Complexiteit — college 11

NP en NP-volledigheid

Mark van den Bergh

2 mei 2025

Universiteit Leiden

Vorige keer

- Eulerkringen: d.e.s.d.a. graad van elke knoop even; constructie
- Hamiltonkringen: controleren makkelijk, oplossen moeilijk
- handelbare/onhandelbare problemen: wel/geen *deterministisch* polynomiaal algoritme om oplossing te *vinden*
- $\mathcal{P}$ handelbaar; $\mathcal{EXP} \setminus \mathcal{P}$ onhandelbaar
- $\mathcal{NP}$: *niet-deterministisch* polynomiaal algoritme, oplossing in polynomiale tijd te *controleren*
- $\mathcal{NPC}$: moeilijkste problemen in $\mathcal{NP}$; handelbaarheid onbekend

De klasse $\mathcal{NP}$

$\mathcal{NP}$ is de klasse van beslissingsproblemen waarvoor een niet-deterministisch polynomiaal algoritme bestaat:

- niet-deterministisch:
  je mag een oplossing gokken → certificaat
- polynomiaal:
  deze kan daarna in polynomiale tijd worden geverifieerd (gecontroleerd)
- ja-instantie:
  voor een ja-instantie bestaat er een oplossing, en dus een certificaat waarmee je aantoont dat het inderdaad een ja-instantie is

Niet-deterministisch algoritme

We kunnen een niet-deterministisch algoritme omschrijven als bestaande uit drie fases: gokken, verifiëren en uitvoer. Het dictaat gebruikt een dergelijke omschrijving in de definitie van $\mathcal{NP}$.

We bekijken de drie fases in detail:

1. Niet-deterministische gokfase

   Er wordt een willekeurige string s in het geheugen geschreven. Elke keer dat het algoritme executeert kan dit een andere string zijn.

   Deze string is vaak een soort gok van de oplossing van het probleem: het beoogde certificaat; s kan echter ook een onzinstring blijken te zijn.

Niet-deterministisch algoritme

2. Deterministische verificatiefase

   Zowel de invoer x van het probleem als de string s mogen hier worden gebruikt. Er wordt True of False geretourneerd, of het programma stopt nooit (het kan bijvoorbeeld belanden in een oneindige loop).

   Hier wordt gecontroleerd of s een oplossing is van het probleem bij de gegeven invoer x, m.a.w. er wordt gecontroleerd of s een ja-antwoord rechtvaardigt.

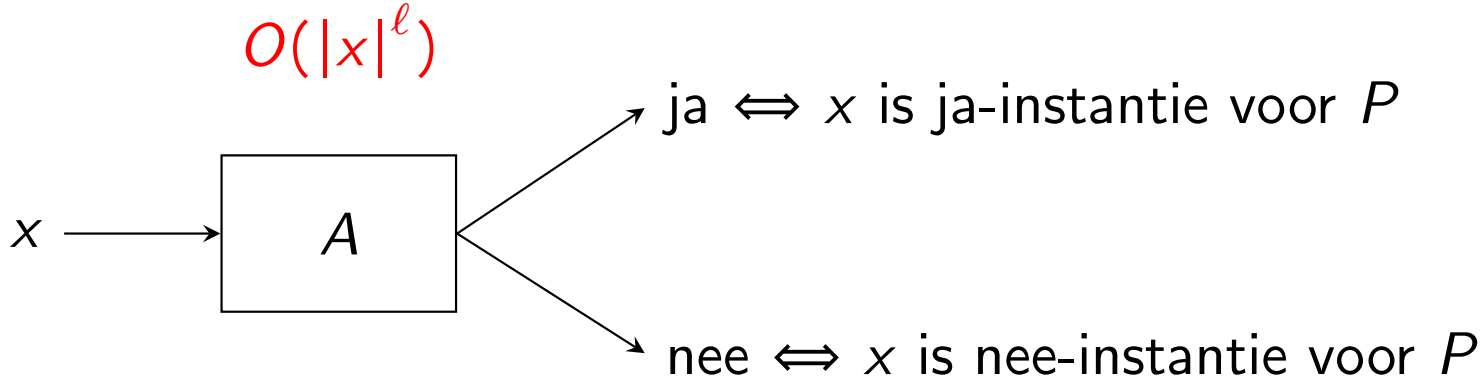
Niet-deterministisch algoritme

3. Uitvoerfase
   Als fase 2 (verificatie) True retourneert, geeft het algoritme antwoord "ja". Anders is er geen uitvoer.

Het aantal stappen dat een niet-deterministisch algoritme doet is het aantal stappen dat in de gokfase nodig is om s te schrijven (dus het aantal karakters waaruit s bestaat) plus het aantal stappen dat wordt gedaan in de verificatiefase (plus één stap voor de uitvoerfase).

