Complexiteitstheorie

Bullet points en dat soort dingen

‘‘sharing is caring’’

Hylke Visser
hylkev@ch.tudelft.nl

De volgorde van de onderwerpen in dit document komt overeen met de volgorde waarin deze onderwerpen tijdens de colleges in 2014 zijn behandeld. Aanpassingen of toevoegingen aan dit document kun je naar me mailen. TeX source komt later nog wel beschikbaar.

Laatste update: 13 apr ’14

Basiskennis

Tijdcomplexiteit: $f : N \to N$ waarbij $f(n)$ het grootste aantal stappen is dat M nodig heeft voor een input van grootte n.

Complexiteit van een probleem: de *worst-case complexiteit* van het beste algoritme, uitgevoerd op het gekozen machinemodel.

Relaties tussen Tm’s: Voor elke *k-tape* ($k \geq 2$) DTm die een probleem oplost in $t(n)$-tijd, bestaat er een *single-tape* DTm die hetzelfde probleem oplost in $\mathrm{O}(t^2(n))$

Sterke Church-Turing these: Elk fysisch realiseerbaar algoritmisch proces kan worden gesimuleerd op een 1-tape DTm met *polynomiale overhead* (met mogelijke uitzondering de quantum computer).

Complexiteitsklasse (N)TIME(t(n)): Verzameling van alle talen beslist door $\mathrm{O}(t(n))$-(N)DTm’s.

Tijd-hierarchie

TIME($f(n)$) $\subseteq$ **NTIME($f(n)$)**

TIME(1)

TIME(log(n))

TIME($\sqrt{\mathbf{n}}$)

TIME(n) Lineair

TIME($\mathbf{n \cdot \log(n)}$)

TIME($\mathbf{n^c}$) Polynomiaal

TIME($\mathbf{n^{\log(n)}}$) Quasi-polynomiaal

TIME($\mathbf{2^{\sqrt{n}}}$) Subexponentieel

TIME($\mathbf{2^{O(n)}}$) Exponentieel

Doenlijk vs Ondoenlijk

Een probleem is doenlijk als het in polynomiale tijd op te lossen is. Een probleem is ondoenlijk als er geen polynomiaal algoritme voor het probleem bestaat. Technologische vooruitgang heeft een positieve invloed op de snelheid waarmee doenlijke problemen kunnen worden opgelost. Bij ondoenlijke problemen is technologische vooruitgang niet van belang.

Complexiteitsklassen

Hiërarchie: P $\subseteq$ NP $\subseteq$ EXP.

P

- Talen die door een DTm in polynomiale tijd beslist worden.
- $\bigcup_{k \geq 0} \text{TIME}(n^k)$

NP

- Talen die door een NDTm in polynomiale tijd beslist worden.
- $\bigcup_{k \geq 0} \text{NTIME}(n^k)$
- Talen die door een DTm in polynomiale tijd geverifieerd worden.
  - Met behulp van een certificaat c
- Het is onbekend of P = NP of P $\neq$ NP

Polynomiale Reducties

- Als een probleem A reduceerbaar is tot een probleem B (A $\leq$ B), kan een efficiënte oplossing voor B gebruikt worden voor het efficient oplossen van A.
- Een reductiefunctie $f : \Sigma^* \to \Sigma'^*$ transformeert een probleem A naar probleem B.
- $w \in \text{A} \Leftrightarrow f(w) \in \text{B}$
  - yes-instanties van A worden afgebeeld op yes-instanties van B
    - $\forall x \in \Sigma^* : x \in A \to f(x) \in B$

  - no-instanties van A worden afgebeeld op no-instanties van B
    * $\forall x \in \Sigma^* : x \notin A \rightarrow f(x) \notin B$
    * $\forall x \in \Sigma^* : f(x) \in B \rightarrow x \in A$

- f is polynomiaal berekenbaar.
- Als A_B een polynomiaal algoritme is voor B, dan is $A_B \circ f$ een polynomiaal algoritme voor A.
- Dan geldt dat A *niet essentieel moeilijker is dan B.*

P is naar beneden gesloten onder $\leq$
Als f een polynomiale-tijd reductie is van A naar B ($A \leq B$), dan geldt: als $B \in P$ dan $A \in P$

