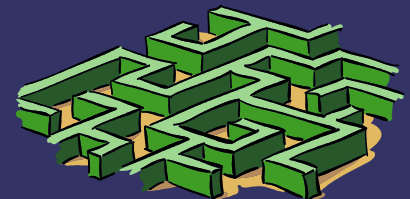


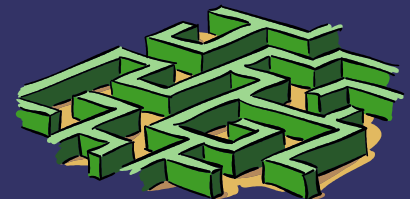
NP-komplette problemer

Simon Jonassen
TDT4120 Algoritmer og Datastrukturer
IDI, NTNU



“There’s no problem so large it can’t be solved by killing the user off, deleting their files, closing their account and reporting their REAL earnings to the IRS”

Simon Travaglia, BOFH



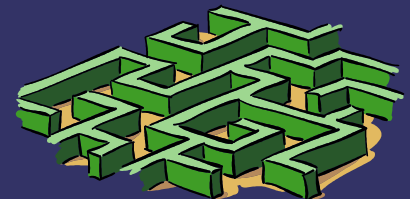
halting problem

- Gitt en algoritme p og en sekvens i .
- $\text{halt}(p,i)$ – returnerer *true* hvis $p(i)$ terminerer, eller *false*



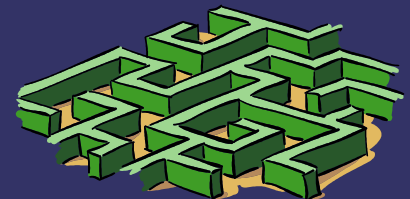
halting problem

- $\text{halt}(p,i)$ – returnerer *true* hvis $p(i)$ terminerer, eller *false*
- $\text{trouble}(t)$:
 if halt(t,t) do_inf_loop
 else return 1
- $\text{trouble}(\text{trouble})$?!?



Motivasjon

- *Det finnes problemer som ikke kan løses, det finnes problemer som ikke kan løses raskere enn i $O(k^n)$.*
- *Vi ønsker å generalisere hvilke problemer kan løses i polynomisk tid, $O(n^k)$.*



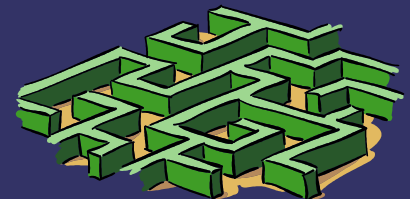
Problemreduksjon

- Et problem A er reduserbar til et problem B hvis det finnes en algoritme som kan oversette alle instanser av A til instanser av B, og et svar for en instans av B kan oversettes til et svar for tilsvarende instanser av A.
- Vi sier at problem B er minst like vanskelig som problem A.



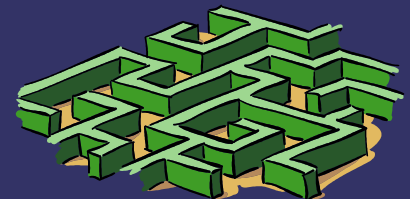
Problemreduksjon

- Hvis et problem A er reduserbar til et problem B i polynomisk tid og problem B kan løses i polynomisk tid, kan også A løses i polynomisk tid.
- $P(B) \rightarrow P(A)$
- Hvis et problem A er reduserbar til et problem B og problem A ikke kan løses i polynomisk tid, kan ikke B løses i polynomisk tid heller!
- $\sim P(A) \rightarrow \sim P(B)$

