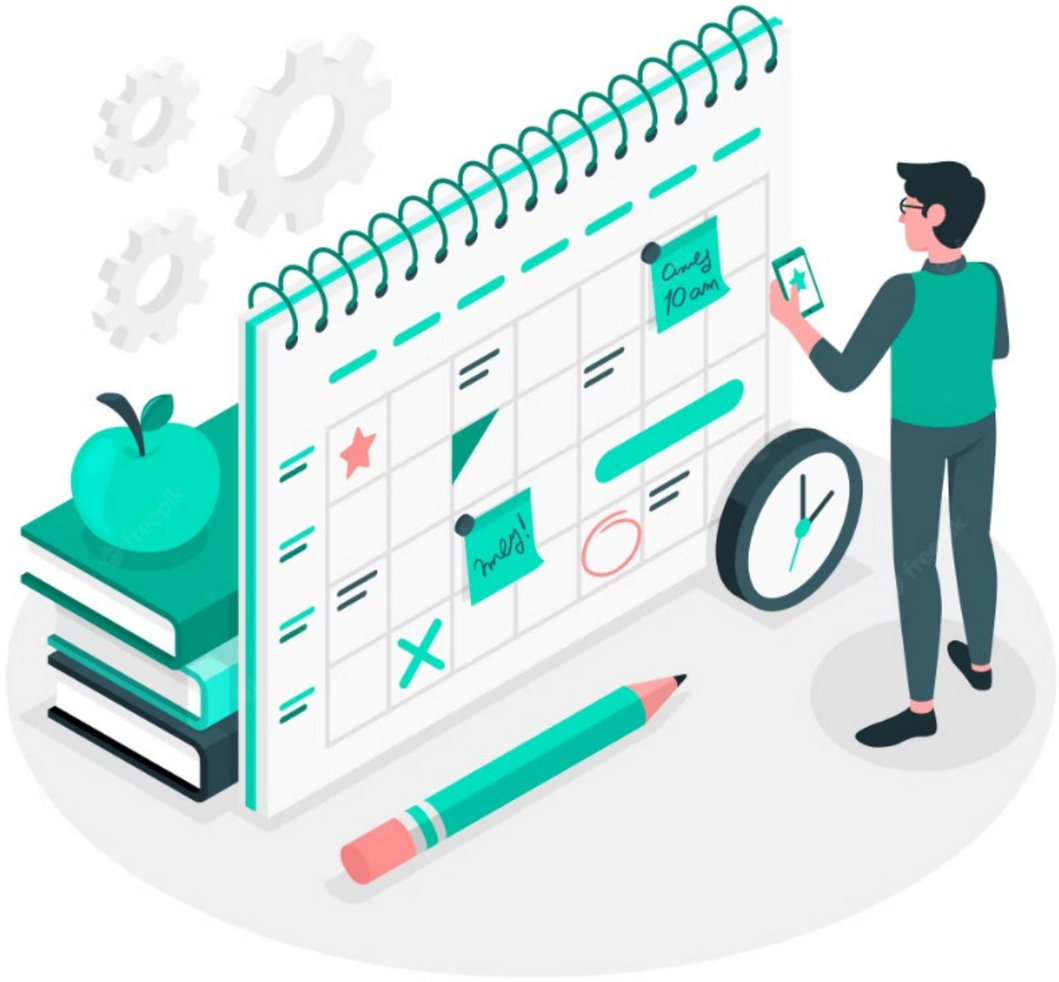
IN2010 - Gruppe 5 & 8

Uke 11 - Kompleksitet og Berenbarhet

Mentimeter

Bli med!

4



Dagens Plan

- → Eksamenstips og 2010-konkuranse
- → Pensumgjennomgang
- → Gruppeoppgaver
- → Siste gruppetimer

Tips til eksamensøving

Intuisjon: Å forstå ideene og konseptene er viktigere enn å pugge de.

Se sammenhengen mellom problem, og gjenkjenne når en algoritme skal brukes.