Eigenschappen van polynomiale reducties
Reflexiviteit, Transitiviteit

Correctheid van Reducties bewijzen

- Bewijs de betrouwbaarheid van f
  - yes-instanties van A $\rightarrow$ yes-instanties van B
  - no-instanties van A $\rightarrow$ no-instanties van B
- Bewijs de polynomialiteit van f

NP-hard en NP-compleet (NPC)

Een probleem B is NP-hard (onder $\leq$) als voor iedere X in NP geldt dat $X \leq B$.
Een probleem A is NP-compleet (onder $\leq$) als geldt dat:

- $A \in NP$
- A is NP-hard (onder $\leq$)

P bevat de eenvoudigste problemen in NP

NPC bevat de moeilijkste problemen in NP

Als P $\neq$ NP bevat NP oneindig veel verschillende moeilijkheidsniveaus

Als A $\in$ NPC, B $\in$ NP en A $\leq$ B dan $B \in NPC$.

Als A $\in$ NPC $\cap$ P dan P = NP

Als A $\in$ NPC en A $\notin$ P dan P $\neq$ NP

A ∈ NPC bewijzen

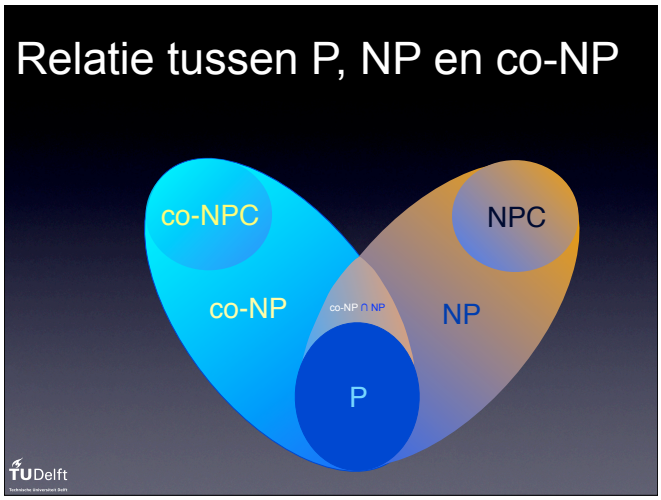
- Bewijs dat A ∈ NP
  - Toon aan dat het probleem polynomiaal verifieerbaar is
- Bewijs dat B ≤ A voor een bekend NPC probleem B
  - Kies een geschikt bekend NPC probleem B
  - Construeer een polynomiale reductie B ≤ A
  - Bewijs de correctheid van deze reductie

co-NP

- Het complement van een probleem A is het probleem $A^c = \Sigma^* - A$
- co-NP bestaat uit alle problemen B die het complement zijn van een probleem A in NP
- co-NP = { B : ∃ A ∈ NP [B = A^c] }
- A in co-NP als een NDTm no-instanties in polynomiale tijd oplost
- A in co-NP als no-instanties polynomiaal verifieerbaar zijn

Als A ≤ B dan $A^c \leq B^c$

Als A ∈ NPC dan A^c ∈ co-NPC



Bewijs met restrictie

Een beslissingsprobleem A is een restrictie van een probleem B als:

- $Yes_A \subseteq Yes_B$

- $\text{No}_\text{A} \subseteq \text{No}_\text{B}$

Als A een restrictie is van B dan geldt A $\leq$ B

Om aan te tonen dat B een NP-compleet probleem is:

- Bewijs dat B $\in$ NP
- Bewijs dat een bekend NP-hard probleem A een restrictie is van B

Ruimtecomplexiteit

SPACE($f(n)$) { L : L wordt geaccepteerd door een DTm die O($f(n)$) ruimte gebruikt }

NSPACE($f(n)$) { L : L wordt geaccepteerd door een NDTm die O($f(n)$) ruimte gebruikt }

SPACE($f(n)$) $\subseteq$ NSPACE($f(n)$)

PSPACE $= \bigcup_{k\geq 0}$ SPACE(n^k)

L $=$ SPACE(log n)
PolyL $= \bigcup_{k\geq 0}$ SPACE($\log^k$ n)

NPSPACE $= \bigcup_{k\geq 0}$ NSPACE(n^k)

