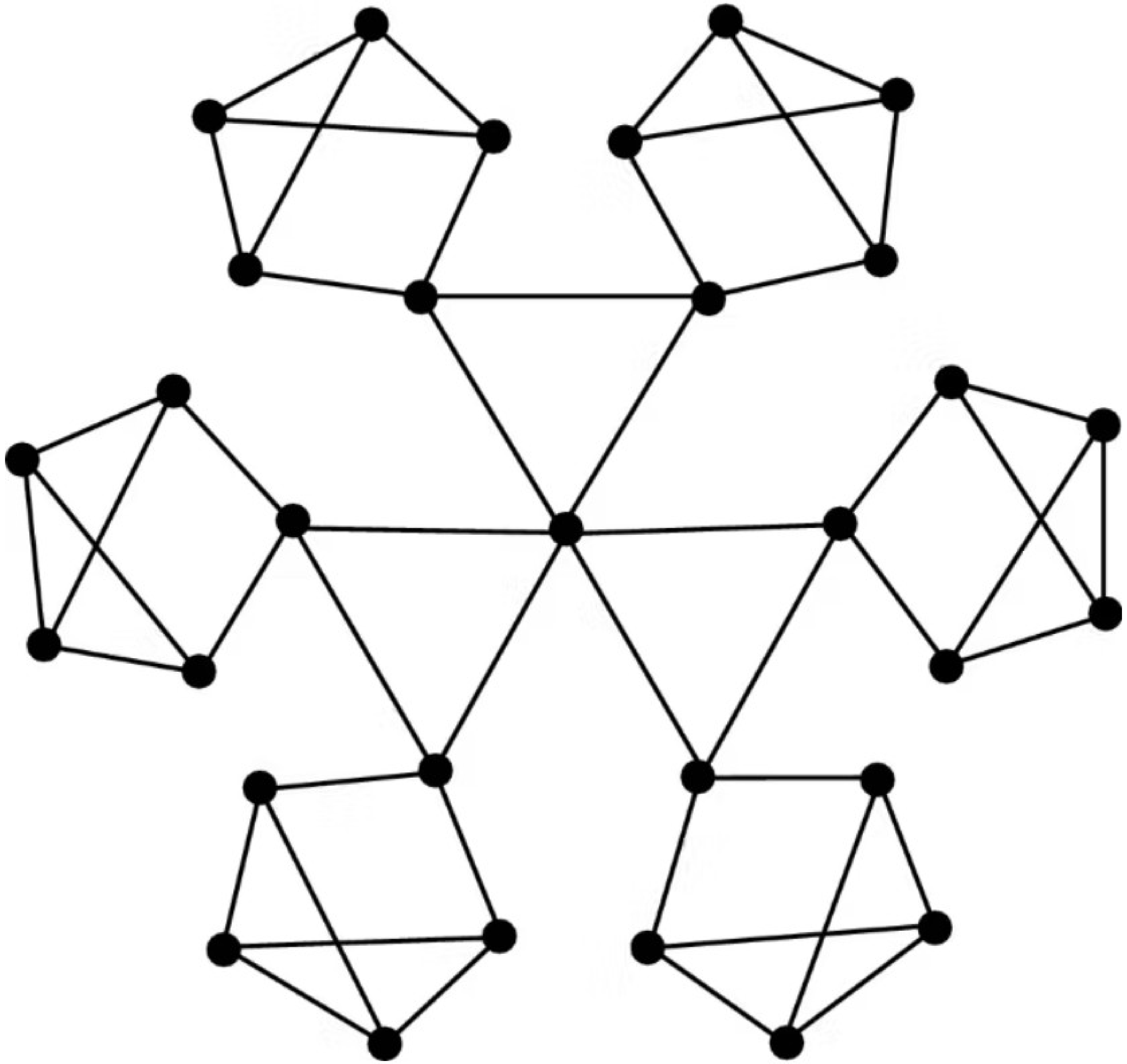
Example of Connected Graph

Sammenhengende grafer

- Rent intuitivt: En graf som henger sammen/ er graf som består av et komponent
- Definisjon: En graf er sammenhengende hvis det finnes en sti mellom hvert par av noder
- Bullet 3