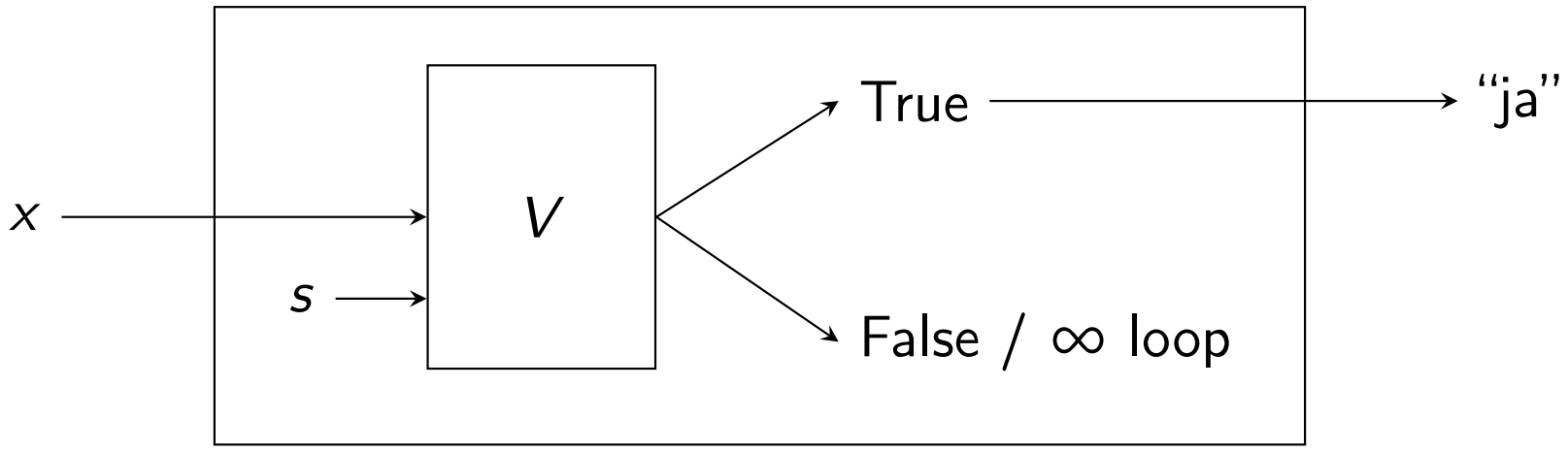
Van vorige week: $P \in \mathcal{P}$

Gegeven een willekeurig beslissingsprobleem $P \in \mathcal{P}$. Dan is er een polynomiaal deterministisch algoritme A dat P oplost voor elke invoer x.



Het algoritme is polynomiaal in de lengte van de invoer, dus worst case is $O(|x|^{\ell})$ ($\ell \geq 0$). Een algoritme is deterministisch als het elke keer dat het wordt uitgevoerd op dezelfde invoer (instantie) hetzelfde doet, en dus dezelfde uitvoer oplevert.

$\mathcal{NP}$ schematisch: een executie



fase 1: s wordt gegenereerd (niet-deterministisch)

fase 2: verificatie V (deterministisch)

fase 3: uitvoerstap

Vragen

Bij een niet-deterministisch algoritme zijn verschillende executies mogelijk voor dezelfde invoer x, afhankelijk van de gegokte s. Dus:

(a) Wat is *het* antwoord (ja/nee) van het algoritme voor invoer x?

(b) Wanneer noemen ze zo'n algoritme polynomiaal?

Antwoord

Definitie

Het antwoord van een niet-deterministisch algoritme A voor invoer x is "ja" $\iff$ er is een executie* van A die "ja" geeft als uitvoer $\iff$ er is een string s waarvoor fase 2 True oplevert. Het antwoord van A is "nee" als er voor geen enkele executie, dus voor geen enkele string s, een uitvoer is.

Er geldt dus: het antwoord van zo'n niet-deterministisch algoritme A voor invoer x is "ja"† $\iff$ er bestaat een string s (certificaat) waarmee je kunt aantonen (in fase 2) dat x een ja-instantie is.

---

*dus een string s

†oftewel: x is een ja-instantie

De klasse $\mathcal{NP}$

- een niet-deterministisch algoritme heet polynomiaal begrensd als voor elke invoer x waarvoor het antwoord "ja" is, er een executie is van het algoritme die "ja" oplevert in hooguit $O(|x|^{\ell})$ stappen (voor zekere $\ell \geq 0$).

  Dus mag in dat geval de string s niet te lang zijn (polynomiaal in $|x|$), en het verificatiealgoritme uit fase 2 moet polynomiaal zijn begrensd in $|x|$ (en $|s|$).

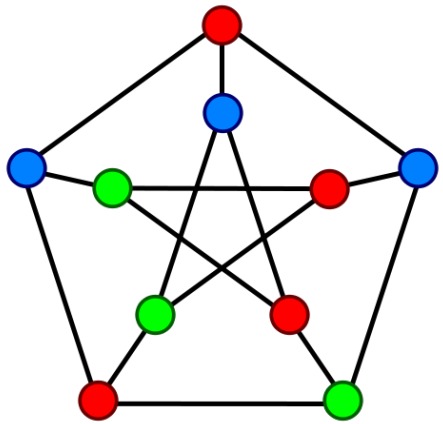
- $\mathcal{NP}$ is de klasse van beslissingsproblemen waarvoor er een polynomiaal begrensd niet-deterministisch algoritme bestaat.

- $\mathcal{NP}$ betekent: $\mathcal{N}$on-deterministisc $\mathcal{P}$olynomial time
  (en dus NIET: niet-polynomiaal!)

Voorbeelden $\mathcal{NPC}$-problemen

Van honderden problemen is bekend dat ze NP-volledig zijn. We bekijken hier zes zeer bekende $\mathcal{NPC}$-problemen:

- Hamiltonkring
- Handelsreizigersprobleem
- Graafkleuring
- Kliek
- SAT (CNF-satisfiability)
- Subset Sum



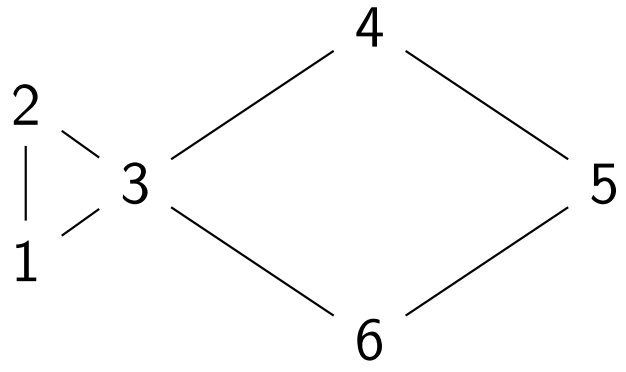
Hamiltonkring

Een Hamiltonkring in een ongerichte (of gerichte) graaf is een kring die **elke** knoop precies **één** keer bevat.

Beslissingsprobleem HC

Gegeven een graaf $G = (V, E)$. Heeft G een Hamiltonkring?

Voorbeeld



is een nee-instantie voor HC.