NL $=$ NSPACE(log n)
PolyNL $= \bigcup_{k\geq 0}$ NSPACE($\log^k$ n)

P $\subseteq$ NP $\subseteq$ PSPACE
PSPACE = NPSPACE

PSPACE-hard en PSPACE-compleet

Een probleem is PSPACE-compleet als:

- A $\in$ PSPACE
- X $\leq$ A voor elk PSPACE probleem X

Getalproblemen en Pseudo-polynomiale algoritmen

I_A: instantie van probleem A

$|I_A|$: grootte van I_A

max(I_A): grootte van het grootste getal in I_A (unaire codering)

A is een getalprobleem als er geen enkele polynoom $p(\ )$ bestaat zodat voor alle instanties I_A geldt dat $\max(I_A) \leq p(|I_A|)$

Een algoritme X_A voor een getalprobleem A is een pseudo-polynomiaal algoritme als de tijdcomplexiteit van X_A gelijk is aan $O(q(|I_A|, \max(I_A)))$ voor een polynoom q(,) en voor alle instanties I_A van A.

Sterk NP-complete problemen

A_p: het subprobleem van A bestaande uit instanties I_A met kleine getallen:
$\max(I_A) \leq \mathrm{p}(|I_A|)$
A is sterk NP-compleet (SNPC) als:

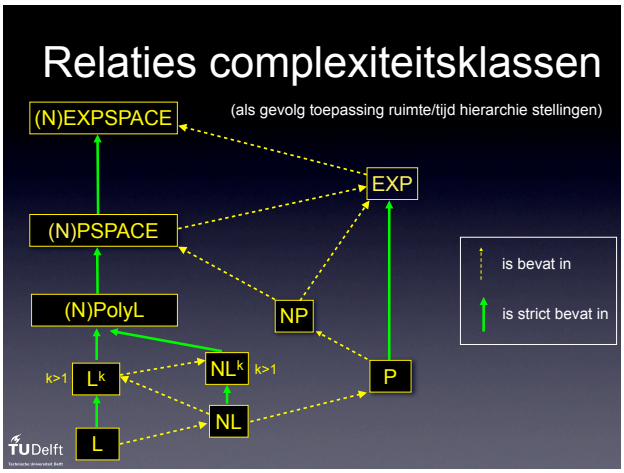
- A $\in$ NP
- er bestaat een polynoom p met $A_p \in$ NPC

Oftewel, de grootte van de getallen is niet van invloed op de moeilijkheid van het probleem

Tijd en ruimte construeerbaar / separatie

- Hoveel moet de hoeveelheid tijd of geheugen toenemen om een verschil te maken?
- Een functie $f : N \to N$ is ruimte-construeerbaar als er een DTM bestaat die voor input 1^n als output $[f(n)]_2$ berekent in O(f(n))-ruimte.
- Een functie $f : N \to N$ is tijd-construeerbaar als er een DTM bestaat die voor input 1^n als output $[f(n)]_2$ berekent in O(f(n))-tijd.

Gevolg:



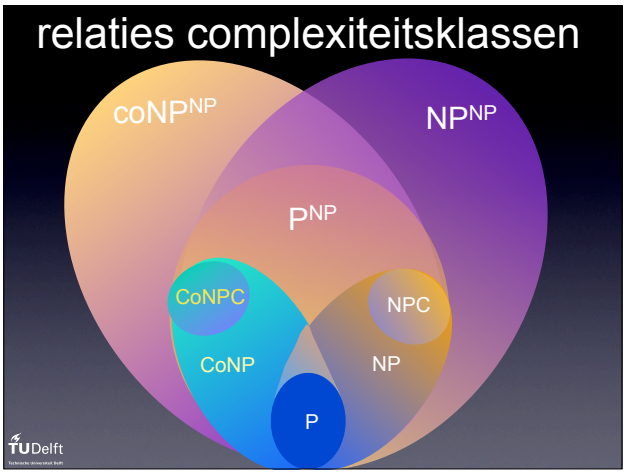
Orakels / Relativized Complexity Classes

$\mathbf{C^Y}$ bevat talen die een C algoritme hebben als een Y orakel wordt gebruikt

$\mathbf{P^{NP}}$ bevat talen die polynomiaal beslisbaar zijn door gebruik van een NP orakel

$\mathbf{NP^{NP}}$ bevat talen die polynomiaal verifieerbaar zijn door gebruik van een NP orakel

$\mathbf{co - NP^{NP}}$ bevat talen waarvan het complement verifieerbaar is in polynomiale tijd door gebruik van een NP-orakel



Zero Knowledge Proof Systems

P*rover* wil **V***erfifier* ervan overtuigen dat **P** in het bezit is van informatie X, zonder de inhoud van X aan **V** prijs te geven.