Ikke-Sammenhengende graf



2-Sammenhengende grafer

- En graf er 2-sammenhengende dersom den forblir sammenhengende etter vi har fjernet mindre enn 2 noder
- Altså at den er sammenhengende etter å ha fjernet 1 node
- Det kan ikke bare gjelde for en spesifikk node, men envher vilkårlig node

1

ALGORITHM: NAIV ALGORITME FOR Å SJEKKE OM EN GRAF ER 2-SAMMENHENGENDE

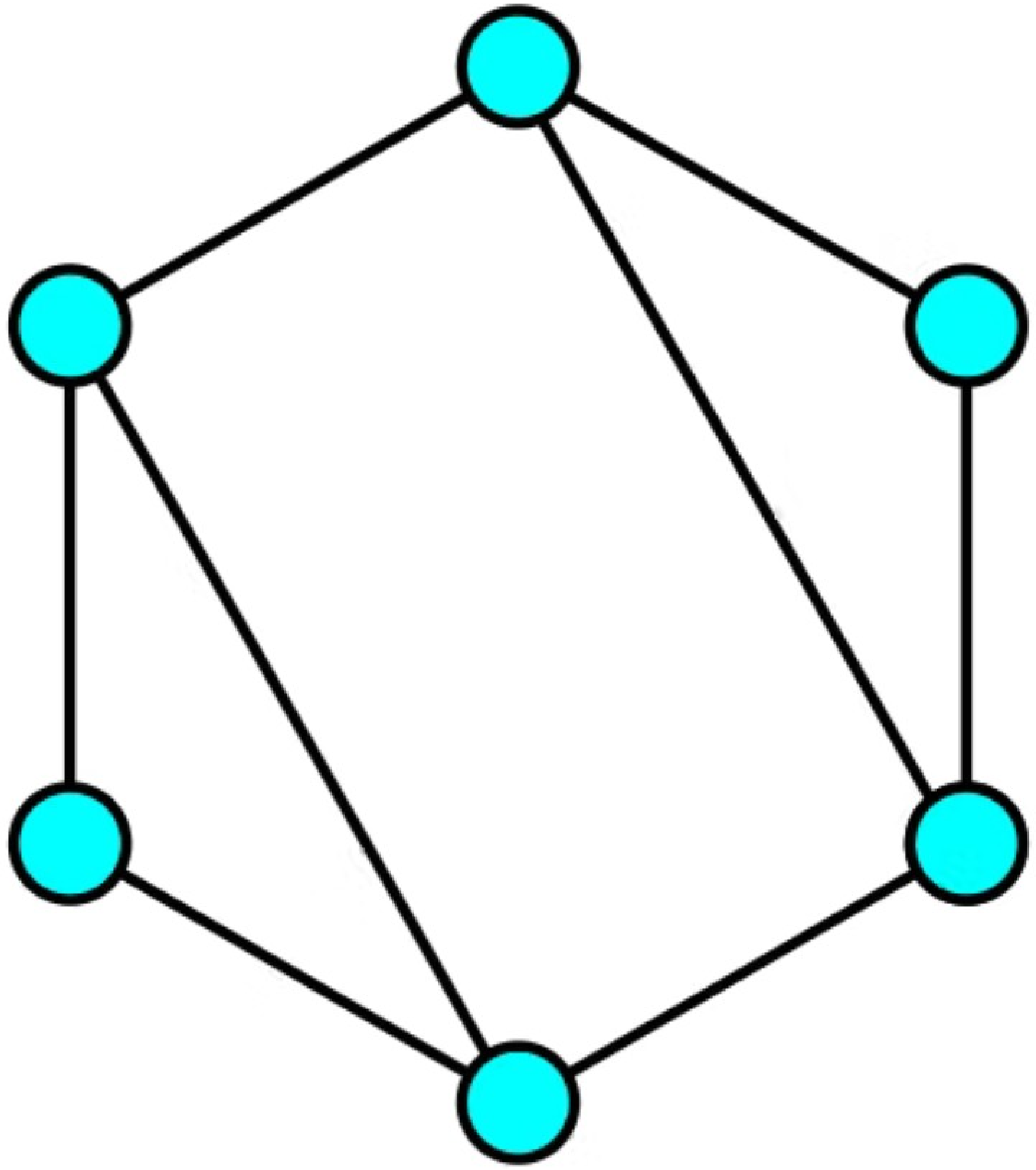
Input: En sammenhengende graf $G = (V, E)$