HC $\in \mathcal{NP}$

Een polynomiaal begrensd *niet-deterministisch algoritme* voor HC (invoer $\mathcal{G} = (V, E)$, $V = \{1, 2, \ldots, n\}$) (zie ook vorige week):

1. Fase 1 (gokfase)
   Er wordt een string s gegenereerd, hierna te interpreteren als een rij gehele getallen.
2. Fase 2 (verificatiefase)
   Er wordt gecontroleerd of s een Hamiltonkring voorstelt:
   (1) controleer dat er precies n integers staan: $O(|s|)$
   (2) controleer dat elke integer tussen 1 en n is: $O(|s|)$
   (3) controleer dat alle knopen uit s verschillen: $O(|s|^2)$
   (4) controleer dat tussen opeenvolgende knopen uit s een tak zit in de graaf (en tussen de eerste en de laatste): $O(|s| \times |E|)$.

HC $\in \mathcal{NP}$

Als de vier tests positief zijn wordt True geretourneerd. Zodra een test negatief uitvalt wordt False geretourneerd (of wordt een oneindige loop begonnen).

3. Fase 3 (uitvoerfase)
   Als fase 2 True oplevert wordt "ja" uitgevoerd, anders geen uitvoer.

Merk op. Tests (1) t/m (3) controleren dat s een rij van n verschillende knopen van G voorstelt (syntactische controle), test (4) controleert dat het een Hamiltonkring is.

Voor ja-instanties bestaat er een string s die een Hamiltonkring voorstelt, en dus een executie die "ja" oplevert, waarbij $|s| \in O(|G|)$. Zo'n ja-executie is dus polynomiaal in $|G|$. Voor nee-instanties bestaat zo'n string s niet.

Handelsreizigersprobleem

Handelsreizigersprobleem of Travelling Salesperson Problem (TSP)

Optimalisatieprobleem

Gegeven een volledige*, ongerichte graaf $G = (V, E)$ met gewichten op de takken. Geef een Hamiltonkring in G met minimaal totaalgewicht.

Beslissingsprobleem TSP

Gegeven een volledige, ongerichte graaf $G = (V, E)$ met gewichten op de takken, en een geheel getal $k \geq 0$. Bestaat er in G een Hamiltonkring met totaalgewicht $\leq k$?

---

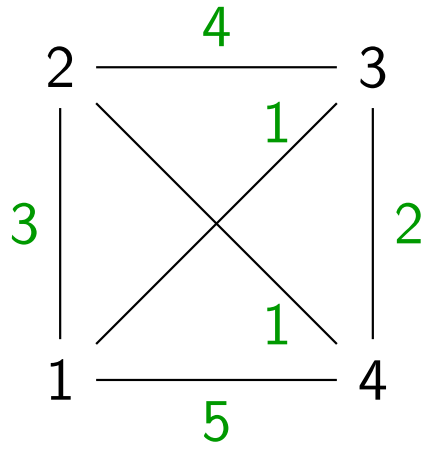
*tussen elk tweetal knopen van G zit een tak

TSP: voorbeeld

Voorbeeld

Een Hamiltonkring in onderstaande graaf is bijvoorbeeld 1, 2, 3, 4, 1. Deze heeft totaalgewicht 14.

De Hamiltonkring met minimaal gewicht is 2, 4, 3, 1, 2. Deze heeft totaalgewicht 7.



:)

https://xkcd.com/287/

Graafkleuring

Een kleuring van (de knopen van) een ongerichte graaf $G = (V, E)$ is een afbeelding $c\colon V \to S$, waarin S een eindige verzameling (van kleuren) is, met de eigenschap dat als $(v, w) \in E$ dan $c(v) \neq c(w)$. Met andere woorden: aangrenzende knopen hebben niet dezelfde kleur.

Optimalisatieprobleem

Gegeven een ongerichte graaf $G = (V, E)$. Vind een kleuring van G met zo weinig mogelijk kleuren.

Beslissingsprobleem Kleur

Gegeven een ongerichte graaf $G = (V, E)$ en een geheel getal $k > 0$. Bestaat er een kleuring van G die hooguit k kleuren gebruikt (oftewel: is G k-kleurbaar)?

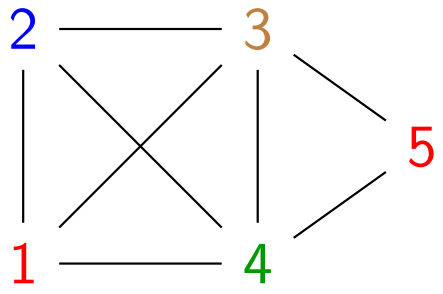
Kleur: vervolg

Opmerking

Het is equivalent om te vragen naar een kleuring met precies k kleuren: ze hoeven niet allemaal te worden gebruikt.

Voorbeeld

Onderstaande graaf kan worden gekleurd met 4 kleuren, maar niet met minder dan 4 (waarom niet)?
Kleur bijvoorbeeld 1 en 5 rood, 2 blauw, 3 bruin en 4 groen.



Kleur $\in \mathcal{NP}$

Laat $V = \{1, 2, \ldots, n\}$ en de mogelijke kleuren $1, 2, \ldots, k$. Een polynomiaal begrensd *niet-deterministisch algoritme* A voor Kleur, met als invoer $x = (G, k)$, is:

1. Fase 1 (gokfase)
   Er wordt een string s gegenereerd, hierna te interpreteren als een rij gehele getallen.

2. Fase 2 (verificatiefase)
   Er wordt gecontroleerd of s een goede kleuring voorstelt (s_i wordt geïnterpreteerd als de kleur van knoop i):

   (1) controleer of er precies $n = |V|$ integers staan (elke knoop een kleur): kan polynomiaal, $O(|s|)$

   (2) controleer of elke integer tussen 1 en k is (er worden k kleuren gebruikt): kan polynomiaal, $O(|s|)$

   (3) controleer of aangrenzende knopen verschillend zijn gekleurd.

Kleur $\in \mathcal{NP}$

Als de drie tests positief zijn wordt True geretourneerd. Zodra een test negatief uitvalt wordt False geretourneerd (of wordt een oneindige loop begonnen).

3. Fase 3 (uitvoerfase)
   Als fase 2 True oplevert wordt "ja" uitgevoerd, anders geen uitvoer.

Merk op. Tests (1) en (2) controleren of s een kleuring van de knopen voorstelt, test (3) controleert of de kleuring correct is.