Løsningene til eksamensoppgaver er som oftest algoritmer fra pensum, men du må selv innføre en liten endring for at du skal løse oppgaven helt. Typ kattis

https://github.com/amaduswaray/IN2010-Gruppe-5

Mentimeter

Pensumgjennomgang

Alt før nå:

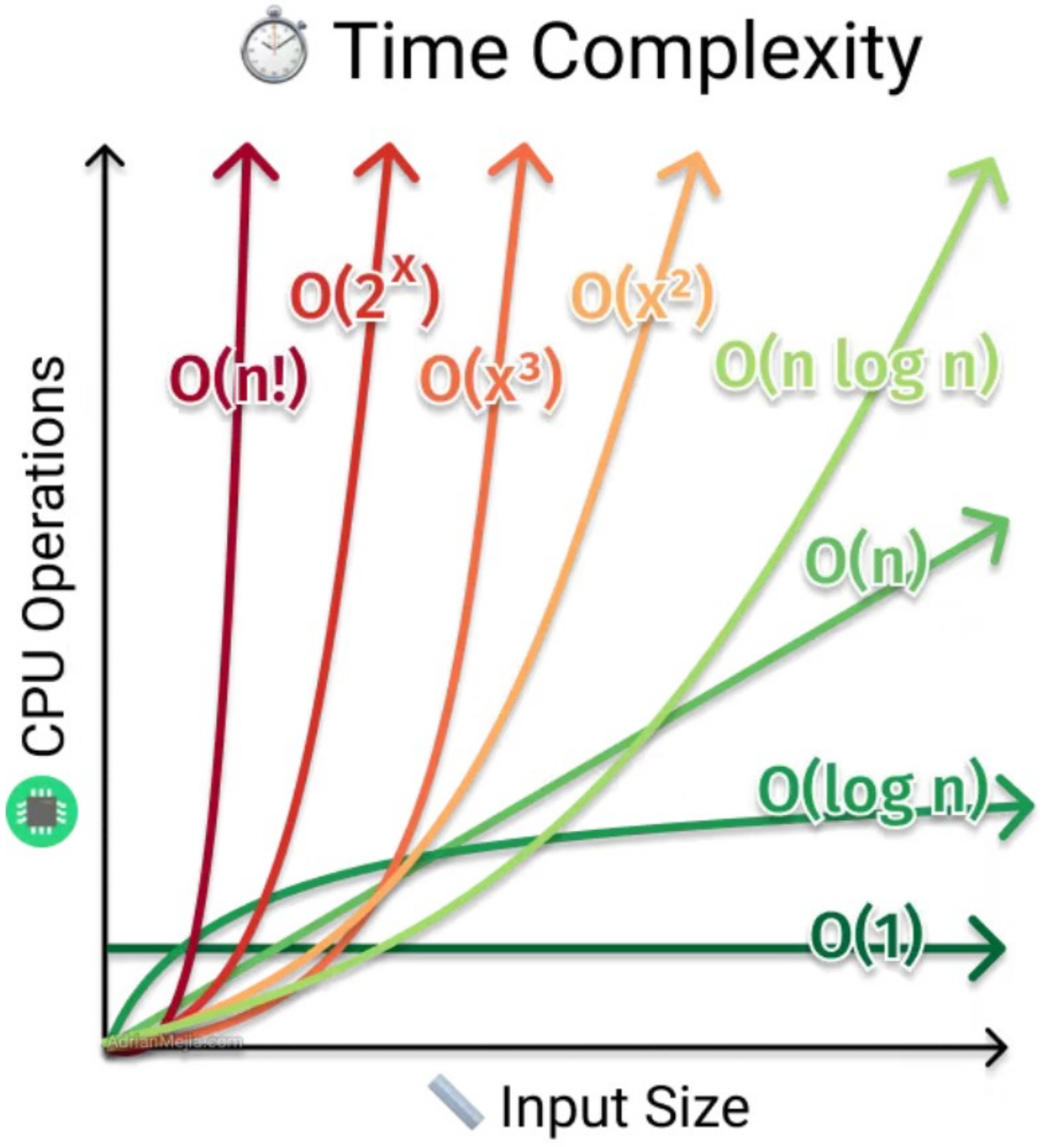
- Løsninger
- Algoritmer
- Det er noe mer abstrakt med det hele
- Det er viktig å være klar over det skille på "løsninger" og "problemer