Output: Gir **true** hvis grafen er 2-sammenhengende, **false** ellers

```
Procedure IsBiconnectedNaive(G)
    for v ∈ V do
        G′ = (V′, E′) ← G with v removed
        visited ← empty set
        u ← any vertex u ∈ V′
        DFSVisit(G′, u, visited)
        if visited ≠ V′ then
            return false
    return true
```

Intuitiv, men treg løsning

1



En graf kan være mer enn 2-sammennhengende

- Altså at vi må fjerne flere noder for at grafen ikke skal bli sammenhengende
- Dette kalles k-sammenhengende
- Da vil vi heller se etter seperasjonsnodene, istedenfor å gå gjennom alle noder i grafen

Seperasjonsnoder

- Dette er nodene som gjør at grafen er sammenhengene
- En 2-sammenhengende graf kan ha flere seperasjonsnoder
- Bullet 3

ALGORITHM: FINN ALLE SEPARASJONSNODER I EN SAMMENHENGENDE GRAF

```
Input: En graf sammenhengende G = (V, E)
Output: Returnerer alle separasjonsnoder i G
depth ← empty map
low ← empty map
seps ← empty set
Procedure SeparationVertices(G)
    s ← choose arbitrary vertex from V
    depth[s] ← 0
    low[s] ← 0
    children ← 0

    for (s, u) ∈ E do
        if u ∉ depth then
            SeparationVerticesRec(G, u, 1)
            children ← children + 1

    if children > 1 then
        add s to seps
    return seps

Procedure SeparationVerticesRec(G, u, d)
    depth[u] ← d
    low[u] ← d

    for (u, v) ∈ E do
        if v ∈ depth then
            low[u] ← min(low[u], depth[v])
            continue

        SeparationVerticesRec(G, v, d + 1)
        low[u] ← min(low[u], low[v])
        if d ≤ low[v] then
            add u to seps
```

