

13. december

\mathcal{NP} -fuldstændighed

Datalogisk indsigt

- Der findes problemer som kan løses effektivt (polynomiel tid)
- Der findes problemer som ikke kan løses effektivt
- Der findes problemer som slet ikke kan løses

hvorfor?

- $\mathcal{NP} = \mathcal{P}$ er et af datalogiens største åbne problemer
- Det vrimler med problemer som ikke kan løses effektivt
- Det er ikke nemt at se forskel på svære og lette problemer
- Reduktion (transformation) er generelt anvendelig
- ($\mathcal{NP} = \mathcal{P}$ forekommer i “The Simpsons”)

Datalogiens største spørgsmål

Hvis kan bevise $\mathcal{NP} = \mathcal{P}$ fås 1 million dollar

- <http://www.claymath.org/MillenniumPrizeProblems/>



CLAY MATHEMATICS INSTITUTE

Dedicated to increasing and disseminating mathematical knowledge

MILLENNIUM PRIZE PROBLEMS

Statement from the Directors and Scientific Advisory Board

| Birch and Swinnerton-Dyer Conjecture | Hodge Conjecture | Navier-Stokes Equations | P vs NP |
Poincare Conjecture | Riemann Hypothesis | Yang-Mills Theory || Rules etc |



In order to celebrate mathematics in the new millennium, The Clay Mathematics Institute of Cambridge, Massachusetts (CMI) has named seven "Millennium Prize Problems." The Scientific Advisory Board of CMI selected these problems, focusing on important classic questions that have resisted solution over the years. The Board of Directors of CMI have designated a \$7 million prize fund for the solution to these problems, with \$1 million allocated to each. During the Millennium meeting held on May 24, 2000 at the Collège de France, Timothy Gowers presented a lecture entitled "The Importance of Mathematics," aimed for the general public, while John Tate and Michael Atiyah spoke on the problems. The CMI invited specialists to formulate each problem.



Millennium Prize
Problem.ram

One hundred years earlier, on August 8, 1900, David Hilbert delivered his famous lecture about open mathematical problems at the second International Congress of Mathematicians in Paris. This influenced our decision to announce the millennium problems as the central theme of a Paris meeting.

The rules that follow for the award of the prize have the endorsement of the CMI Scientific Advisory Board and the approval of the Directors. The members of these boards have the responsibility to preserve the nature, the integrity, and the spirit of this prize.

Paris, May 24, 2000

Please send inquiries regarding the Millennium Prize Problems to prize.problems@claymath.org.



| Birch and Swinnerton-Dyer Conjecture | Hodge Conjecture | Navier-Stokes Equations | P vs NP |

Effektive algoritmer

Garey og Johnson

Lette og svære problemer

\mathcal{P} (let)	\mathcal{NP} -fuldstændig (svær)
<p>Euler cycle Given a graph $G = (V, E)$. Is there a cycle which visits each edge exactly once?</p>	<p>Hamilton cycle Given a graph $G = (V, E)$. Is there a cycle which visits each node exactly once?</p>
<p>Shortest path Given a weighted graph $G = (V, E, c)$. Is there a path from a to b with length $\leq k$?</p>	<p>Longest path Given a weighted graph $G = (V, E, c)$. Is there a path from a to b with length $\geq k$?</p>
<p>Edge cover Given a graph $G = (V, E)$. Is it possible to choose $\leq k$ edges so every node is incident to a chosen edge?</p>	<p>Node cover Given a graph $G = (V, E)$. Is it possible to choose $\leq k$ nodes so every edge is incident to a chosen node?</p>
<p>Chinese postman Given a weighted graph $G = (V, E, c)$. Is there a cycle which visits each edge at least once and has length $\leq k$?</p>	<p>Traveling salesman Given a weighted graph $G = (V, E, c)$. Is there a cycle which visits each node at least once and has length $\leq k$?</p>
<p>2CNF-SAT Given a boolean expression in 2CNF-form. Is it possible to assign truth values to variables so expression becomes true?</p>	<p>3CNF-SAT Given a boolean expression in 3CNF-form. Is it possible to assign truth values to variables so expression becomes true?</p>

Oversigt

- Hvad er \mathcal{P} , \mathcal{NP} og \mathcal{NP} -fuldstændighed
- Eksistens af \mathcal{NP} -fuldstændigt problem
- Bevis af \mathcal{NP} -fuldstændighed ved reduktion
- Grænsen mellem \mathcal{NP} -fuldstændigt og \mathcal{P}

Løsning af \mathcal{NP} -fuldstændige problemer

- Branch-and-bound
- Approximation
- (Meta)heuristikker

Formalisme

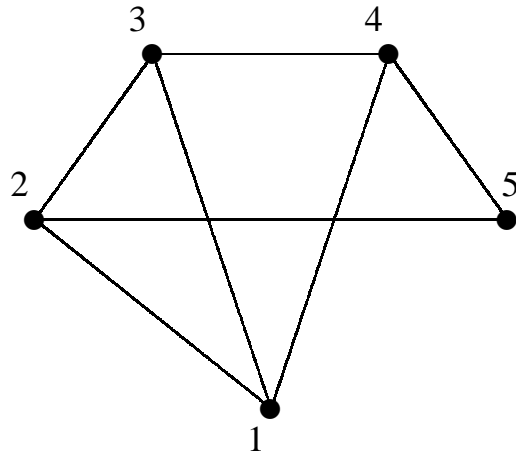
Nemt at nå fejlagtige konklusioner. Derfor stringent formalisme.