Er geldt: het antwoord van A op invoer $x = (G, k)$ is "ja" $\Leftrightarrow$ er bestaat een goede string s waarop fase 2 True oplevert $\Leftrightarrow$ er bestaat een correcte kleuring van de knopen van $G \Leftrightarrow x = (G, k)$ is een ja-instantie voor Kleur.

Verder: voor een ja-executie, dus met s een correcte kleuring, is $|s| \in O(|V|) \subseteq O(|x|)$. Ergo: A is polynomiaal begrensd.

Kliek

Een kliek in een ongerichte graaf $G = (V, E)$ is een deelverzameling $W \subseteq V$ zodanig dat voor elk tweetal knopen $u, v \in W$ $(u \neq v)$ geldt dat $(u, v) \in E$. Met andere woorden: tussen elk tweetal knopen uit W zit een tak. De grootte van de kliek is het aantal knopen van die kliek.

Optimalisatieprobleem

Gegeven een ongerichte graaf $G = (V, E)$. Geef een kliek met zo veel mogelijk knopen (een maximale kliek).

Beslissingsprobleem Kliek

Gegeven een ongerichte graaf $G = (V, E)$ en een geheel getal k $(0 \leq k \leq |V|)$. Bestaat er in G een kliek ter grootte ten minste k?

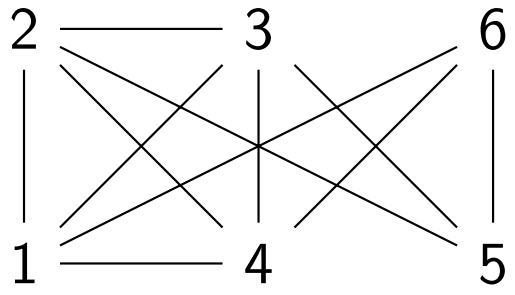
Kliek: vervolg

Opmerking

Het is equivalent om te vragen naar een kliek ter grootte gelijk aan k. Immers: elke deelverzameling van een kliek is zelf weer een kliek.

Voorbeeld

In onderstaande graaf is $\{1, 4, 6\}$ een kliek ter grootte 3. Een maximale kliek in deze graaf is er een ter grootte 4: $\{1, 2, 3, 4\}$.



SAT: terminologie

- een literaal (Engels: literal) is een Boolese variabele of de negatie daarvan (dus x of $\neg x$).
- een clausule (Engels: clause) is een disjunctie ($\vee$) van literalen. Voorbeeld: $x_1 \vee \neg x_3 \vee x_4 \vee \neg x_6$.
- een logische formule ϕ staat in CNF (conjunctieve normaalvorm) als hij bestaat uit een conjunctie ($\wedge$) van clausules.
  Voorbeeld: $\phi = (x_1 \vee x_2) \wedge (x_1 \vee \neg x_2 \vee \neg x_3) \wedge \neg x_1$.
- een waardering (waarheidstoekenning) van een verzameling logische variabelen $\{x_1, x_2, \ldots, x_n\}$ is een toekenning van de waarde True of False aan elk van de logische variabelen uit die verzameling.

SAT

Beslissingsprobleem SAT (CNF-satisfiability)

Gegeven een logische formule ϕ in CNF. Bestaat er een waardering van de in ϕ voorkomende logische variabelen zodat ϕ de waarde True krijgt (dus een waardering die ϕ waar maakt)?

Voorbeeld
De waardering w met $w(x_1)$ = False, $w(x_2)$ = True en $w(x_3)$ = False maakt de logische formule

$$(x_1 \vee x_2) \wedge (x_1 \vee \neg x_2 \vee \neg x_3) \wedge \neg x_1$$

waar. Deze formule is dus een ja-instantie voor SAT.

Subset Sum

Beslissingsprobleem SUM

Gegeven een getal $t \in \mathbb{N}$ en een eindige verzameling $S \subset \mathbb{N}$.
Bestaat er een deelverzameling $T \subseteq S$ met $\sum_{s \in T} s = t$?

Voorbeeld 1
Neem $S = \{1, 7, 28, 3, 2, 5, 9, 32, 41, 11, 8\}$ en $t = 30$, dan is dit een ja-instantie voor het probleem. Immers voldoet $T = \{7, 3, 9, 11\}$.

Voorbeeld 2
Neem $S = \{1, 4, 16, 64, 256, 1040, 1093, 1284, 1344\}$ en $t = 3754$, dan is dit een ja-instantie voor het probleem. Immers voldoet $T = \{1, 4, 16, 256, 1040, 1093, 1344\}$.

Samenvattend

- voor alle zes voorbeeldproblemen is eenvoudig een exponentieel algoritme op te schrijven.
- in alle gevallen kan in polynomiale tijd worden gecontroleerd of een (polynomiaal begrensde) kandidaatoplossing een echte oplossing (= "dat wat je zoekt") is $\rightsquigarrow$ in $\mathcal{NP}$
- voor al deze problemen geldt: het lijkt onmogelijk om een deterministisch polynomiaal algoritme op te schrijven

Certificaten

Probleem	Invoer	Certificaat
HC	G	Hamiltonkring
TSP	(G, k)	Hamiltonkring met totaalgewicht $\leq k$
Kleur	(G, k)	kleuring met $(\leq)$ k kleuren
Kliek	(G, k)	kliek met $(\geq)$ k knopen
SAT	ϕ	waarmakende waardering
Sum	(S, t)	deelverzameling met som t

Lineair?

1. Invoer: een array A met $n > 0$ gehele getallen
   Algoritme:

```
1 for i := 1 to n do
2 |   print A[i];
3 od
```

   Complexiteit: $\Theta(n)$

2. Invoer: een geheel getal $n > 0$
   Algoritme:

```
1 for i := 1 to n do
2 |   print '1';
3 od
```

   Complexiteit: $\Theta(n) = \Theta(2^{\lg n})$

En nu nog even dit...

Vraag: wat is eigenlijk de lengte van de invoer?

Voorbeeldprobleem

Gegeven een geheel getal $n > 1$. Heeft n echte delers, m.a.w., is $n = a \times b$ voor zekere $a, b > 1$? M.a.w.: is n (niet) priem?