Completeness: if the statement is true, the honest verifier (that is, one following the protocol properly) will be convinced of this fact by an honest prover.

Soundness: if the statement is false, no cheating prover can convince the honest verifier that it is true, except with some small probability.

Zero-knowledge: if the statement is true, no cheating verifier learns anything other than this fact. This is formalized by showing that every cheating verifier has some simulator that, given only the statement to be proven (and no access to the prover), can produce a transcript that "looks like" an interaction between the honest prover and the cheating verifier.

Optimaliseringsproblemen

Decisie (dec): Gegeven een instantie x, een integer d en een kostenfunctie $c(\ )$, is er een oplossing y van x met kosten $c(y) > d$ (of $c(y) < d$)?

Optimalisatie (opt): Gegeven een instantie x en een kostenfunctie $c(\ )$, bepaal een oplossing y van x met minimale/maximale kosten $c(y)$.

Optimal value (val): Gegeven een instantie x en een kostenfunctie $c(\ )$, bepaal de minimale/maximale kosten $c(y)$ haalbaar voor een oplossing y van x.

Kenmerken:

- Voor elke instantie $x \in A$ is er een verzameling $F(x)$ van mogelijke oplossingen (feasible solutions).
- De kostenfunctie $c(\ )$ bepaalt voor iedere $y \in F(x)$ de kosten $c(y)$ van y.
- Er is een optimale oplossing $\text{opt}(x) \in F(x)$ met minimale (of maximale) kosten.

– Vaak wordt opt(x) ook gebruikt om de kosten van de optimale oplossing aan te geven

Een optimaliseringsprobleem A is een **NPO-probleem** (Niet-Deterministisch Polynomiaal Optimaliseringsprobleem) als:

1. voor elke $y \in F(x)$ geldt dat $|y|$ polynomiaal begrensd is in $|x|$
2. voor elke $x \in A$ is het probleem $y \in F(x)$? een P-probleem (als 1.)
3. voor elke $y \in F(x)$ is $c(y)$ polynomiaal berekenbaar

Turing Reducties

$A \leq_{\mathrm{T}} B$: Er is een algoritme voor A dat orakelaanroepen voor B gebruikt en polynomiaal zou zijn als orakelaanroepen O(1)-tijd kosten.

- $A \leq_{\mathrm{T}} A$
- Als $A \leq_{\mathrm{T}} B$ en $B \leq_{\mathrm{T}} C$, dan $A \leq_{\mathrm{T}} C$

en

- A-dec $\leq_{\mathrm{T}}$ A-opt
- A-val $\leq_{\mathrm{T}}$ A-dec
- A-opt $\leq_{\mathrm{T}}$ A-val als A zelf-reduceerbaar is (A-dec $\in$ NPC)

Een (optimaliserings, value, decisie) probleem A-opt is

- **NP-hard** als $\forall B \in \mathrm{NP}, B \leq_{\mathrm{T}} A$
- **NP-easy** als $\exists B \in \mathrm{NP}, A \leq_{\mathrm{T}} B$
- **NP-equivalent** als $A \in$ NP-hard $\cap$ NP-easy

Als A-dec $\in$ (co)NPC, dan:

- A-opt $\in$ NP-equivalent
- A-val $\in$ NP-equivalent

Approximatie algoritmen

Een approximatie algoritme geeft altijd een feasible oplossing, maar niet noodzakelijk een optimale.

Performance ratio van M voor een instantie $x \in A$:

$$R_{\mathrm{M}}(x) = \frac{\max\{\mathrm{opt}(x), c(M(x))\}}{\min\{\mathrm{opt}(x), c(M(x))\}}$$

Dit is altijd een getal ≥ 1; hoe dichter bij 1 hoe beter.