Afgørlighedsproblemer

Abstrakt problem

Probleminstans

SHORTEST-PATH

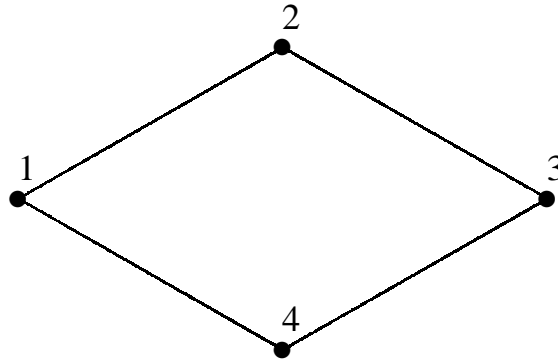


- Optimeringsproblem for generelt
- Afgørlighedsproblem
- Nemt at transformere "løsninger"
- Intet tab af generalitet

Afgørlighedsproblem

PATH = { $\langle G, u, v, k \rangle$:
 $G = (V, E)$ is an undirected graph
 $u, v \in V$,
 $k \geq 0$ is an integer,
there exists a path from u to v in G
whose length is at most k }

Alfabet: { ciffer, "{", "}", ",", " " }



Instans:

$I = \langle \{1, 2, 3, 4\}, \{\{1, 2\}, \{2, 3\}, \{3, 4\}, \{1, 4\}\}, 1, 3, 2 \rangle$

Afgørlighedsproblemet:

PATH(I) = 1?

Probleminstans, effektiv kodning

Ideelt set skulle vores argumentation være uafhængig af inddataformat. Men inddataformat har betydning.

Eksempel

Primtalstest

- Input: tal k
- For $i = 1$ to k do “prøv om $i|k$ ”

Kodning af input

unær kodning	$n = \Theta(k)$	$\Theta(n)$
binær kodning	$n = \Theta(\log_2 k)$	$\Theta(2^n)$
decimal kodning	$n = \Theta(\log_{10} k)$	$\Theta(10^n)$
unit cost kodning	$n = \Theta(1)$?

Standard kodning $\langle G \rangle$

Effektiv (binær eller polynomielt relateret)

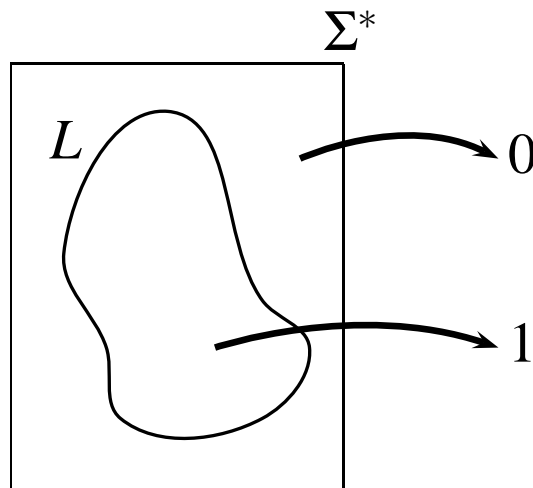
Polynomielt relaterede kodninger

Krav: findes polynomiell tidsfunktioner som afbilder instanser mellem de to kodninger.

Afgørlighedsproblemer som sprog

- Afgørlighedsproblem: instans $\rightarrow \{0, 1\}$
- Instans: alfabet (“{” “}” “,” tal)
- Binært alfabet: $\Sigma = \{0, 1\}$
- $\Sigma^* = \{\epsilon, 0, 1, 00, 01, 10, 11, 000, \dots\}$
- Afgørlighedsproblem er et *sprog* L

$$L = \{x \in \Sigma^* \mid \text{problem}(x) = 1\}$$

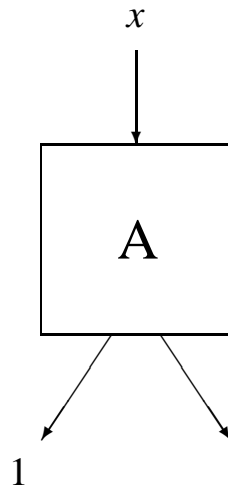


Note

Hvis streng $x \in \Sigma^*$ ikke repræsenterer en instans (“syntax-fejl”) returneres 0.