Avgjørelsesproblemer

- Et avgjørelsesproblem er basically et problem/spørsmål som vi kan avgjøre om det er mulig å løse eller ikke.
- Svaret/outputten på avgjørelsesproblemer er entan ja eller nei
- Eksempel: Er det mulig sortere en liste?
- Eksempel: Er det mulig å telle alle i klasserommet?

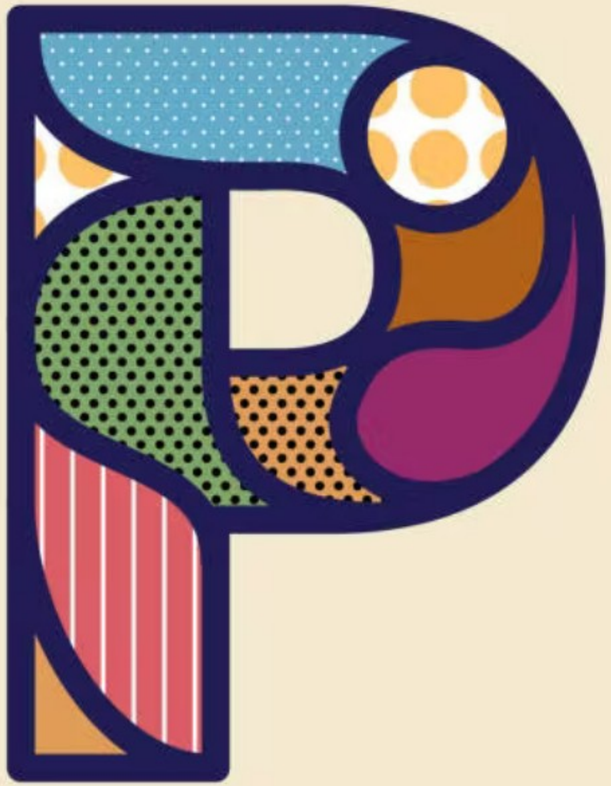
Løse avgjørelsesproblemer

- Dersom et avgjørelsesproblem gir oss en JA instans, så betyr det at vi kan løse den.
- Det å løse et problem betyr at det finnes en algoritme som løser det
- Eksempel: Er det mulig å sortere en argoritme? Ja
- Løsning: Hvilken som helts sorteringsalgoritme vi har lært i faget



Men nå er spørsmålet: Hvor fort kan dette problemet løses?

- → Det er her kompleksitetsklassene kommer in
- → Alle algoritmer har en kjøretidskompleksitet
- → Dette kan brukes for å klassifisere problemene de løser

