Complexiteit 2023 — college 8

4 april 2023

Inleiding complexiteitstheorie

De eerste helft van het semester, héél kort samengevat

- Analyse van algoritmen
- "Hoeveel werk" doet een gegeven algoritme?
  - Tellen basisoperaties
  - In orde van grootte: O, Ω, Θ, best/worst/average case
- "Hoeveel werk" kost een probleem minstens
  - Ondergrens voor worst case van elk algoritme (op basis van specifieke basisoperatie)
  - Beslissingsbomen, adversary-argumenten, ad hoc
- Problemen en algoritmen voor:
  - Zoeken en selectie
  - Sorteren
  - Polynoomevaluatie en matrixvermenigvuldiging
  - Eulerkringen en Hamiltonkringen

Complexiteitstheorie

De tweede helft van het semester

Complexiteitstheorie

(meer overzicht volgt aan het eind van het college)

Vandaag

- Inleiding complexiteitstheorie
- Handelbare, onhandelbare en onbeslisbare problemen
- De complexiteitsklassen $\mathcal{P}$ (formeel) en $\mathcal{NP}$ en $\mathcal{NPC}$ (informeel)

Handelbaar/onhandelbaar

n	10	100	$1\,000$	$100\,000$	$10\,000\,000$
$\lg n$	3	6	9	16	23
$\sqrt{n}$	3	10	32	316	$3\,162$
n	10	100	$1\,000$	$100\,000$	$10\,000\,000$
$n \lg n$	33	664	$9\,966$	$1\,660\,964$	$\approx 10^8$
n^2	100	$10\,000$	$1\,000\,000$	10^{10}	10^{14}
n^{10}	10^{10}	10^{20}	10^{30}	10^{50}	10^{70}
2^n	$1\,024$	$\approx 10^{30}$	$\approx 10^{301}$	$\approx 10^{30\,102}$	veel
$n!$	$3\,628\,800$	$\approx 10^{157}$	$\approx 10^{2\,567}$	$\approx 10^{456\,573}$	veel
n^n	10^{10}	10^{200}	$10^{3\,000}$	$10^{500\,000}$	$10^{70\,000\,000}$

Ter vergelijking:

- het aantal protonen in het heelal heeft 79 cijfers
- het aantal microseconden sinds de Oerknal heeft 24 cijfers

Handelbaar/onhandelbaar

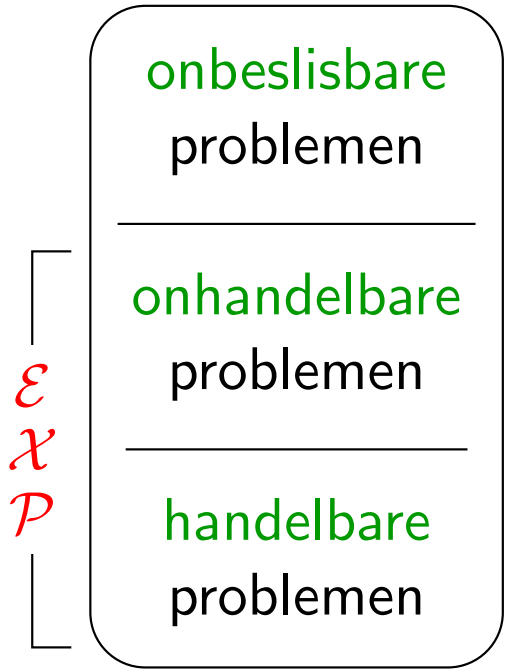
Met één miljoen (10^6) instructies per seconde:

n	10	20	50	100	300
n^2	$\frac{1}{10\,000}$ sec	$\frac{1}{2\,500}$ sec	$\frac{1}{400}$ sec	$\frac{1}{100}$ sec	$\frac{9}{100}$ sec
n^5	$\frac{1}{10}$ sec	3,2 sec	5,2 min	2,8 uur	28,1 dag
2^n	$\frac{1}{1000}$ sec	1 sec	35,7 jaar	400 biljoen eeuwen	75-cijfers veel eeuwen
n^n	2,8 uur	3,3 biljoen jaar	70-cijfers veel eeuwen	185-cijfers veel eeuwen	728-cijfers veel eeuwen

Ter vergelijking: de Oerknal was ongeveer 15 miljard jaar geleden.
($\approx 2 \times 13!$ en $\approx 2^{34}$)

Indeling problemen

We kunnen problemen globaal indelen in drie klassen:



Problemen waarvoor geen algoritme mogelijk is (stop/halting-probleem)

Problemen waarvoor geen polynomiaal algoritme mogelijk is

Problemen waarvoor een polymiaal algoritme bestaat: $\mathcal{P}$

Beslisbare problemen:

- in acceptabele tijd oplosbaar (deterministisch)
- in niet-acceptabele tijd oplosbaar (deterministisch)

$\mathcal{P}$ en $\mathcal{EXP}$

$\mathcal{EXP}$ is de klasse van problemen waarvoor een exponentieel algoritme bestaat, dus met worst case-complexiteit $O(2^{p(n)})$ voor een of ander polynoom p en n een maat voor de invoergrootte. Er bestaan ook nog $2\mathcal{EXP}$, $3\mathcal{EXP}$, ...

$\mathcal{P}$ is de klasse van problemen* waarvoor een polynomiaal algoritme bestaat, dus met worst case-complexiteit $O(p(n))$.

Onder $\mathcal{EXP}$ vallen uiteraard ook alle polynomiaal oplosbare problemen: $\mathcal{P} \subset \mathcal{EXP}$.

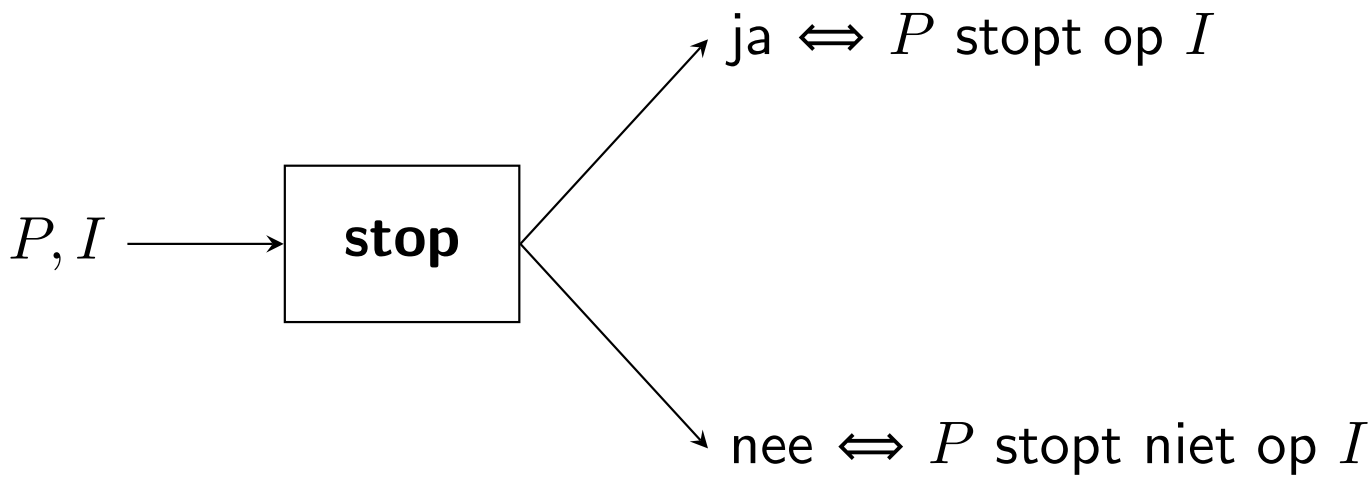
De moeilijkste problemen uit $\mathcal{EXP}$ zijn problemen die echt een exponentieel algoritme nodig hebben, dus problemen met $\Theta(2^{p(n)})$ als ondergrens voor de complexiteit. Problemen die (minstens) exponentieel zijn noemen we onhandelbaar.

*later beperken we ons tot beslissingsproblemen

Halting-probleem

Stopprobleem:

Bestaat er een algoritme **stop** dat voor een willekeurig programma P en invoer I kan bepalen of P wel of niet stopt op invoer I?