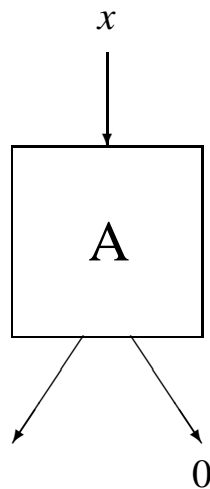
Genkendelse med algoritmer

Streng $x \in \Sigma^*$ *accepteres (genkendes)* af algoritme A



A standser med værdien 1

Streng $x \in \Sigma^*$ *afvises* af algoritme A



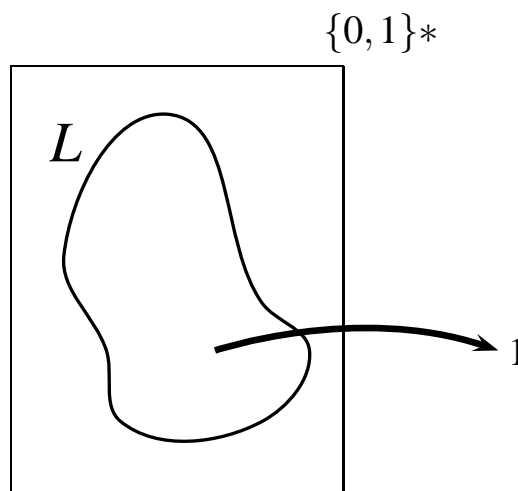
A standser med værdien 0

Genkendelse med algoritmer

Sprog $L =$ afgørlighedsproblem

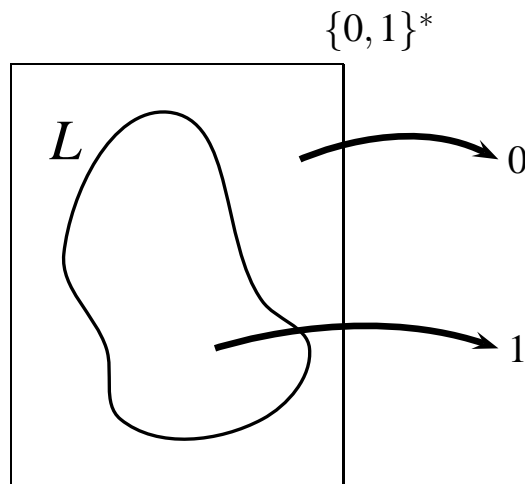
Sprog L accepteres (genkendes) af algoritme A

$$\forall x \in L : x \text{ accepteres af } A$$



Sprog L afgøres af algoritme A

$$\forall x \in \{0,1\}^* : x \text{ accepteres eller afvises af } A$$



Accept i polynomiell tid

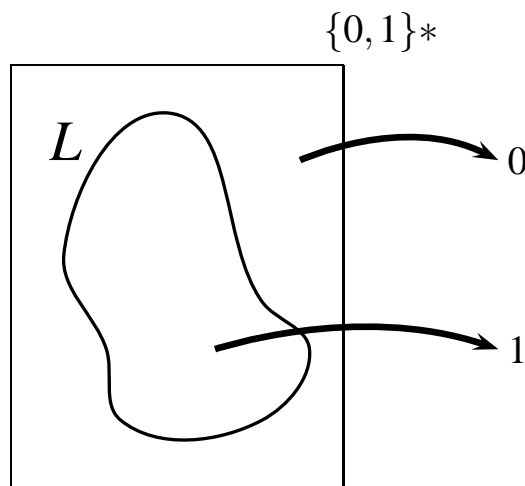
L accepteres i polynomiell tid af en algoritme A

$$\exists k : \forall x \in L, |x| = n : x \text{ accepteres af } A \text{ i tid } O(n^k)$$

L afgøres i polynomiell tid af en algoritme A

$$\exists k : \forall x \in \{0, 1\}^*, |x| = n : x \text{ accepteres eller} \\ \text{afvises af } A \text{ i tid } O(n^k)$$

Klassen \mathcal{P} er mængden af sprog L for hvilke der findes en algoritme A som afgør L i polynomiell tid.



Accept i polynomielt tid

$\begin{aligned} &\text{afgøres polynomielt tid} \\ &x \in L \Rightarrow 1 \\ &x \notin L \Rightarrow 0 \end{aligned}$	\Leftrightarrow	$\begin{aligned} &\text{accepteres polynomielt tid} \\ &x \in L \Rightarrow 1 \end{aligned}$
--	-------------------	--

Husk

L accepteres i polynomielt tid af A

$$\exists c, k, n' : \forall x \in L, |x| = n : x \text{ accepteres af } A \\ \text{i tid } cn^k \text{ for } n > n'$$

L afgøres i polynomielt tid af A

$$\exists c, k, n' : \forall x \in \{0, 1\}^*, |x| = n : x \text{ accepteres eller} \\ \text{afvises af } A \text{ i tid } cn^k \text{ for } n > n'$$

Bevis

\Rightarrow oplagt

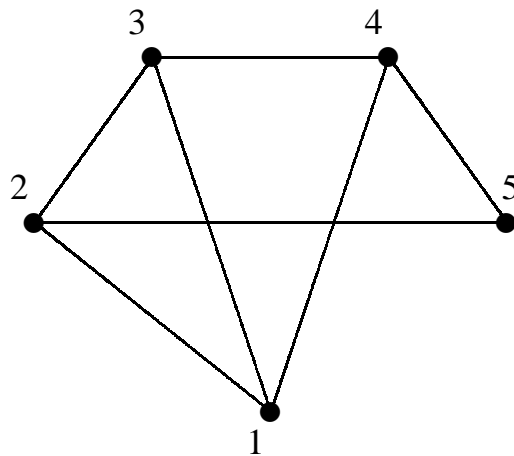
\Leftarrow Antag $n > n'$. For givne konstanter c, k lad algoritme køre i tid cn^k . Hvis ej standset udskriv "0".

Verifikation

- *Verificere* : at bekræfte at en forelagt “potentiel løsning” vitterlig er en løsning.
- *Certifikat* : “ægte” løsning

Eksempel

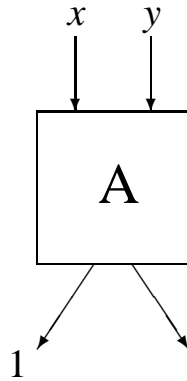
HAM-CYCLE = { $\langle G \rangle$: G har en Hamilton kreds }



- *Certifikat* : liste af knuder (v_1, v_2, \dots, v_n)
- *Verificere* : kontroller (v_1, v_2, \dots, v_n) er Hamilton kreds

Verifikation

Algoritme A : ordinært input x , ekstra input y



A verificerer x hvis der findes y så A accepterer (x, y)

Sproget L verificeret af A er

$$\{x \in L : \exists y \in \{0, 1\}^* \text{ så } A \text{ accepterer } (x, y)\}$$

\mathcal{NP} er klassen af problemer for hvilke der findes polynomielt-tids verificerende algoritme.