Pseudokode

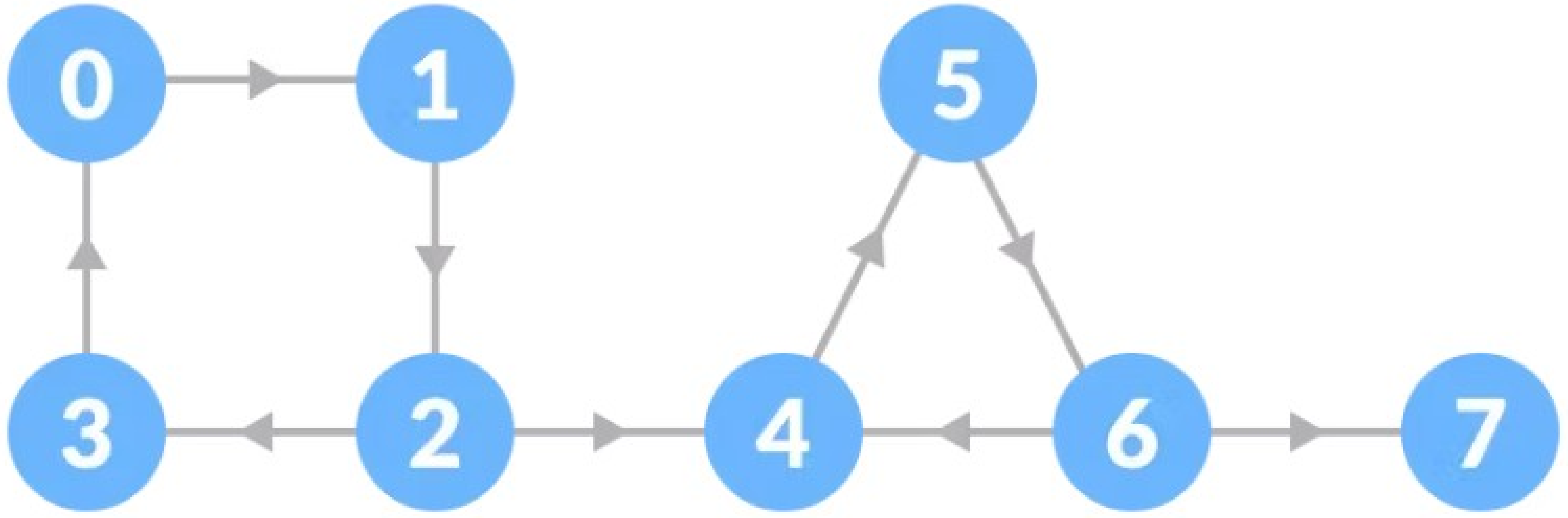
Visualisering med notability

1

Sterk sammenhengende komponenter

1

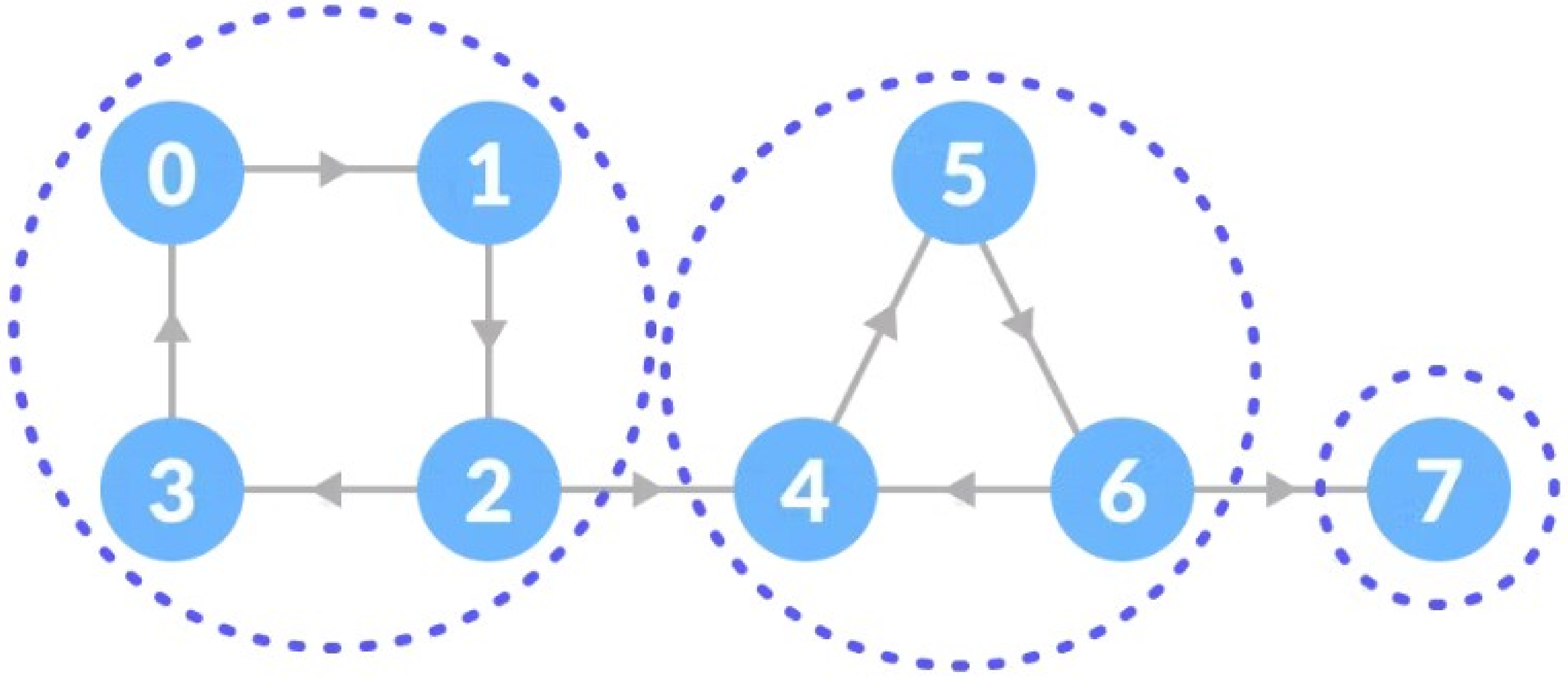
Mentimeter



Sterk sammenhengende grafer

- Gjelder for rettede grafer
- Definisjon: En rettet graf er sterkt sammenhengende dersom det finnes en sti mellom alle par av noder