De klasse $\mathcal{P}$

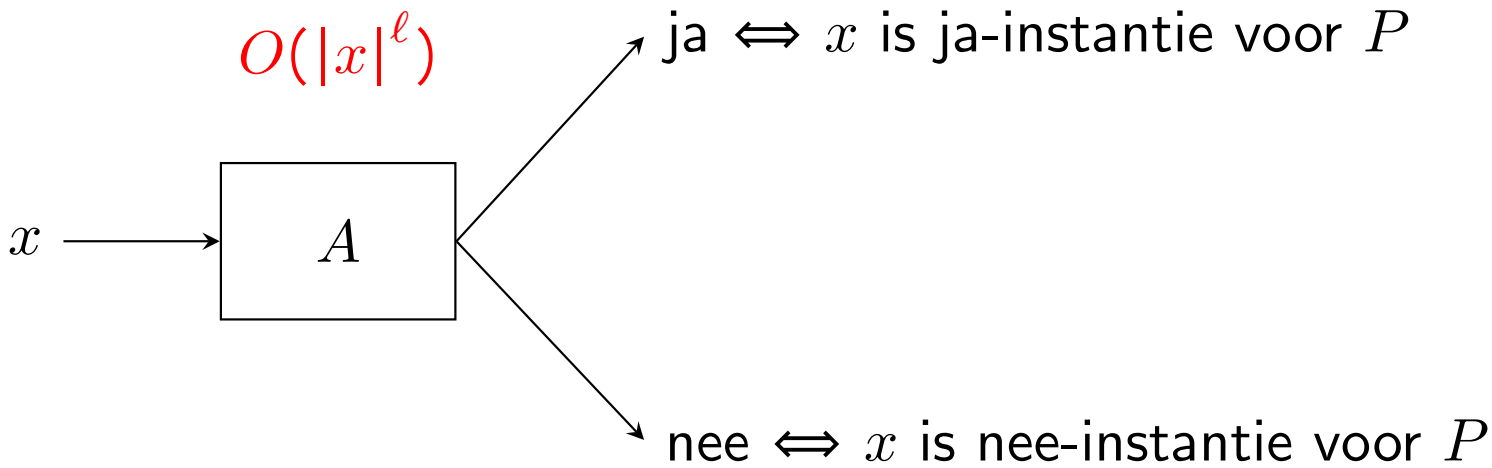
Definitie
Een algoritme heet polynomiaal begrensd als zijn worst case-complexiteit van boven is begrensd door een functie die polynomiaal is in de lengte van de invoer. Dus: worst case is $O(p(n))$ voor een zeker polynoom p, en met n (een maat voor) de lengte van de invoer.
Eenvoudiger geformuleerd: worst case is $O(|x|^{\ell})$ met $|x|$ de lengte van de invoer x en $\ell \geq 0$.

Definitie
Een probleem heet polynomiaal begrensd als er een polynomiaal begrensd algoritme voor bestaat.

Definitie
De klasse van beslissingsproblemen die polynomiaal begrensd zijn noteren we als $\mathcal{P}$.

$P \in \mathcal{P}$

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



Een algoritme is deterministisch als het elke keer dat het wordt uitgevoerd op dezelfde invoer (instantie) hetzelfde doet, en dus dezelfde uitvoer oplevert.

Handelbaar $\leftrightarrow \mathcal{P}$

Vraag: waarom handelbaar = polynomiaal begrensd?

- als een probleem niet in $\mathcal{P}$ zit, is het zeker onhandelbaar
- de klasse $\mathcal{P}$ heeft mooie afsluitingseigenschappen: een algoritme dat bestaat uit een eindige opeenvolging en/of samenstelling van polynomiaal begrensde algoritmen is zelf ook weer polynomiaal begrensd (zie de opgaven)
- de klasse $\mathcal{P}$ is onafhankelijk van het berekeningsmodel en van gebruikte coderingen: als een probleem polynomiaal is begrensd in het ene model, dan ook in een ander model (voor alle redelijke/praktische berekeningsmodellen)

$\mathcal{EXP}$ en $\mathcal{NPC}$

Het blijkt buitengewoon moeilijk te zijn om te bewijzen dat voor een probleem ieder algoritme complexiteit $\Omega(2^{p(n)})$ heeft (dus minstens exponentieel is in n).

Er zijn maar relatief weinig niet-triviale problemen waarvoor zo'n ondergrens is bewezen. Bijvoorbeeld: gegeneraliseerd schaken op een $n \times n$ bord (is er een winnende strategie?), en een aantal problemen uit de formele talen en logica.