$L \in \mathcal{NP}$ hvis og kun hvis der eksisterer polynomielt verifikationsalgoritme A , og konstant c så

$$\forall x \in L, \exists y, |y| = O(|x|^c) : A \text{ accepterer } (x, y)$$

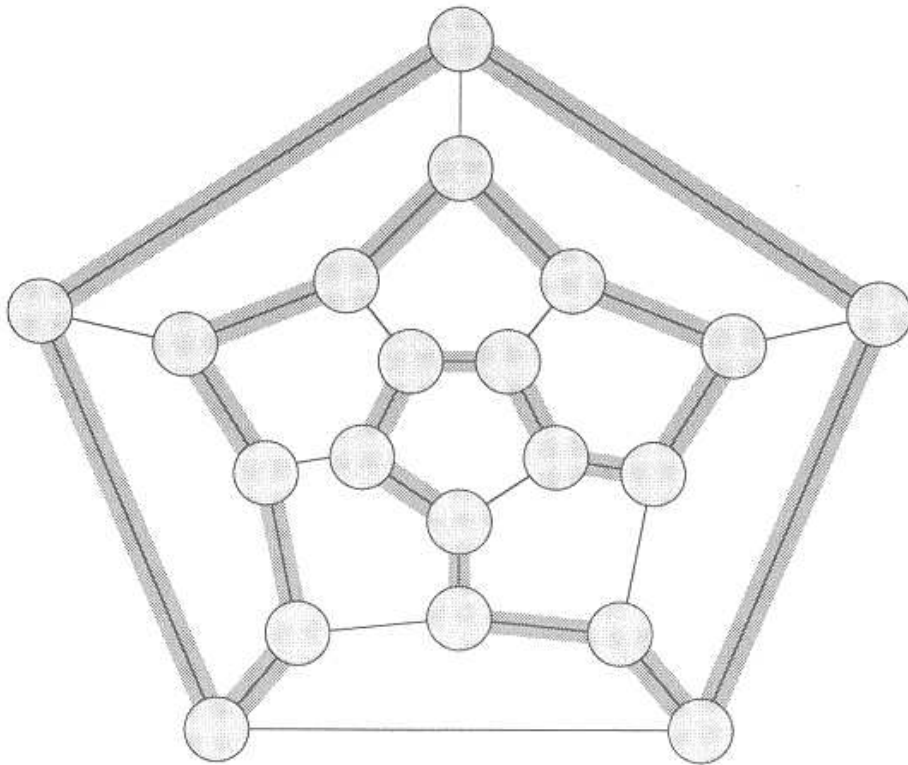
Verifikation

Bemærk

$$\mathcal{P} \subseteq \mathcal{NP}$$

Eksempel

HAM-CYCLE er i klassen \mathcal{NP} .



Nondeterministisk polynomiel

Historisk:

\mathcal{NP} er klassen af problemer der kan løses af en nondeterministisk Turing Maskine i polynomiel tid.

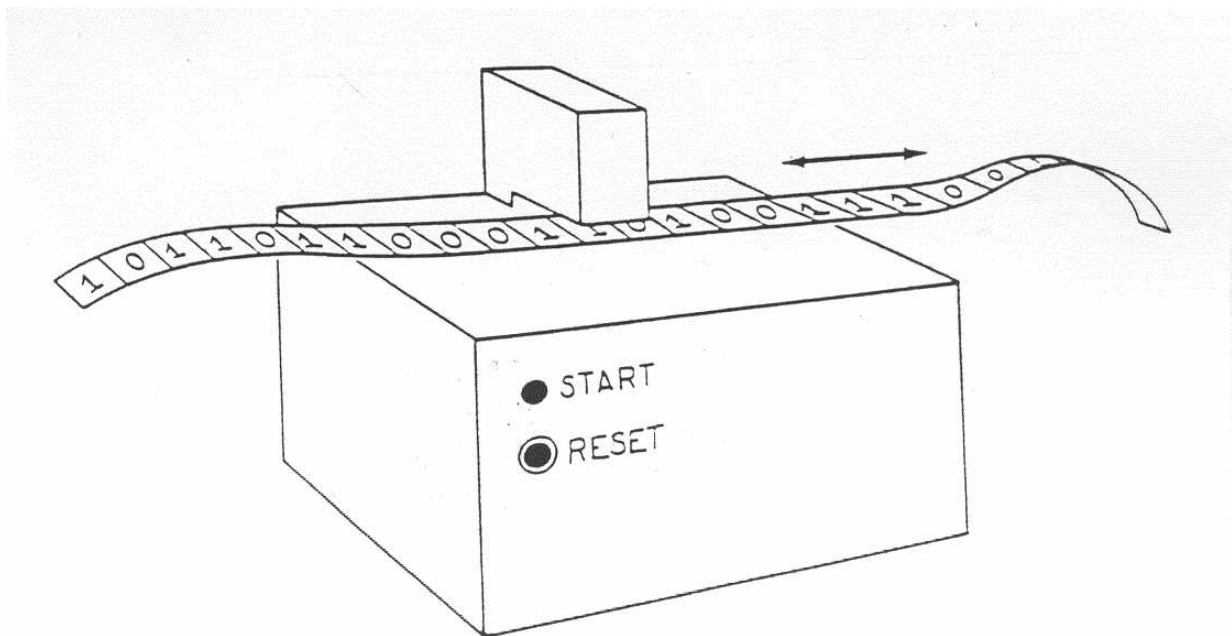


Figure 90 A Turing machine conceptualization

$$\delta(p, a) = (q, b, d)$$

Når i tilstand p læser symbol a : skriv symbol b , flyt i retning d , og antag tilstand q .