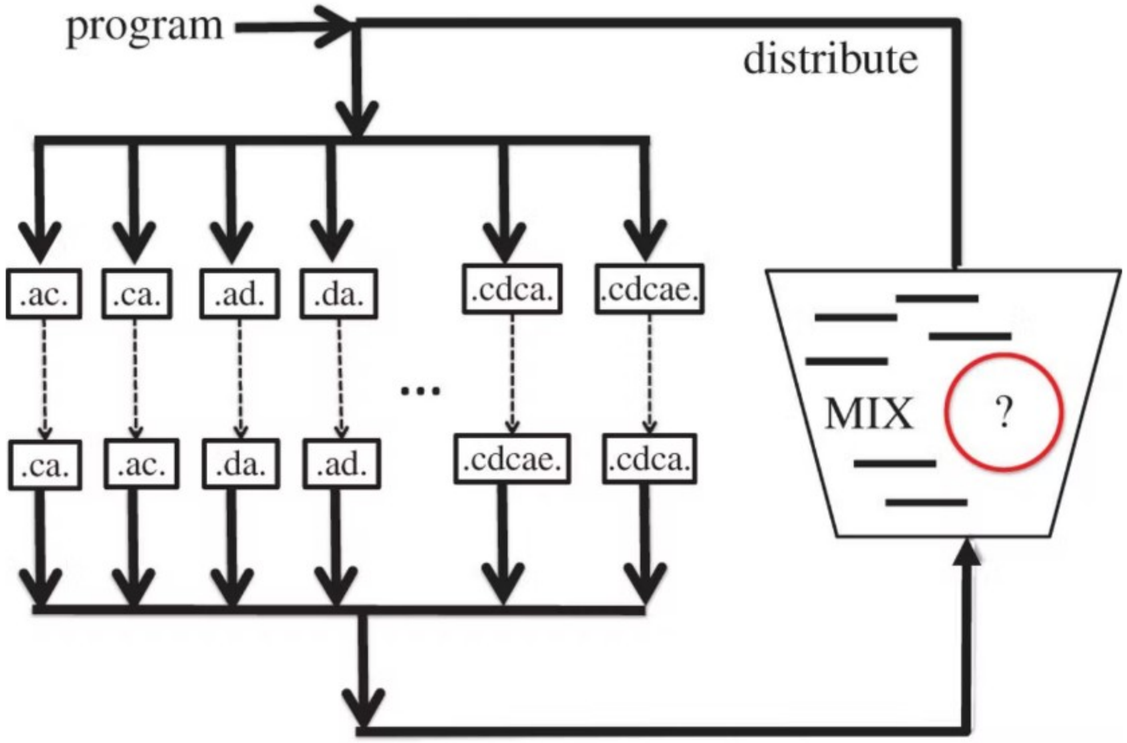


Kompleksitetklassen P:

- Klassen P er en samling av alle avgjørelsesproblemer som kan løses i polynomiell tid
- Polynomiell tid: O(n^p), der p er et polynom
- Nøkkelordet her er at problemet LØSES i polynomisk tid
- Dette gjelder også for alt som kjører raskere

Verifisering av avgjørelsesproblemer

- Gitt en løsning på et problem, så kan du bestemme/verifisere om det er korrekt eller ikke.
- Eksempel: Er det mulig å sortere en liste? Ja
- Hvordan verifiserer vi dette?
- Dersom vi får en sortert liste, så buurde vi si at den er sortert
- Dersom vi får en usortert liste, så burde vi si at den er usortert
- Algoritmer som gjør dette er det vi kaller verifiseringsalgoritmer