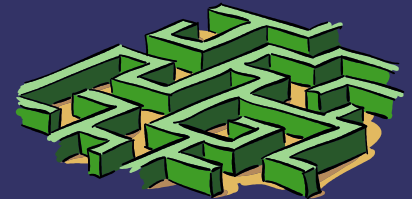
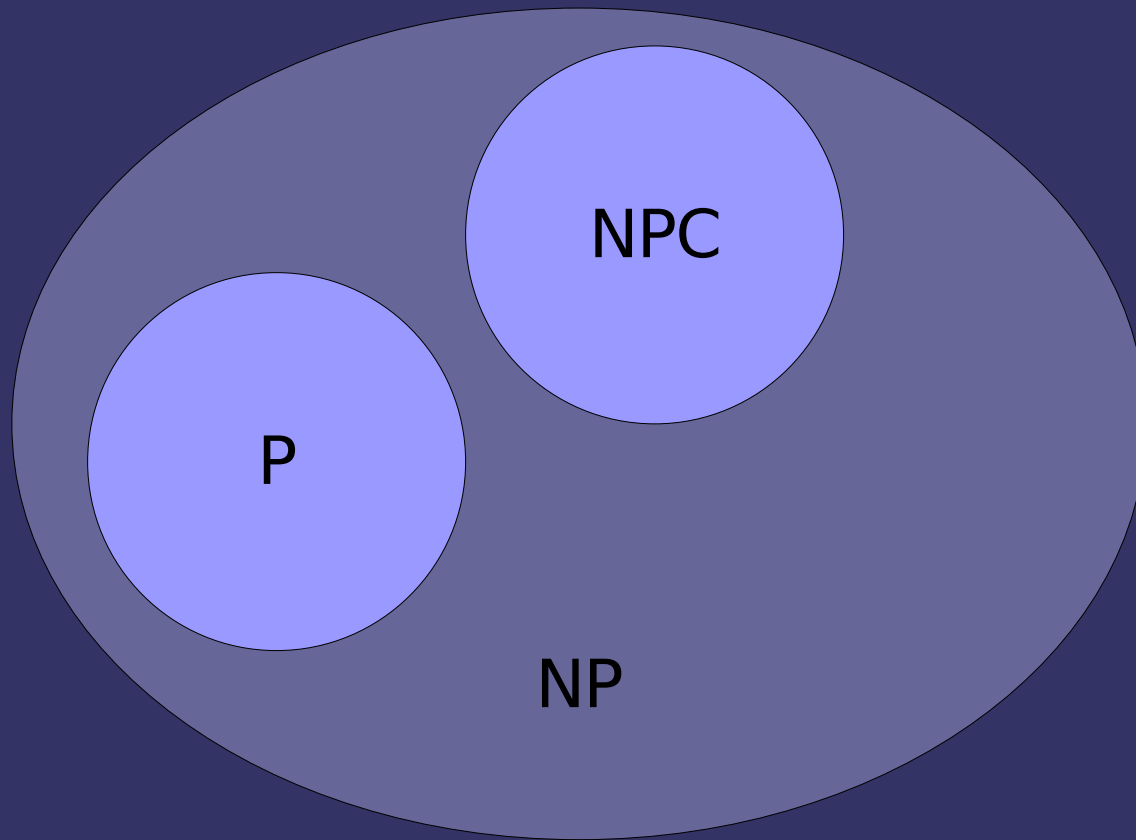


Problemlasser

- NP :- alle problemer som kan verifiseres av en polynomisk algoritme.
- P : - alle problemer som kan løses av en polynomisk algoritme.
- NPC :- alle problemer i NP som er minst like vanskelige som alle andre problemer i NP.



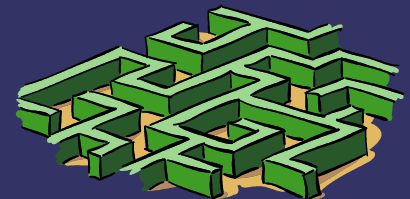
Problemklasser



Problemlasser

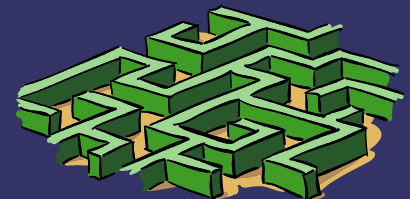
Reduksjonsresultater:

- Hvis det finnes minst ett problem i NPC som kan løses i polynomisk tid, kan alle problemer i NP løses i polynomisk tid.
- Men!!! Hvis det finnes minst ett problem i NP som ikke kan løses i polynomisk tid, så kan ingen NPC-problem løses i polynomisk tid.



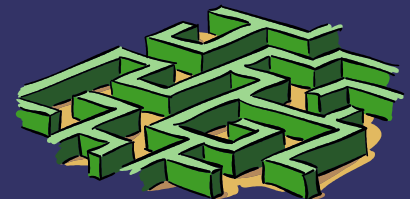
C-SAT

- Circuit Satisfiability : Gitt en boolsk kombinatorisk krets, finn en kombinasjon av inputverdier som gir 1 som output.
- $f(x_1, \dots, x_n) = 1$
- $O(2^n)$ kombinasjoner



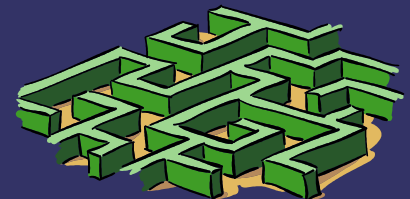
Hvordan kan vi avgjøre om et problem er i NP?

- Alternativ 1: Finn en algoritme som kan verifisere en mulig løsning til dette problemet.
- Alternativ 2: Vis at problemet er i P i stedet.



Hvordan kan vi avgjøre om et problem er i P?

- Alternativ 1: Finn et problem i P som dette problemet kan reduseres til i polynomisk tid. Løs det andre problemet i stedet.
- Alternativ 2: Lag en algoritme som kan løse dette problemet i polynomisk tid og bevis at resultatet blir riktig.



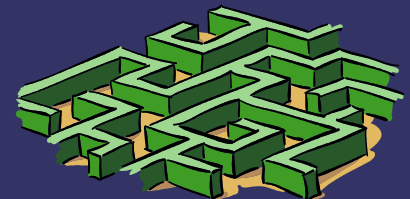
Hvordan kan vi avgjøre om et problem er i NPC?

- -Alternativ 1: Vis at problemet er i NP. Vis at problemet er *NP-hard*, altså minst like vanskelig som alle andre problemer i NP.

Eksempel: C-Sat

- - Alternativ 2: Finn et problem i NPC som kan reduseres til dette problemet .

Eksempel: C-Sat kan reduseres til Sat, som kan reduseres til 3CNF-Sat, som ...



*“All good things in life are either illegal,
immoral, or NP-Complete”*