M wordt een r-approximatie algoritme voor A genoemd als de performance ratio $R_{\mathrm{M}}(x)$ voor iedere instantie x kleiner of gelijk is aan r

$$\forall x \; R_{\mathrm{M}}(x) \leq r$$

A $\in$ APX: Een NPO-probleem A is goed-benaderbaar als er een polynomiaal c-approximatie algoritme voor A bestaat met $c < \infty$.

Boolese Circuits

Een boolese circuit C_m bestaat uit $m + n$ knopen (poorten/gates) verbonden door lijnen (wires). De eerste m poorten zijn input poorten, de overige n poorten zijn circuit poorten. Iedere poort is een boolese functie $\wedge, \vee, \neg$.

size(C): aantal poorten van C

- bepaalt de sequentiele tijd van berekeningen

depth(C): langste pad van input naar output

- bepaalt de parallelle tijd van berekeningen

Een **circuit familie** C is een oneindige lijst $(C_1, ..., C_m, ...)$ van circuits C_m; iedere C_m heeft m input variabelen.

Als t een superlineaire functie is $(t(n) \geq n)$, dan geldt:
als $A \in TIME(t(n))$ dan heeft A een circuit size complexiteit $O(t^2(n))$

Een probleem is efficient parallelliseerbaar als het kan worden opgelost (**NC** / Nick's Classes):

- in extreem korte berekeningstijd
  - poly-logaritmische tijd
  - depth = $O(\log^{O(1)} n)$
- met een redelijk aantal processoren
  - polynomiaal begrensd aantal processoren
  - size = $O(n^{O(1)})$

NC^i is de klasse van problemen die kunnen worden opgelost met een familie van circuits waarvoor geldt dat:

- size(C) = $n^{O(1)}$
- depth(C) = O($\log^i n$)

NC is de klasse { A | $\exists i$. A $\in$ NC^i }

Een NC-algoritme is een algoritme dat in O($\log^{O}(1)$)-tijd en met O($\log^{O}(1)$)-processoren kan worden uitgevoerd.

NC $\subseteq$ P

Een probleem A is P-compleet als

- $A \in P$
- voor iedere $B \in P$ geldt $B \leq_{\text{NC}} A$

Problemen in P - NC worden inherent sequentieel genoemd.
Voor deze problemen zijn geen efficiënte parallelle algoritmen gevonden.

Probablistische Algoritmen

Een probabilistisch algoritme is een algoritme dat naast de input x een random source gebruikt om tijdens de executie random keuzes te maken.

Voordelen:

- Efficientie
- Eenvoudig ontwerp
- Inspiratiebron voor deterministisch algoritme

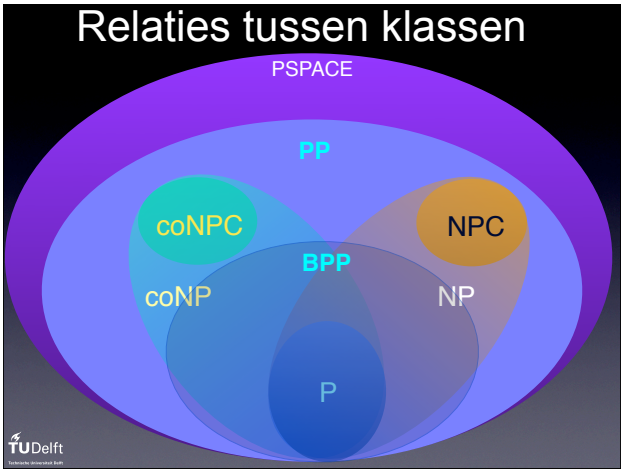
Een Probablistische Turing Machine (PTm) is een speciale NDTm waarbij in iedere configuratie een probablistische keuze gemaakt wordt uit de twee mogelijk toepasbare instructies.