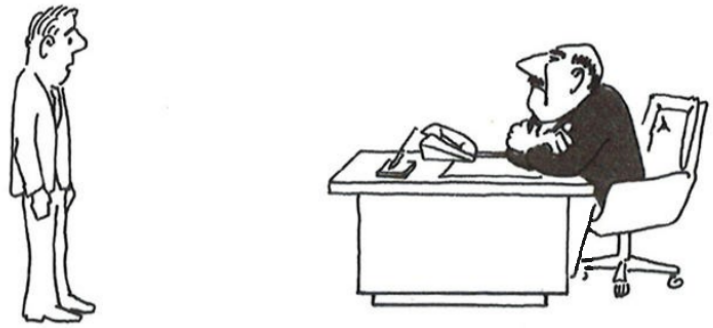
Er zijn echter heel veel problemen waarvoor we geen polynomiaal algoritme hebben, maar ook geen exponentiële ondergrens $\Rightarrow \mathcal{NPC}$.

$\mathcal{NPC}$

De klasse van NP-volledige problemen $\mathcal{NPC}$ (Engels: NP-complete) heeft enkele interessante eigenschappen, zoals:

1. voor geen enkel NP-volledig probleem is tot dusver een polynomiaal algoritme gevonden. Men vermoedt dus dat ze onhandelbaar zijn, maar dat heeft tot dusver ook nog niemand kunnen bewijzen.
2. als er een polynomiaal algoritme bestaat voor ook maar één NP-volledig probleem, dan is meteen elk NP-volledig probleem in polynomiale tijd oplosbaar.
3. omgekeerd: als er van één enkel NP-volledig probleem wordt bewezen dat het onhandelbaar is, dan zijn alle NP-volledige problemen onhandelbaar.

:)

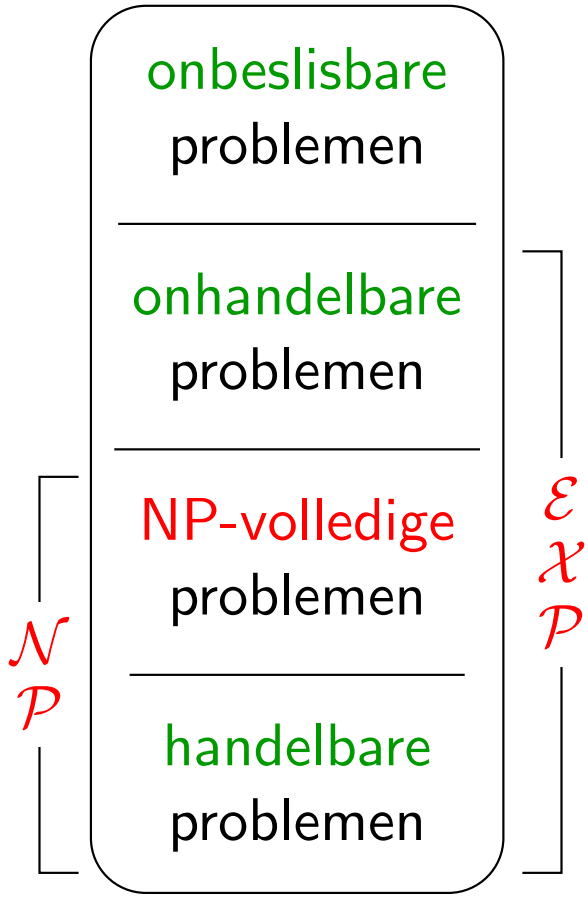


"I can't find an efficient algorithm, I guess I'm just too dumb."

"I can't find an efficient algorithm, but neither can all these famous people."

M.R. Garey en D.S. Johnson, Computers and intractability, Freeman, 1979

Uitgebreidere indeling



Problemen waarvoor geen algoritme mogelijk is (stop/halting-probleem)

Problemen waarvoor geen **deterministisch** polynomiaal algoritme mogelijk is

???

Problemen waarvoor een **deterministisch** polynomiaal algoritme bestaat: $\mathcal{P}$

$\mathcal{NP}$ is de klasse van problemen waarvoor een niet-deterministisch polynomiaal algoritme bestaat. We zullen dit allemaal nog preciezer maken.

$\mathcal{NP}$ informeel

$\mathcal{P}$ is de klasse van problemen waarvoor een deterministisch polynomiaal algoritme bestaat.
$\mathcal{NP}$ is de klasse van problemen waarvoor een niet-deterministisch polynomiaal algoritme bestaat (precieze definitie later).

Niet-deterministisch: je mag een oplossing gokken.
Polynomiaal: deze kan daarna in polynomiale tijd worden gecontroleerd.