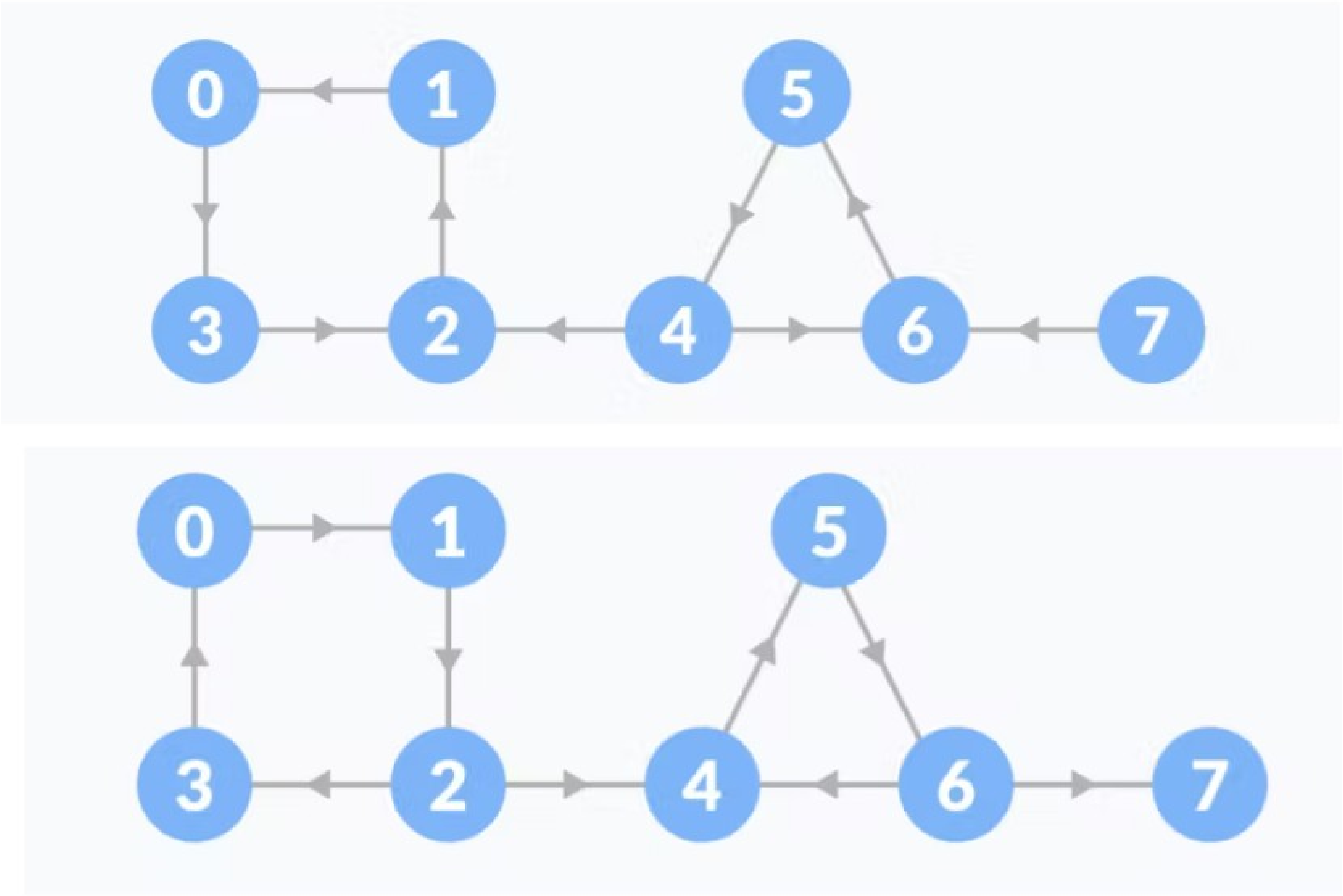
1



Sterk sammenhengende komponenter

- Definisjon: Dersom det finnes en sti mellom alle par av noder i et komponent, så er den sterk sammenhengende
- Basically kan man tenke at et sterkt sammenhengende komponent er en sykel

2



Reverserte grafen

- Den reverserte grafen er definert slik at alle kantene i en graf bytter retning

Å finne sterk sammenhengende komponenter

Rent intuitive så kjører du DFS på den original grafen G
Deretter kjører du DFS på den reverserte grafen.

Dere finner gode illustrasjoner her:
https://www.programiz.com/dsa/strongly-connected-components

Kosaraju

- → Gjør et dybde først søk
- → Når en node er visited, så legges det til en stack
- → På slutten av DFS søket vil vi ha en stack i rekkefølgen nodene ble visited