Priemdelers

Algoritme:

```
// gewoon alle mogelijke delers proberen
1 gevonden := False;
2 m := 2;
3 while not gevonden and m < n do
4 |   if n mod m = 0 then
5 |   |   gevonden := True;
6 |   else
7 |   |   m := m + 1;
8 |   fi
9 od
// m is nu de kleinste deler > 1 van n
```

Lengte invoer

De worst case-complexiteit van dit algoritme is $\Theta(n)$ (indien we de berekening van n mod m tellen als één stap, anders $O(n^2)$).

De lengte van de invoer is het aantal karakters van de codering, in dit geval van het getal n. Als we de unaire codering gebruiken is het algoritme dus lineair in de lengte van de invoer. Nemen we de binaire codering, of in het algemeen de ℓ-aire codering met $\ell > 1$, dan is het algoritme exponentieel in de lengte van de invoer (voor elke $\ell > 1$ dus).

Het algoritme is dus niet polynomiaal, maar exponentieel!

Primes is in $\mathcal{P}$

In 2002 is bewezen door Manindra Agrawal en zijn PhD-studenten Neeraj Kayal en Nitin Saxena dat het probleem in $\mathcal{P}$ zit. De zogenaamde AKS-test is een priemgetaltest die polynomiaal is in het aantal cijfers (dus de lengte) van n.

In 2006 ontvingen zij hiervoor de Fulkerson Prize (discrete wiskunde) en de Gödel Prize (theoretische informatica).

Ander voorbeeld

Knapzakprobleem

Gegeven een knapzak met capaciteit S (een geheel getal > 0) en n objecten met gewichten $s_1, s_2, \ldots, s_n$ en met waarde $w_1, w_2, \ldots, w_n$. (Alle s_i en w_i zijn geheel en > 0.)
Gevraagd een deelverzameling van de objecten met totaalgewicht $\leq S$ en maximale totaalwaarde.

Het knapzakprobleem kan worden opgelost met een algoritme met complexiteit $O(nS)$ (dynamisch programmeren, zie Algoritmiek).

Dit is niet polynomiaal, maar exponentieel als functie van de lengte van de invoer!

Pseudopolynomialiteit

Het algoritme voor het knapzakprobleem op de vorige slide doet er alléén exponentieel lang over als de betrokken getallen exponentieel groot zijn (vergeleken met hun aantal).

Als we aannemen dat alle getallen (d.w.z. capaciteit, gewichten en waardes) polynomiaal zijn begrensd, is het algoritme dat ook. Deze **begrensde versie** van het knapzakprobleem is dus **polynomiaal oplosbaar**.

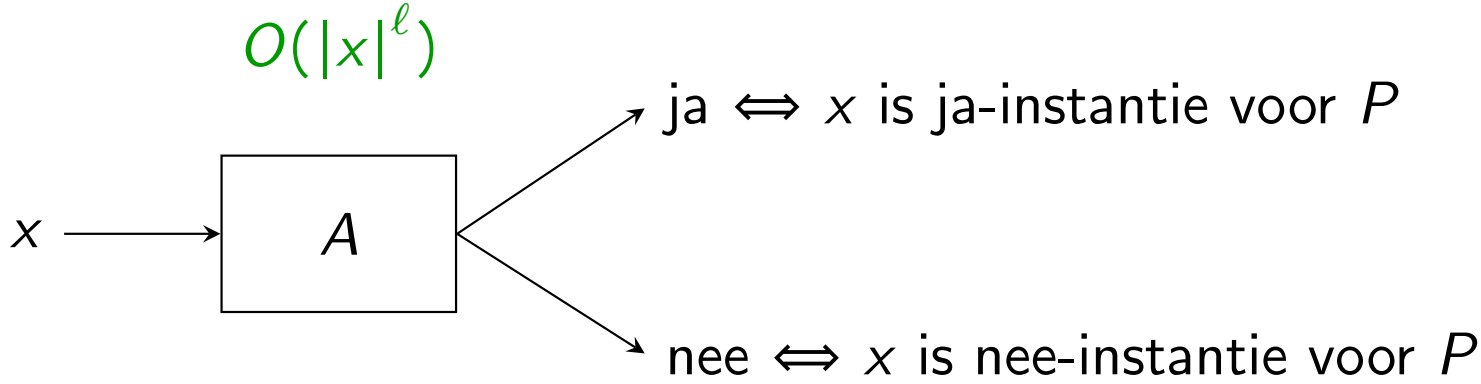
In de praktijk is het in veel toepassingen zeldzaam om zulke extreem grote getallen tegen te komen, en zal de rekentijd van zo'n algoritme dus bijna altijd polynomiaal zijn.

Dit soort algoritmen en problemen worden dan ook pseudopolynomiaal genoemd.

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

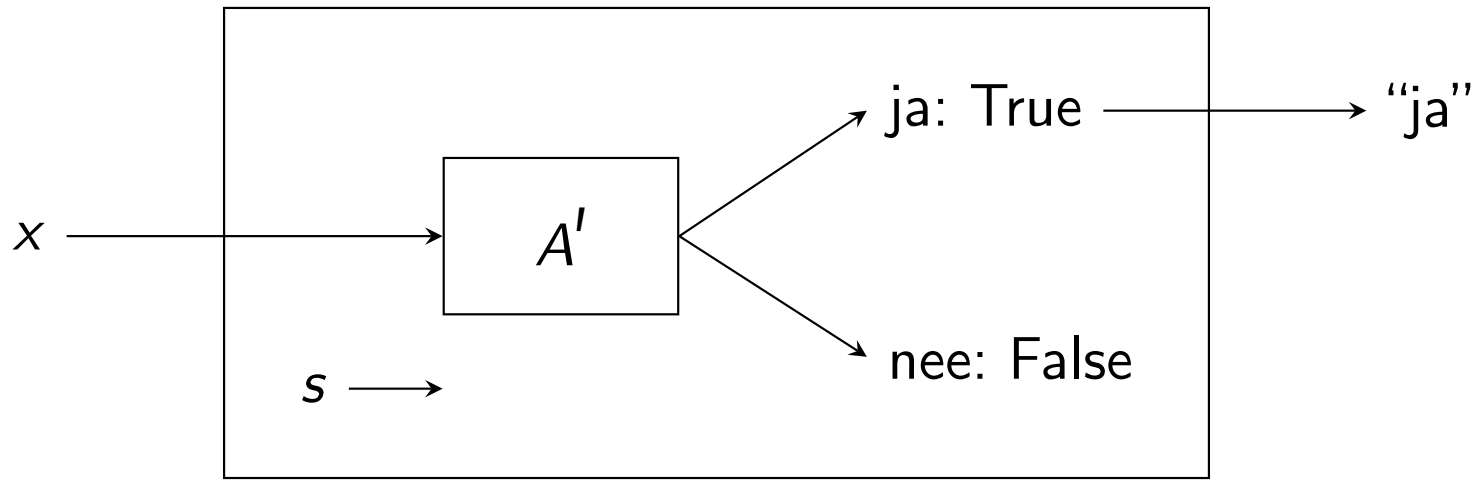
Stelling: $\mathcal{P} \subseteq \mathcal{NP}$