$\mathcal{P}$: beslissingsproblemen die polynomiaal oplosbaar zijn.
$\mathcal{NP}$: beslissingsproblemen die polynomiaal verifieerbaar zijn (althans, de ja-instanties; laat je niet foppen).
$\mathcal{NPC}$: bevat de moeilijkste problemen uit $\mathcal{NP}$.

M.a.w.: voor $\mathcal{P}$ is oplossing in polynomiale tijd te *vinden*, voor $\mathcal{NP}$ is oplossing in polynomiale tijd te *controleren*.

$\mathcal{P}$, $\mathcal{NP}$ en $\mathcal{NPC}$

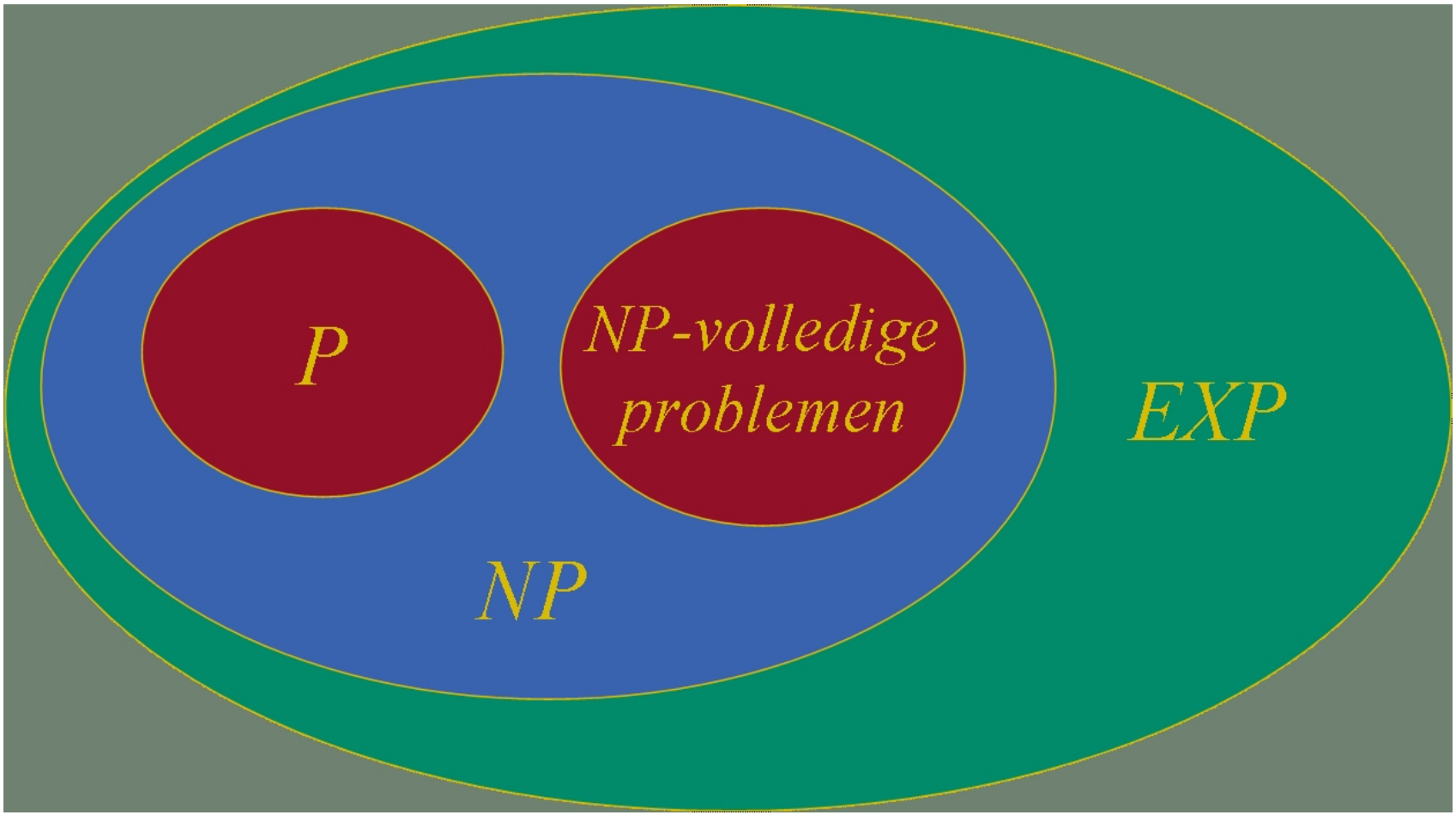
We weten dat $\mathcal{P} \subseteq \mathcal{NP}$ en $\mathcal{NPC} \subseteq \mathcal{NP}$. In theorie kan het zijn dat $\mathcal{P} = \mathcal{NP}$, maar dat is onwaarschijnlijk (zie later).