Een taal L wordt geaccepteerd door een PTm M met foutmarge ε als

- $x \in L$ impliceert Pr[M accepteert x] $\geq 1 - \varepsilon$
- $x \notin L$ impliceert Pr[M verwerpt x] $\geq 1 - \varepsilon$

Bounded Probablistic Polynomial Time (BPP):
$L \in$ BPP als er een polynomiale PTm M bestaat die L accepteert met een foutmarge $\varepsilon = 1/3$
Probablistic Polynomial Time (PP):
$L \in$ PP als er een polynomiale PTm M bestaat waarvoor $x \in L \Leftrightarrow P$[M accepteert x] > 0.5
Relaties:

- NP $\subseteq$ PP en BPP $\subseteq$ PP
- BPP $\subseteq$ $\text{NP}^{\text{NP}} \cap$ co-NP^{NP}



Mind changing machines

n-trial machines zijn TMs die n verschillende outputs (achter elkaar) mogen geven en waarbij de laatste output als resultaat van de berekening telt.

P $\neq$ NP

Church-Turing These (CTT): ieder fysisch realiseerbaar berekeningsproces kan worden gesimuleerd met een deterministische Tm.

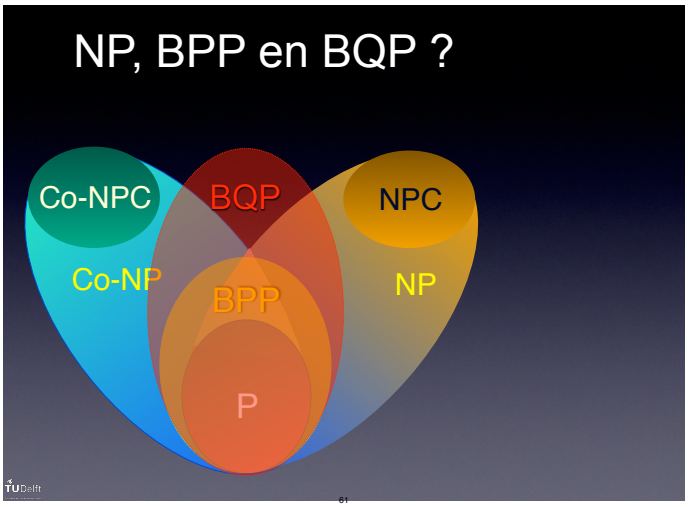
Sterke Church-Turing These (SCTT): ieder fysisch realiseerbaar efficient berekeningsproces kan worden gesimuleerd met een deterministische Tm en met polynomiale overhead.

Sterke P $\neq$ NP These (SPNPT): een NP-hard probleem kan niet worden opgelost door een fysisch proces in polynomiale tijd.

Quantum Computers

- Quantum computers opereren op quantum bits (qubits). Een qubit kent twee basistoestanden:
  - spin-down ($|0\rangle$)
  - spin-up ($|1\rangle$)
- Superstate $|\psi\rangle$ is een lineaire combinatie van de basistoestanden:
  - $|\psi\rangle = a|0\rangle + b|1\rangle$ met waarschijnlijkheidsamplituden a, b $\in \mathbb{C}$
  - Meting van een qubit state $|\psi\rangle$ resulteert altijd in een basistoestand $|0\rangle$ of $|1\rangle$.
  - Deze toestanden treden op met waarschijnlijkheid $|a|^2$, respectievelijk $|b|^2$ (hieruit volgt dat $|a|^2 + |b|^2 = 1$)
- Een quantum systeem van n qubits kent 2^n basistoestanden. Een superpositie toestand $|\psi\rangle$ is te schrijven als $|\psi\rangle = a_1|000...0\rangle + a_2|000...1\rangle + ... + a_{2^n}|111...1\rangle$ met $\Sigma_{i=1..2^n}|a_i|^2 = 1$.
- Een quantumberekening is een serie lineaire transformaties van superposities.

BQP is de klasse van alle problemen waarvoor een quantum computer het juiste antwoord geeft in polynomiale tijd met kans ≥ 0.667
Relaties: P $\subseteq$ BPP $\subseteq$ BQP $\subseteq$ PP $\subseteq$ PSPACE



- Soms wordt met quantum computing exponentiële versnelling bereikt. (Shor's Algoritme)
- Voor aantal problemen kan met quantum computing "slechts" een kwadratische versnelling worden bereikt. (Grover's Algoritme)
- Er zijn voldoende redenen om de Sterke Church Turing These als verworpen te beschouwen op grond van resultaten behaald met quantum computing.
- Er zijn geen redenen om de Sterke P $\neq$ NP These te verwerpen: ook quantum computing is er (nog) niet in geslaagd NPC problemen in polynomiale tijd op te lossen.