- → Deretter reverserer vi grafen, og gjør et nytt DFS søk

ALGORITHM: FINN DE STERKT SAMMENHENGENDE KOMPONENETENE AV EN GRAF

Input: En rettet graf $G = (V, E)$

Output: Returner de sterkt sammenhengende komponenetene til G

```
Procedure StronglyConnectedComponents(G)
    stack ← DFSTopSort(G)
    G_r ← ReverseGraph(G)
    visited ← empty set
    components ← empty set
    while stack is not empty do
        u ← stack.pop()
        if u ∉ visited then
            component ← empty set
            DFSVisit(G_r, u, visited, component)
            add component to components
    return components
```

Kosaraju

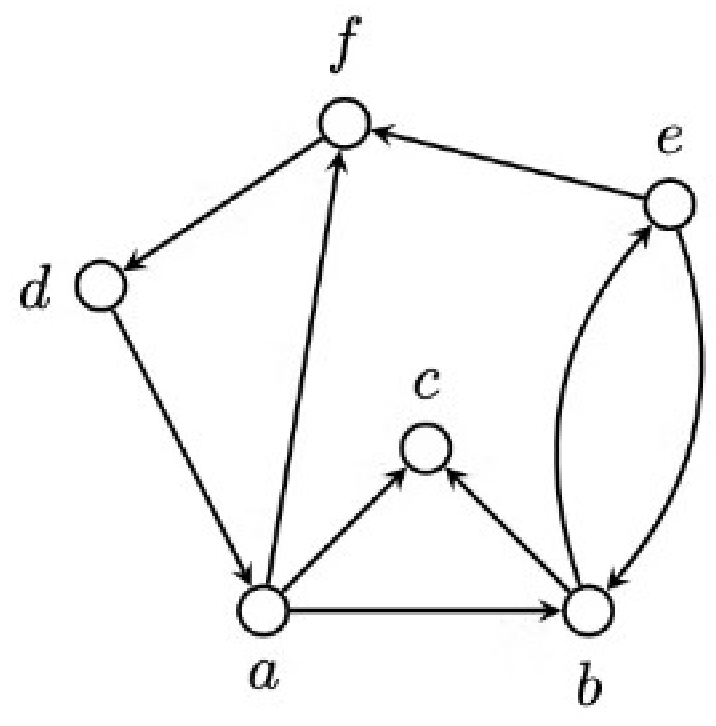
Oppgaver

Eksamen 2019

Mentimeter

1b SCCs i grafer (vekt 4%)

Anta gitt den rettete grafen G med noder $a, \ldots, f$ i Figur 1.