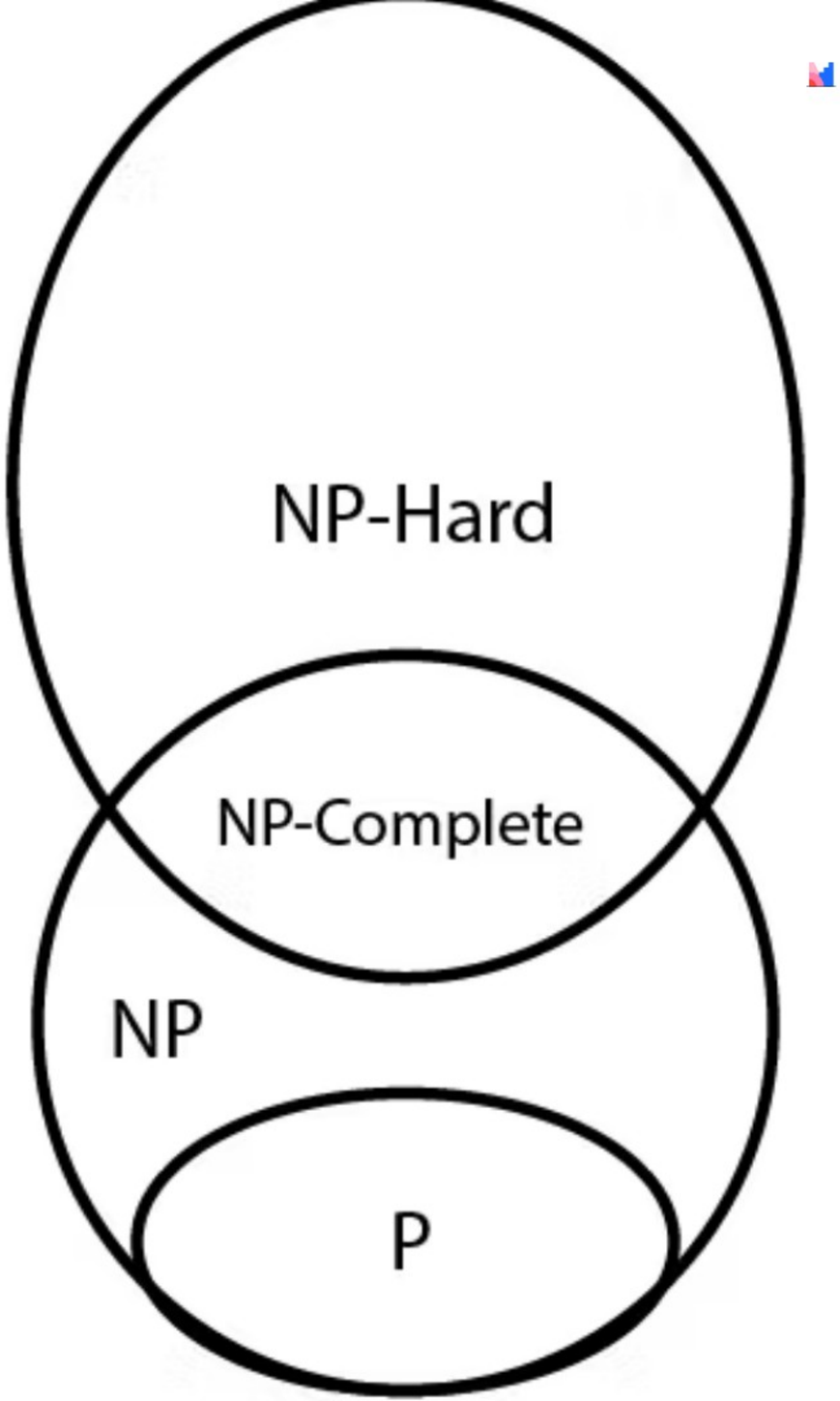


Verifiserings/Ikke deterministiske algoritmer

- Dette er en algoritme som har et sertifikat(fasit) på hva en løsning av problemet skal se ut som
- Eksempel på sertifikat: Sortering
- For å vite at noe er sortert så må elementene være i en ordnet rekkefølge.
- Dersom alle elementene er ordnet så kan vi returnere JA
- Ellers return nei
- Eksempel: Oblig 1 BST checker

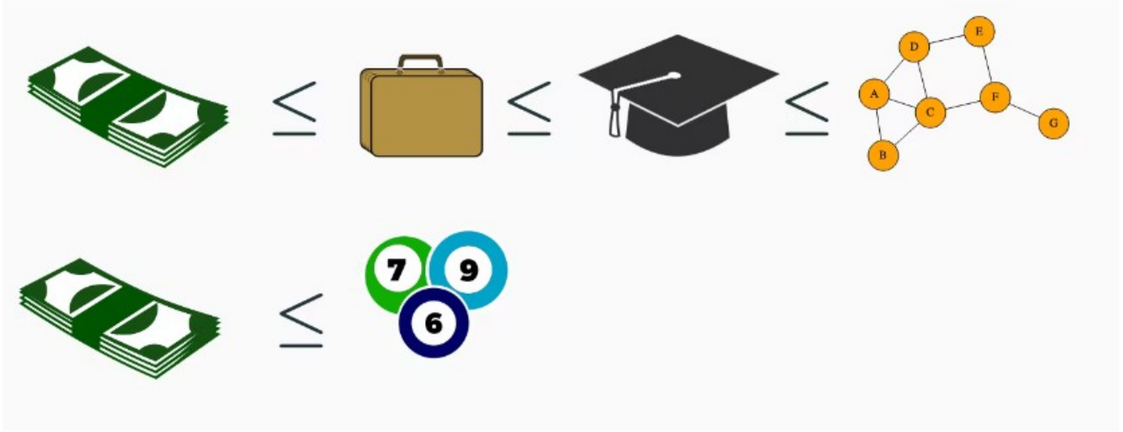
Kompleksitetsklassen NP

- Et problem som kan verifisere(løses av en ikke deterministisk algoritme)
- Den ikke deterministiske algoritmen må kjøre i polynomiell tid.
- NP = Non-Determenistic polynomial time