$$\delta(p, a) = (q_1, b_1, d_1) \vee (q_2, b_2, d_2) \vee \dots \vee (q_m, b_m, d_m)$$

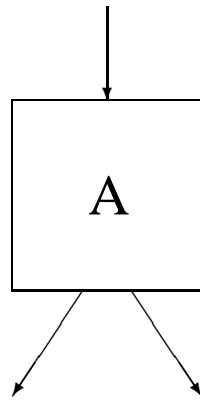
\mathcal{NP} og $\text{co-}\mathcal{NP}$

$\text{co-}\mathcal{NP}$ er mængden af sprog L så

$$\bar{L} \in \mathcal{NP}$$

hvor komplement

$$\bar{L} = \{x \in \Sigma^* \mid \text{problem}(x) = 0\}$$



Eksempel

HAM-CYCLE =

$$\{ \langle G \rangle : G \text{ is a hamiltonian graph} \}$$

HAM-CYCLE-COMPLEMENT =

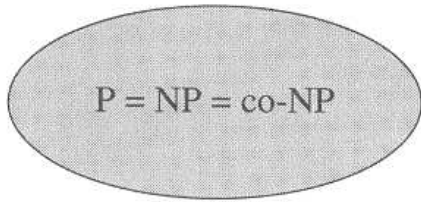
$$\{ \langle G \rangle : G \text{ is a not a hamiltonian graph} \}$$

formentlig ikke i \mathcal{NP} .

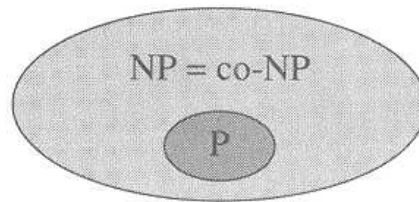
Certifikat?

\mathcal{NP} og $\text{co-}\mathcal{NP}$

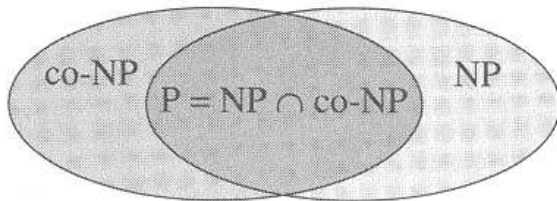
Oplagt at $\mathcal{P} \subseteq \text{co-}\mathcal{NP}$.



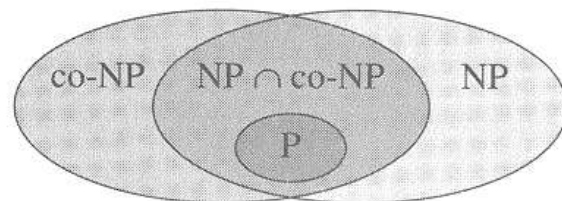
(a)



(b)



(c)



(d)

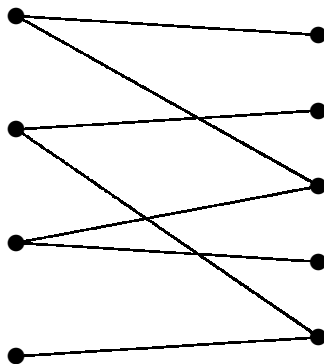
Givet et \mathcal{NP} -fuldstændigt problem L .
Hvis $\bar{L} \in \mathcal{NP}$ så er $\mathcal{NP} = \text{co-}\mathcal{NP}$.

Reduktion

At løse et problem v.h.a. et andet

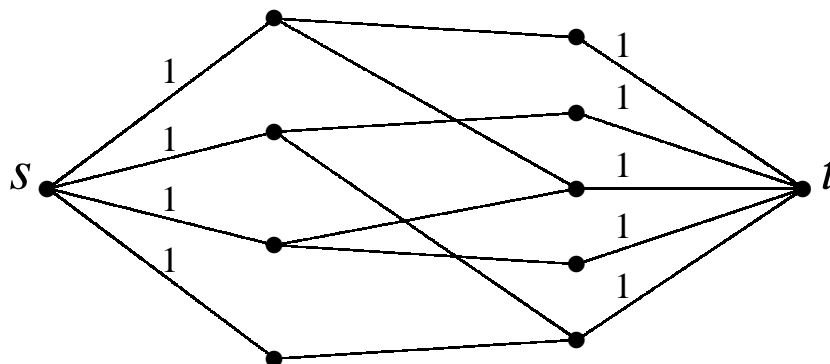
Eksempel

Parring i todelt graf: BIP-MATCH



Find en parring M (i.e. en delmængde af kanter så ingen har fælles endepunkter) med flest mulige kanter.