Figur 1: Rettet graf

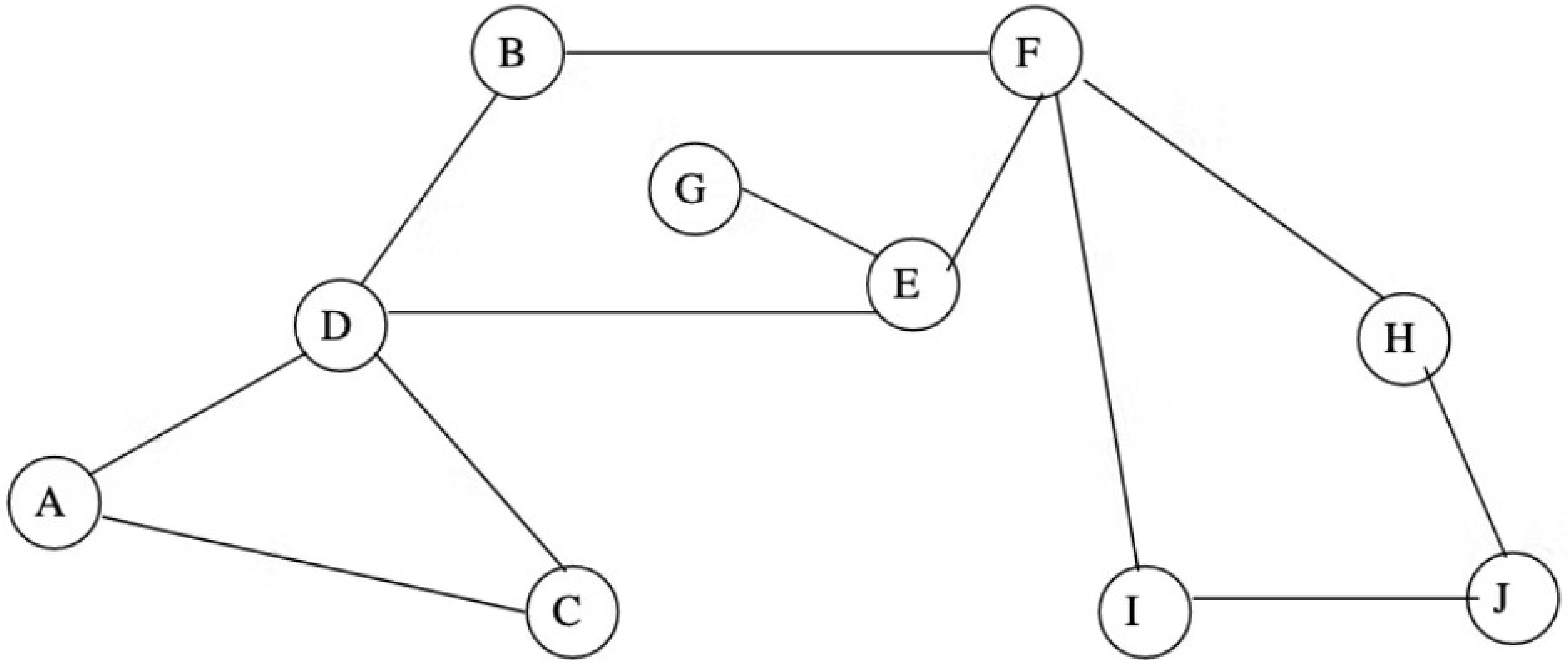
1. Hvilke sterkt sammenhengende komponenter (SCCer, strongly-connected components på engelsk) har vi i G? Bare lag en liste.
2. Vis hvordan SCCs er bestemt algoritmisk. Gi trinnene i algoritmen. Ett trinn skal tilsvare å følge en kant i grafen mellom traversering, ikke mer detaljert enn det.
3. Anta nå en *u*rettet graf. Beskriv (ingen kode er nødvendig) hvordan man kan bestemme SCCs av urettede grafer på en måte som er enklere enn måten for rettet grafer?[1] Innebærer denne forenklingen også en forbedring med henblikk på worst-case tidskompleksitet? Forklar kort.