Bewijs:
Neem een willekeurig probleem $P \in \mathcal{P}$. Dan is er een polynomiaal deterministisch algoritme A dat P oplost.



Het volgende niet-deterministische algoritme is dan een polynomiaal begrensd algoritme voor P.

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

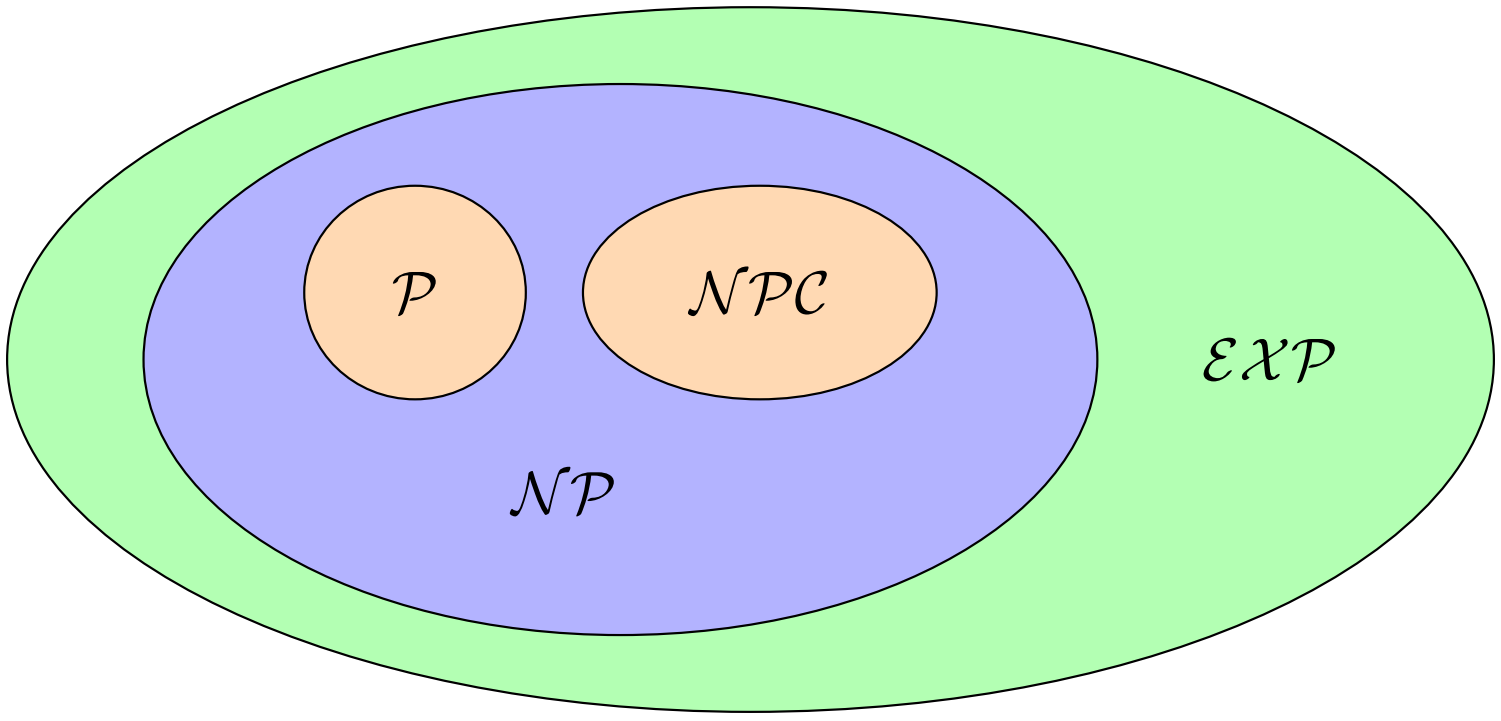


fase 1: s wordt gegenereerd; $O(|s|)$

fase 2: negeer s en voer A uit op x; $O(|x|^{\ell})$

fase 3: $O(1)$

De meest waarschijnlijke relatie



Vermoeden: inclusie $\mathcal{P} \subseteq \mathcal{NP}$ strikt.