Wat niet kan, is dat $\mathcal{P} \neq \mathcal{NP}$ en $\mathcal{P} \cup \mathcal{NPC} = \mathcal{NP}$. Met andere woorden, als $\mathcal{P}$ ongelijk is aan $\mathcal{NP}$, dan zijn er problemen in $\mathcal{NP}$ die *noch* in $\mathcal{P}$ zitten, noch in $\mathcal{NPC}$ (Ladner, 1975). Deze problemen worden ook wel $\mathcal{NPI}$ (NP-intermediate) genoemd.

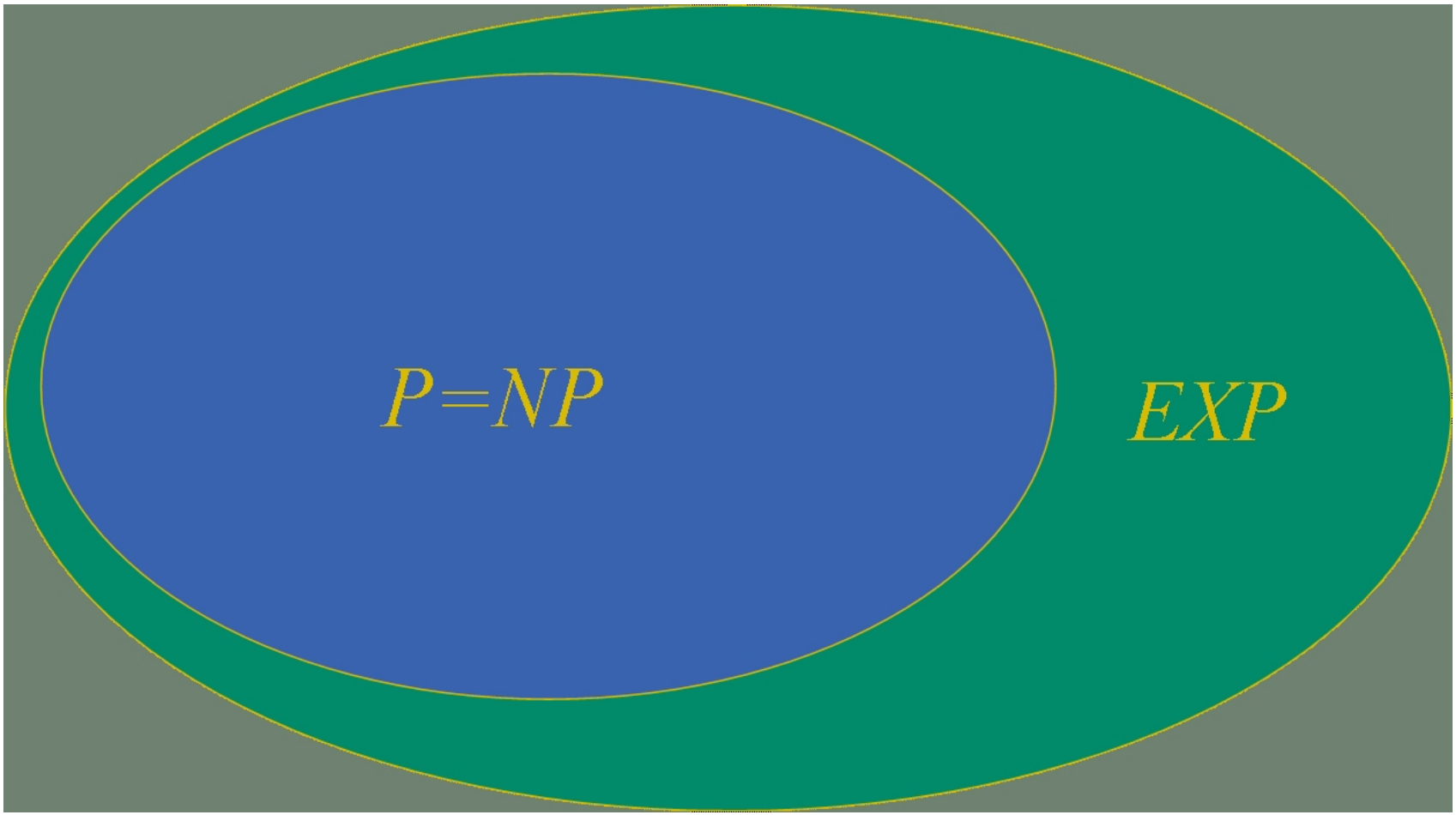
Het bestaan van $\mathcal{NPI}$-problemen is hypothetisch. Ze bestaan onder de aanname dat $\mathcal{P} \neq \mathcal{NP}$, dat weten we zeker, maar we hebben nog geen probleem kunnen identificeren. Er zijn wel enkele veelbelovende kandidaten, zoals het graafisomorfismeprobleem*.