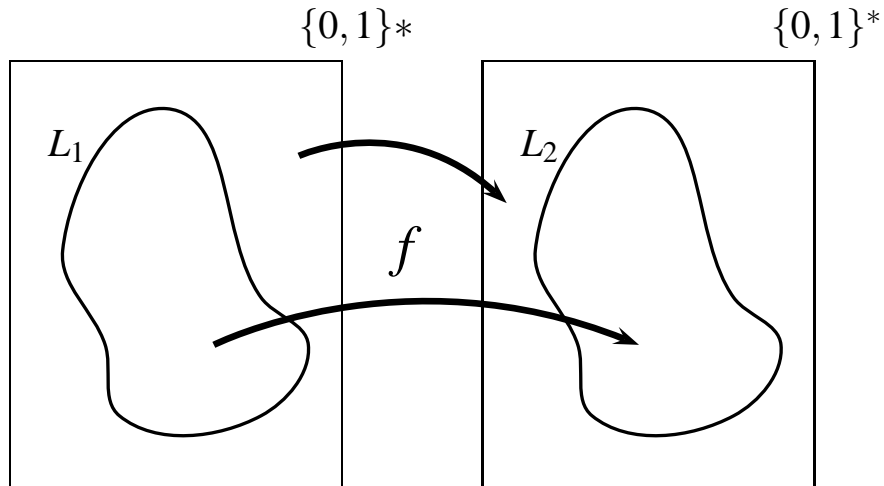
$$\text{BIP-MATCH} \leq \text{MAX-FLOW}$$



f er max strømning med heltallig værdi. \Leftrightarrow
“midterste” kanter udgør en maximal parring.

Reduktion

$$L_1 \leq L_2 \Leftrightarrow \begin{aligned} &\exists f : \{0, 1\}^* \rightarrow \{0, 1\}^*, \\ &x \in L_1 \text{ hvis og kun hvis } f(x) \in L_2 \end{aligned}$$



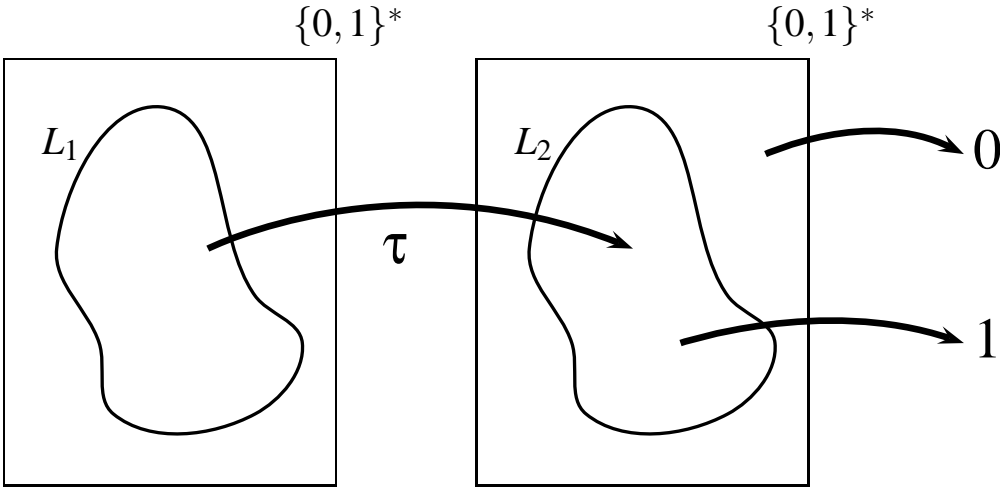
Polynomiell reduktion

$$L_1 \leq_{pol} L_2$$

f er polynomieltids afbildning

Reduktion

Hvis $(L_1 \leq_{pol} L_2$ og $L_2 \in \mathcal{P})$ så $L_1 \in \mathcal{P}$



\mathcal{NP} -fuldstændige problemer

Et problem Q kaldes \mathcal{NP} -fuldstændigt \Leftrightarrow

1 $Q \in \mathcal{NP}$

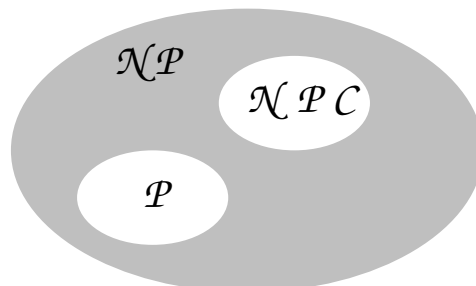
2 $\forall R \in \mathcal{NP} : R \leq_{pol} Q$

Et problem som opfylder (2) kaldes \mathcal{NP} -hårdt.

Sætning

Hvis der findes et \mathcal{NP} -fuldstændigt problem som er løseligt i polynomiell tid, så er $\mathcal{NP} = \mathcal{P}$.

Hvis et problem i \mathcal{NP} ikke kan løses i polynomiell tid så kan ingen \mathcal{NP} -fuldstændige problemer løses i polynomiell tid.