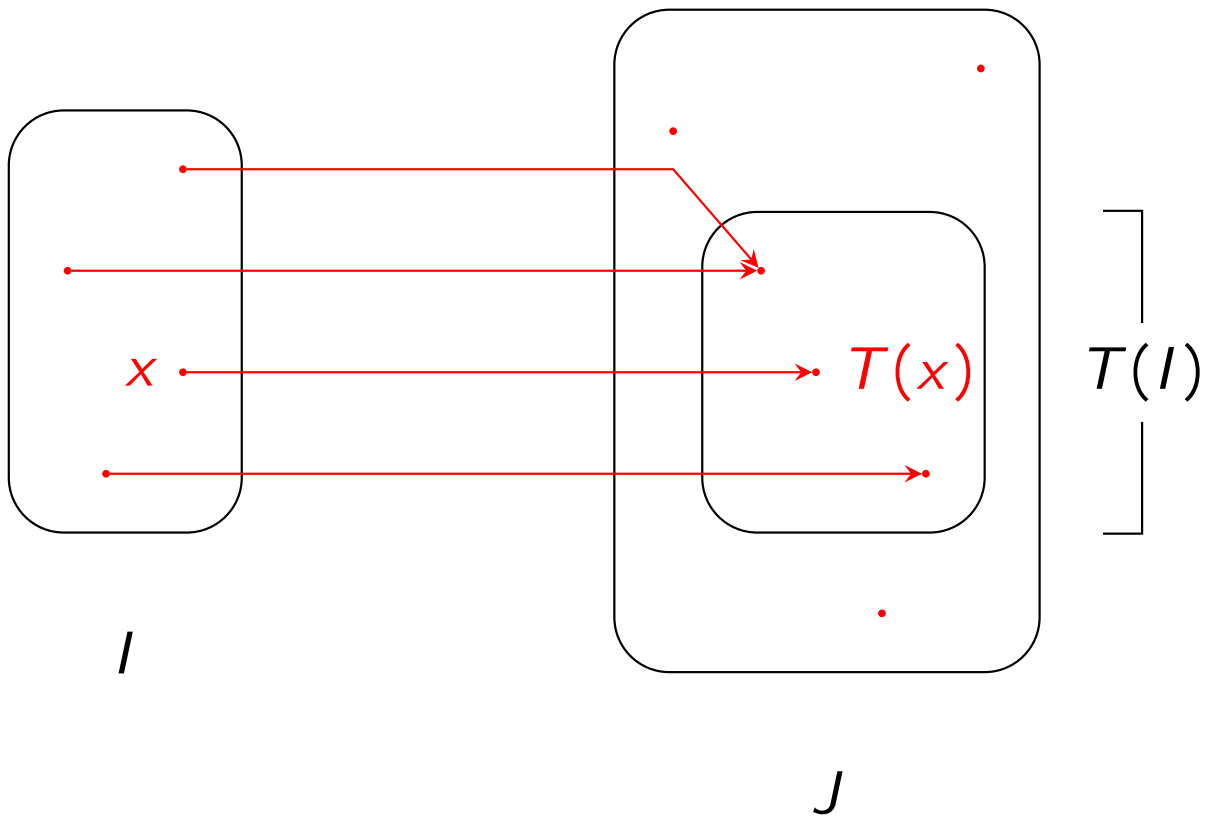
“Moeilijkste”

- Niet mogelijk om van elk probleem gewoon te bepalen “hoe moeilijk” het is; dat was juist het probleem met $\mathcal{NPC}$.
- Wat wél kan: problemen *relateren*. Probleem P is *minstens net zo moeilijk als* probleem Q.
- M.a.w.: als probleem P een efficiënte oplossing heeft, dan heeft Q dat ook.
- Dan is probleem P een moeilijkste probleem in $\mathcal{NP}$ (dus in $\mathcal{NPC}$) als het minstens net zo moeilijk is als álle andere problemen in $\mathcal{NP}$.*
- Relateren $\rightsquigarrow$ *reducties*

*Inclusief de andere “moeilijkste” problemen.

Reducties

Zij $T : I \to J$ een functie van de invoerverzameling I van een beslissingsprobleem P naar de invoerverzameling J van een beslissingsprobleem Q.
T beeldt dus elke $x \in I$ af op een $T(x) \in J$.



Reducties

Definitie

T heet een polynomiale reductie (of *polynomiale transformatie*) van P naar Q als geldt:

1. T kan worden berekend in polynomiaal begrensde tijd (als functie van $|x|$). D.w.z.: de constructie van $T(x)$ uit x kan in $O(|x|^k)$ stappen in de worst case ($k \geq 0$).
2. voor elke x uit I geldt: als x een ja-instantie is voor P dan is $T(x)$ een ja-instantie voor Q.
3. voor elke x uit I geldt: als x een nee-instantie is voor P dan is $T(x)$ een nee-instantie voor Q.

3'. voor elke x uit I geldt: als $T(x)$ een ja-instantie is voor Q dan is x een ja-instantie voor P. (Dit is equivalent met 3.)

Reduceerbaar

Definitie
Een probleem P is polynomiaal reduceerbaar (of polynomiaal transformeerbaar) naar Q als er een polynomiale reductie bestaat van P naar Q.

Notatie: $P \leq_P Q$.

Voorbeeld: HC1 $\leq_P$ HC2

HC1: **gegeven** een *gerichte* graaf $G = (V, E)$.
Vraag: heeft G een Hamiltonkring?

HC2: **gegeven** een *ongerichte* graaf $G = (V, E)$.
Vraag: heeft G een Hamiltonkring?

Bewering: HC1 $\leq_P$ HC2: zie volgende sheet.

Opmerking: er geldt ook: HC2 $\leq_P$ HC1. Bedenk zelf een eenvoudige reductie.

Een reductie van HC1 naar HC2

Transformatie T die een gerichte graaf $G = (V, E)$ afbeeldt op een ongerichte graaf $T(G) = G' = (V', E')$:

- $V' = \{v_1, v_2, v_3 \mid v \in V\}$: elke knoop $v \in V$ wordt afgebeeld op een drietal knopen v_1, v_2, v_3.
- $E' = \{(v_1, v_2), (v_2, v_3) \mid v \in V\} \cup \{(v_3, w_1) \mid (v, w) \in E\}$: binnen elk drietal knopen corresponderend met v loopt een tak tussen v_1 en v_2 en tussen v_2 en v_3, en voor elke tak (pijl) van v naar w in G komt een tak in G' tussen v_3 en w_1.