P og NP

- Om et problem kan løses i polynomiell tid, så kan den også verifiseres i polynomiell tid.
- Alt i P er også i NP
- Men alt i NP er ikke i P
- Det er ikke bevist at P=NP
- Men det er også ikke bevist at P≠NP



Polynomtidsreduksjoner

- Dette handler om å transformere et problem til et annet
- En reduksjonsfunksjon transofrmerer en instans fra det ene problemet, og setter det i det andre.
- slik at hvis en instans n gir ja for det ene problemet, så vil den transformerte/reduserte instansen n^t gi et ja instans for det andre problemet

Even	
Instans:	Et tall n
Spørsmål:	Er n et partall?

Odd	
Instans:	Et tall n
Spørsmål:	Er n et oddetall?

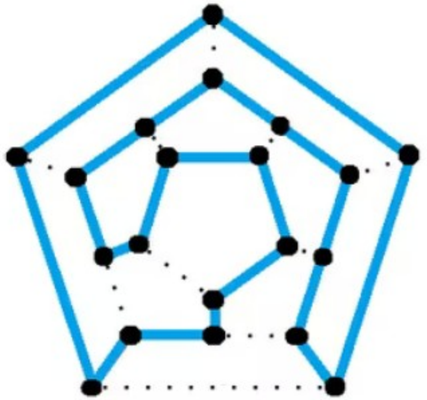
Eksempel

Hamilton sykel og Hamilton Sti

Hamilton sykel:

"Finnes det en sykel der ingen noder blir gjentatt?"