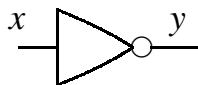
Eksistens af \mathcal{NP} -fuldstændigt problem

- Alle problemer i \mathcal{NP} skal kunne reduceres til Q
- Ethvert \mathcal{NP} -problem har polynomiell verifikationsalgoritme
- En verifikationsalgoritme kan afvikles på en computer
- Naivt gæt: “Computeren” er \mathcal{NP} -fuldstændig

Computeren som et kredsløb

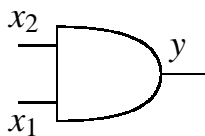
En computer består af “gates”

NOT



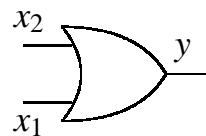
x	y
0	1
1	0

AND



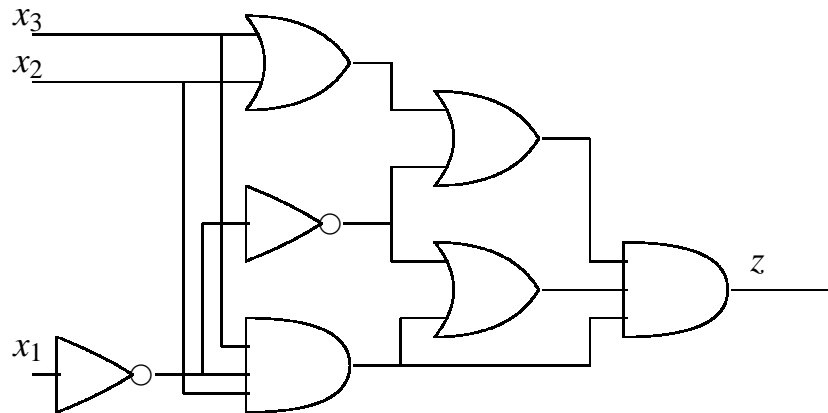
x_1	x_2	y
0	0	0
0	1	0
1	0	0
1	1	1

OR

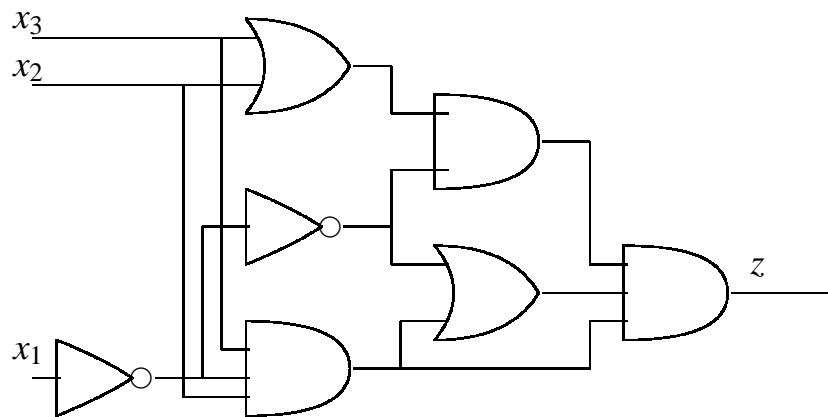


x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	1

Circuit satisfiability



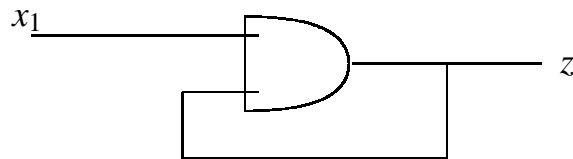
CIRCUIT-SAT: Givet et kredsløb, er det muligt at tildele værdier $\{0, 1\}$ til indgangene x_1, \dots, x_n så udgangen z bliver 1



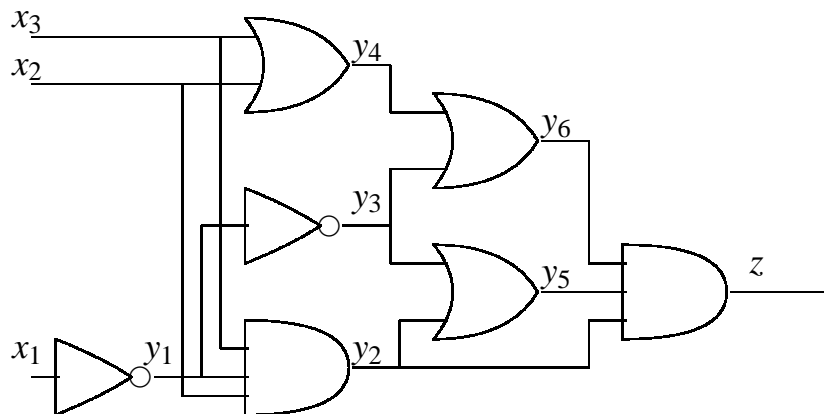
Ovenstående instans har ingen tilfredsstillende tildeling.

Circuit satisfiability er i \mathcal{NP}

Certifikat $Y = \{x_1, \dots, x_n\}$ virker ikke

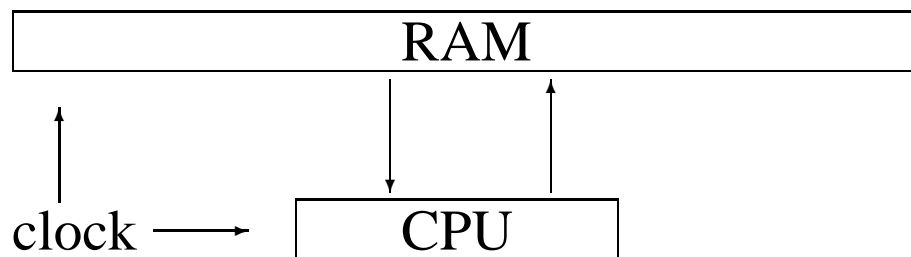


Certifikat $Y = \{x_1, \dots, x_n, y_1, \dots, y_m\}$ virker



Verifikations algoritme: check alle gates, input og output.
Kører i polynomiel tid.