Een reductie van HC1 naar HC2: vervolg

Dan geldt

1. T kan in polynomiaal begrensde tijd worden berekend (constructie van $T(G)$ kan zeker in $O(|G|^q)$, bijv. met $q = 2$)
2. G is een ja-instantie voor HC1 $\Leftrightarrow$ $T(G)$ is een ja-instantie voor HC2, ofwel: G heeft een gerichte Hamiltonkring $\Leftrightarrow$ G' heeft een ongerichte Hamiltonkring

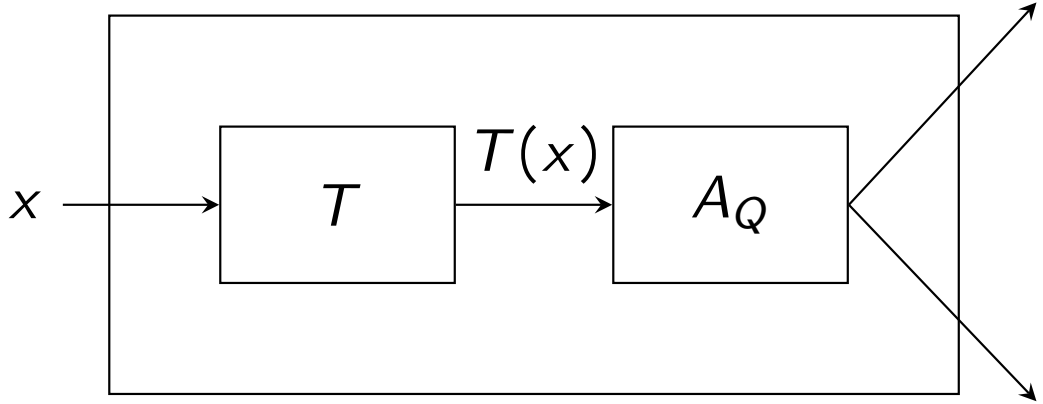
Eigenschappen

Stelling
Als $P \leq_{\mathsf{P}} Q$ en $Q \in \mathcal{P}$, dan ook $P \in \mathcal{P}$.

Lemma
$\leq_{\mathsf{P}}$ is transitief, d.w.z.: als $P_1 \leq_{\mathsf{P}} P_2$ en $P_2 \leq_{\mathsf{P}} P_3$ dan ook $P_1 \leq_{\mathsf{P}} P_3$.

Bewijs stelling

Laat A_Q een polynomiaal deterministisch algoritme zijn voor Q.
Een polynomiaal deterministisch algoritme voor P is dan:



ja $\Leftrightarrow$ $T(x)$ is ja-instantie van Q $\Leftrightarrow$ x is ja-instantie van P

nee $\Leftrightarrow$ $T(x)$ is nee-instantie van Q $\Leftrightarrow$ x is nee-instantie van P

$O(|x|^k) \cdot O(|T(x)|^\ell)$

$O(|x|^{k\ell})$

reductie-eigenschap

NP-hard en NP-volledig

Definitie
Een probleem Q is NP-hard (ook wel: NP-moeilijk) als elk probleem P in $\mathcal{NP}$ polynomiaal reduceerbaar is tot Q: dus $P \leq_{\mathsf{P}} Q$ voor alle $P \in \mathcal{NP}$.

Definitie
Een probleem Q is NP-volledig als

1. $Q \in \mathcal{NP}$
2. Q is NP-hard

Notatie De klasse van NP-volledige problemen geven we aan met $\mathcal{NPC}$ (NP-complete).

NP-volledigheid bewijzen

Stelling

Stel Q is een probleem waarvoor geldt dat $P \leq_{\mathsf{P}} Q$ voor een of andere $P \in \mathcal{NPC}$. Dan is Q NP-hard.
Als bovendien $Q \in \mathcal{NP}$, dan geldt dat $Q \in \mathcal{NPC}$.

Bewijs volgt uit de transitiviteit van $\leq_{\mathsf{P}}$ en de definitie van NP-hard.

M.a.w.: door een bekend NP-volledig probleem te reduceren tot Q reduceren we impliciet alle problemen uit $\mathcal{NP}$ tot Q. Dit geeft ons derhalve een methode om aan te tonen dat een probleem Q NP-volledig is.

Reductiemethode: $Q \in \mathcal{NPC}$

1. Bewijs dat $Q \in \mathcal{NP}$
2. Kies een bekend NP-volledig probleem P
3. Toon aan dat $P \leq_{\mathrm{P}} Q$

   Stap 3 valt uiteen in:

   3a. Geef een functie T van I (de invoerverzameling van P) naar I' (de invoerverzameling van Q) die elke $x \in I$ afbeeldt op een element $T(x)$ van I'

   3b. Laat zien dat $T(x)$ uit x kan worden geconstrueerd in polynomiaal begrensde tijd ($O(|x|^k)$ voor zekere $k \geq 0$)

   3c. Toon aan dat T voldoet aan: $x \in I$ is een ja-instantie voor P $\Leftrightarrow$ $T(x) \in I'$ is een ja-instantie voor Q

Maar...

$$\frac{P \in \mathcal{NPC} \quad P \leq_{\mathsf{P}} Q \quad Q \in \mathcal{NP}}{Q \in \mathcal{NPC}}$$

Wat ontbreekt hier nog?

Maar...

$$\frac{P \in \mathcal{NPC} \quad P \leq_{\mathsf{P}} Q \quad Q \in \mathcal{NP}}{Q \in \mathcal{NPC}}$$

Wat ontbreekt hier nog?

Een **basisgeval**:

$$\frac{}{P_0 \in \mathcal{NPC}}$$

Wordt vervolgd!

Het belangrijkste van vandaag

- zes bekende voorbeeldproblemen $\mathcal{NPC}$
- standaardvorm niet-deterministisch polynomiaal algoritme:
  1. gokfase: genereer string s
  2. verificatiefase: controleer of s een correcte oplossing is in $O(|s|^p \times |x|^q)$
  3. uitvoerfase: "ja" als fase 2 slaagt, anders geen uitvoer
     - als "ja": $|s| \in O(|x|^k)$ en fase 2 dus in $O(|x|^r)$
- polynomiale reductie T van P naar Q: $P \leq_{\mathrm{P}} Q$
  - T beeldt elke invoer x van P af op een invoer $T(x)$ van Q
  - constructie van $T(x)$ uit x is polynomiaal (in $|x|$)
  - reductie-eigenschap: x is ja-instantie voor $P \iff T(x)$ is ja-instantie voor Q

(Werk)college

Volgende college: vrijdag 9 mei, 11u00 – 12u45, BW.0.40

Werkcollege: zodadelijk, 13u15 – 15u00, BW.0.29, BW.0.30
Opgaven: 54, 59abc, 62, 63, 64

Website: `https://liacs.leidenuniv.nl/~graafjmde/COMP`