*László Babai (2015–2017): oplosbaar in $O(2^{p(\lg n)})$, quasi-polynomiaal

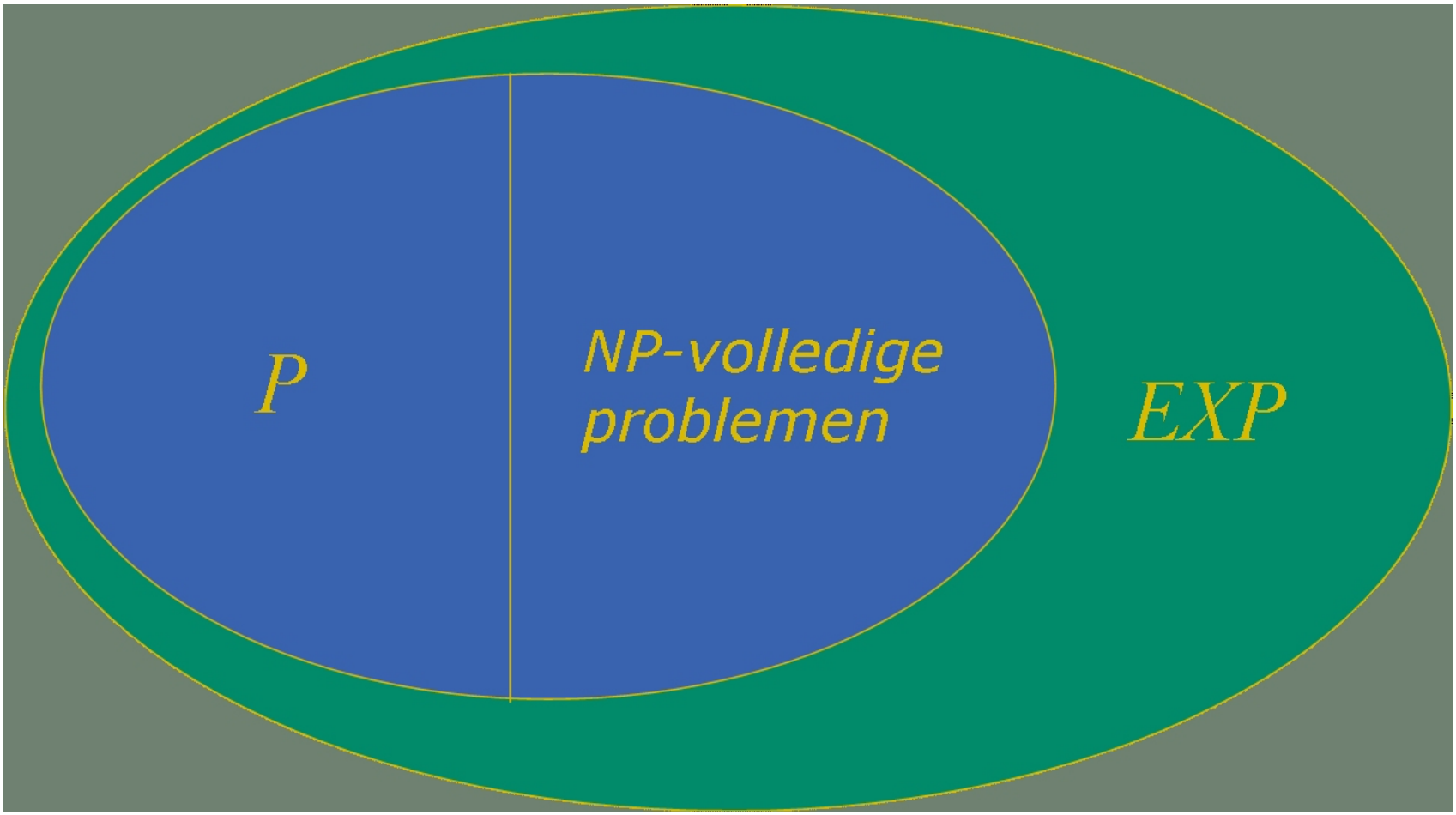
De meest waarschijnlijke relatie



Onwaarschijnlijk, maar mogelijk



Elegant, maar onmogelijk



Beslissingsproblemen

De theorie van NP-volledigheid beperkt zich tot beslissingsproblemen. Bij een beslissingsprobleem zijn slechts twee antwoorden mogelijk: ja of nee.

Probleeminvoer waarop het antwoord *ja* is noemen we ja-instanties; als het antwoord *nee* is spreken we van nee-instanties.

Optimalisatieproblemen worden omgezet naar beslissingsproblemen, en wel zo dat geldt: als het optimalisatieprobleem handelbaar is, dan is het corresponderende beslissingsprobleem dat ook. En dus ook omgekeerd: **als het beslissingsprobleem onhandelbaar is, dan is het corresponderende optimalisatieprobleem dat ook**.

Berekeningsmodel

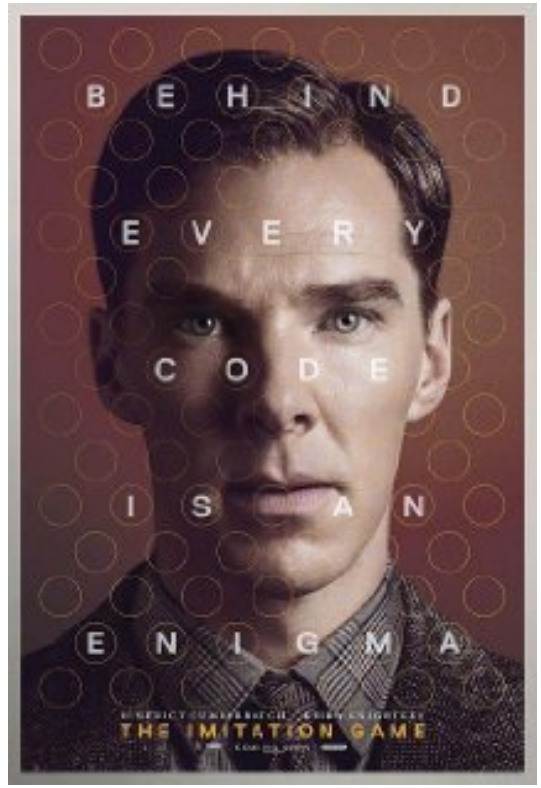
De klassen $\mathcal{P}$ en $\mathcal{NP}$ worden formeel gedefinieerd met behulp van Turing machines. Een Turing machine is een uiterst simpel maar krachtig model van een 'computer'.