Computeren som et kredsløb



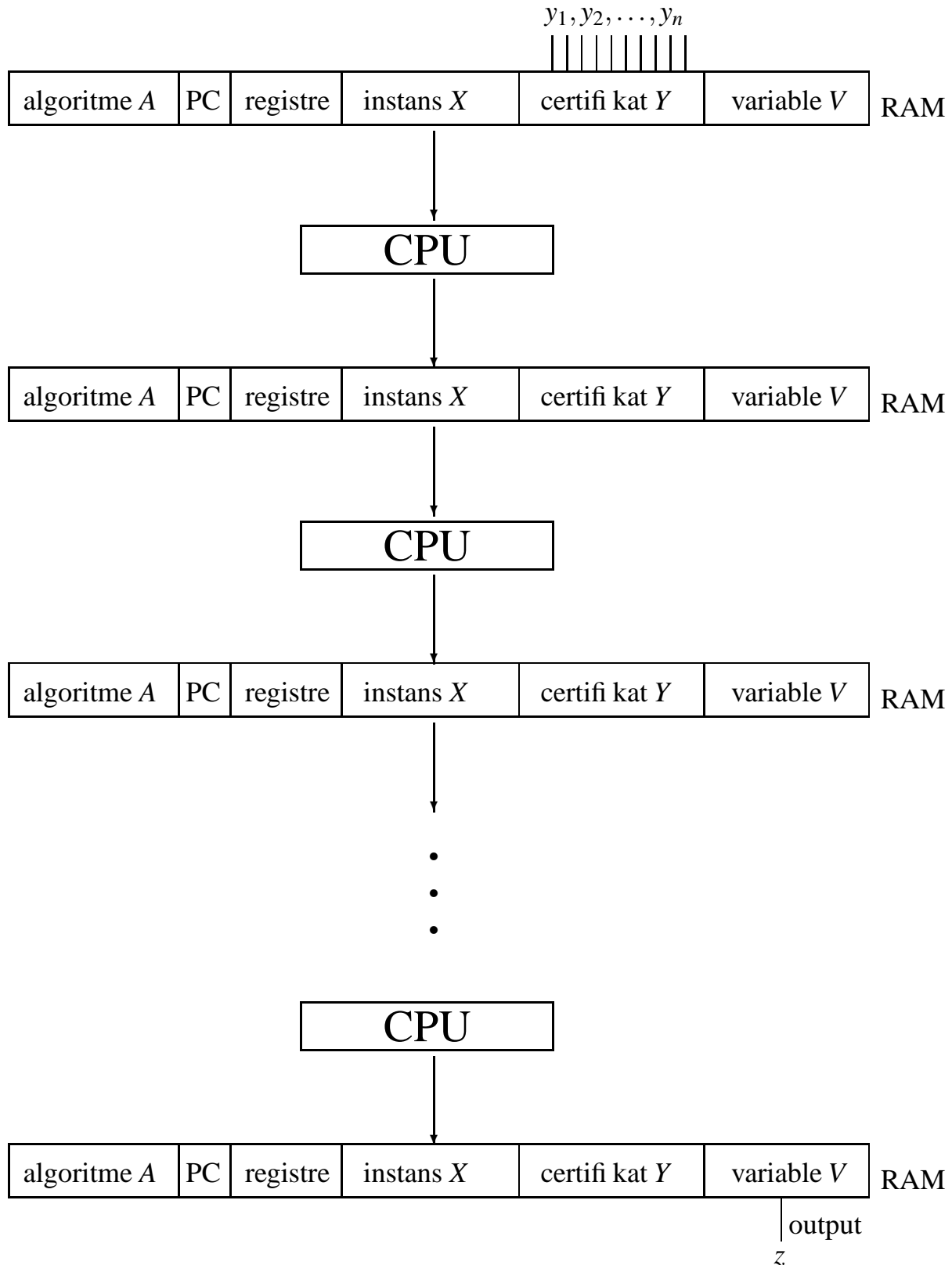
Afvikling af verifikationsalgoritme A på computer

En polynomiel algoritme A er ikke længere end $|X|^k$.

En polynomiel algoritme bruger ikke mere plads (variable) end $|X|^k$

- **CPU, central processing unit**
udfører aritmetisk-logiske operationer i hver klokcyklus
konstant antal gates
- **RAM, random access memory**
kan lagre algoritme A , instans X , certifikat Y , variable V , output Z
 $|A|$ polynomiel i X
 $|Y|$ polynomiel i X
 $|V|$ polynomiel i X
- **OUTPUT**
Værdi $\{0, 1\}$ skrives til RAM
konstant antal gates

Circuit satisfiability er \mathcal{NP} -fuldstændig



Resume

- Abstrakt problem: problemtype
- Konkret problem, problem instans
- Kodning: effektiv (binær eller pol. relateret til binær)
- Abstrakte problemer: sprog over $\{0, 1\}^*$
- $\mathcal{P} = \{L : L \text{ accepteres af en algoritme i polynomiel tid}\}$
- $x \in L$ verificeres af en algoritme hvis der findes certifikat y så algoritmen accepterer (x, y)
- $\mathcal{NP} = \{L : L \text{ verificeres af en polynomiel tids algoritme } A\}$
- $\mathcal{P} \subseteq \mathcal{NP}$
- Q er \mathcal{NP} -fuldstændig hvis $Q \in \mathcal{NP}$ og $\forall R \in \mathcal{NP} : R \leq_{pol} Q$
- CIRCUIT-SAT er \mathcal{NP} -fuldstændig.

Store spørgsmål

- $\mathcal{P} = \mathcal{NP} ?$