Eksamen 2012

2

3a Biconnectivity (vekt 4%)

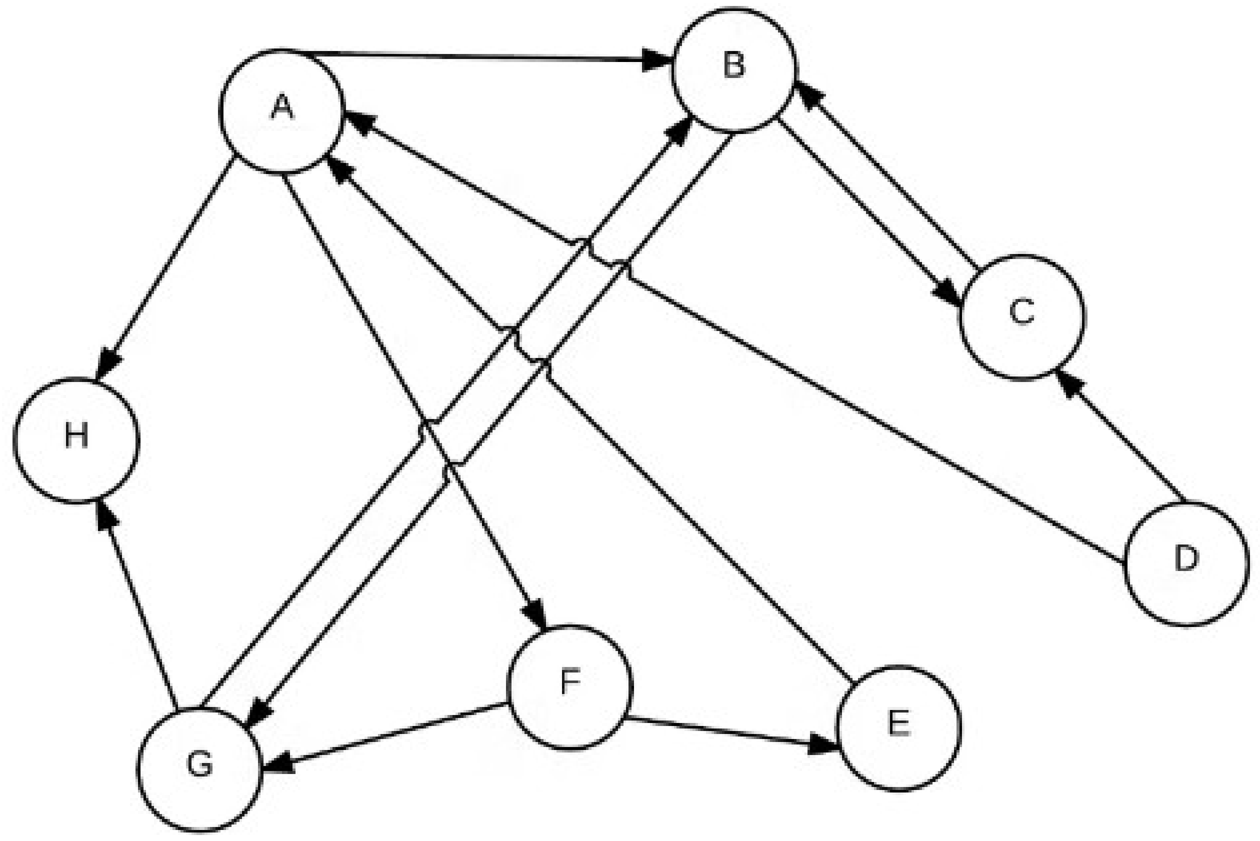
Finn alle *articulation points* for grafen under. Vis et dybde først-spenntre som starter fra node A samt Num- og Low- numrene for hver node.



Eksamen 2013

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.

Eksamen 2014

3

Spørsmpl

Mentimeter

4 questions

2 upvotes