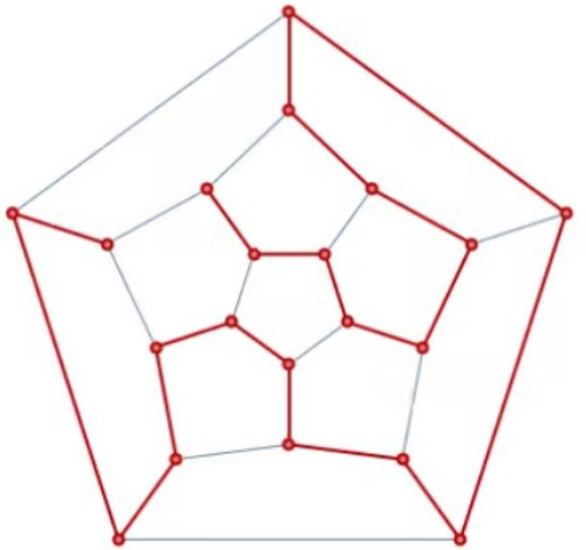
Input: En graf G



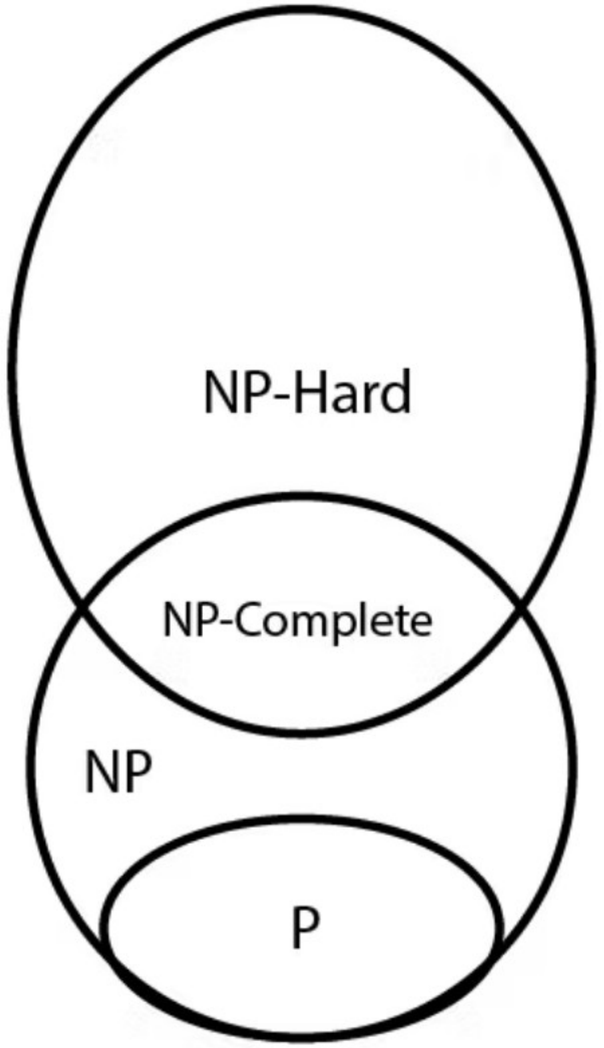
Hamilton sti:

"Finnes det en sti der ingen noder blir gjentatt?"

Input: En graf G



Eksempel 2



De vanskeligste problemene i NP

- De vanskligste problemene i NP er definert slik at alle probeleme i NP kan polynomreduseres til det
- Gitt et Problem L, hvis alle problemer i NP kan reduseres til L, så er L et av de vanskeligste problemene
- Hvis L nå også er i NP, så er l dermed NP-Komplett

Vise NP-Kompletthet

- Gitt et problem L
- 1. Vise at L er i NP(Kan verifiseres i polynomisk tid)
- 2. Redusere et kjent NP-Komplett problem til L

Gruppeoppgaver

Litt om beregnbarhet og kompleksitet 8 poeng

For alle spørsmålene nedenfor, svar **Sant** eller **Usant**.

(a) Det er bevist at $P = NP$.

(b) Det er bevist at $P \neq NP$.

(c) Alle NP-komplette problemer kan polynomtidsreduseres til hverandre.

(d) Alle problemer i P er også i NP.

(e) Alle avgjørelsesproblemer er enten i P eller NP.

(f) Alle problemer hvor JA-instanser kan verifiseres i polynomiell tid, er i NP.

(g) Hvis noen klarer å løse avgjørelsesproblemet **Hamiltonsykel**, **Knapsack** eller **Sudoku** i polynomiell tid, så har de bevist at $P = NP$.

(h) Alle avgjørelsesproblemer kan løses med en rask nok datamaskin.

Eksamen 2021

NP-kompletthet (6 poeng)

a) Anta at du vet at et problem A er NP-komplett og at et problem B er i NP, og at du skal vise at også problem B er NP-komplett. Må du da vise at A kan reduseres til B eller at B kan reduseres til A? Begrunn svaret.

b) I et selskap skal det plasseres N gjester ved et rundt bord. Vi vet hvilke par av gjester som trives sammen, og hvilke som ikke gjør det. Dette er angitt ved hjelp av en boolsk NxN-matrise (som er symmetrisk). Vi ønsker å finne ut om det er mulig å lage en bordplassering slik at alle som sitter ved siden av hverandre trives sammen.

Vi tenker oss av både verdien N og trives-matrisen, kan variere fritt. Du kan anta som kjent at dette problemet er i NP, og din oppgave er å vise at det også er NP-komplett. Du skal bruke ett av de NP-komplette problemene fra pensum (VERTEX-COVER, SUBSET-SUM, HAMILTONIAN-CYCLE og TSP) som utgangspunkt for beviset.

Eksamen 2018

Exercise 2 (NP-completeness)

Let **HC** (Hamiltonian-cycle) = {<G> | The directed graph G contains a Hamiltonian cycle}.

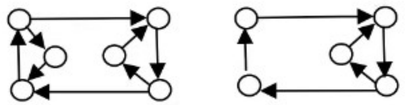
We define it in such a way that graphs with only one vertex, and graphs with two vertices connected to each other, are YES-instances of **HC**.

The problem **HC** is known to be **NP**-complete.

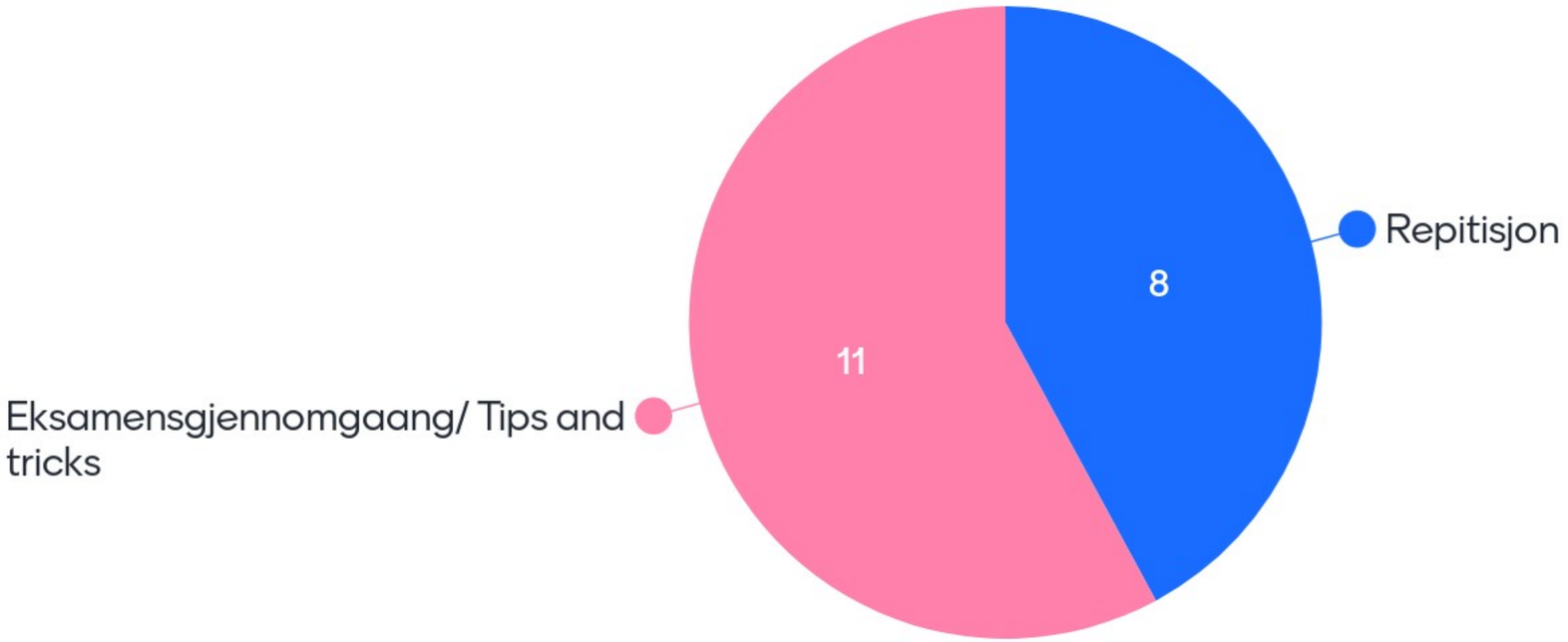
Prove that **2CC (**2-Cycle-cover**),** defined below, is **NP**-complete.
Describe the reduction function *f*, and prove that it is correct. Argue that the reduction is polynomial.

2CC
Input: A directed graph *G*.
Question: Are there two directed, simple, non-overlapping cycles in the graph, each with at least three vertices, that together include all the vertices of *G*? A cycle is simple if it never visits a vertex more than once.

Example
The following graphs show a YES-instance and a NO-instance of **2CC**, respectively:



Siste gruppetimer



Tema som må gås gjennom