Elk probleem dat met een 'normaal' computerprogramma in polynomiale tijd is op te lossen, blijkt ook polynomiaal oplosbaar te zijn met behulp van een Turing machine, en omgekeerd.

We mogen dus gewoon C++-achtige algoritmen blijven gebruiken, maar het is nuttig om iets te weten over de werking van Turing machines. Zie ook het vak Computability.

Alan Turing

1912–1954

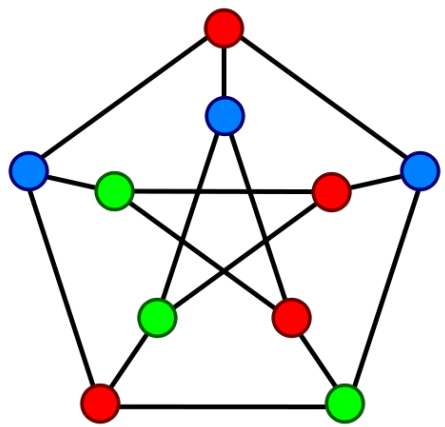


2015

Voorbeelden $\mathcal{NPC}$-problemen

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

- CNF-satisfiability
- Subset Sum
- Hamiltonkring
- Handelsreizigersprobleem
- Kliek
- Graafkleuring



CNF-satisfiability

- een literal is een Boolese variabele of de negatie daarvan (dus x of $\neg x$).
- een clause (clausule) is een disjunctie ($\vee$) van literals.
  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

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 ϕ is dus een ja-instantie voor SAT.

Subset Sum

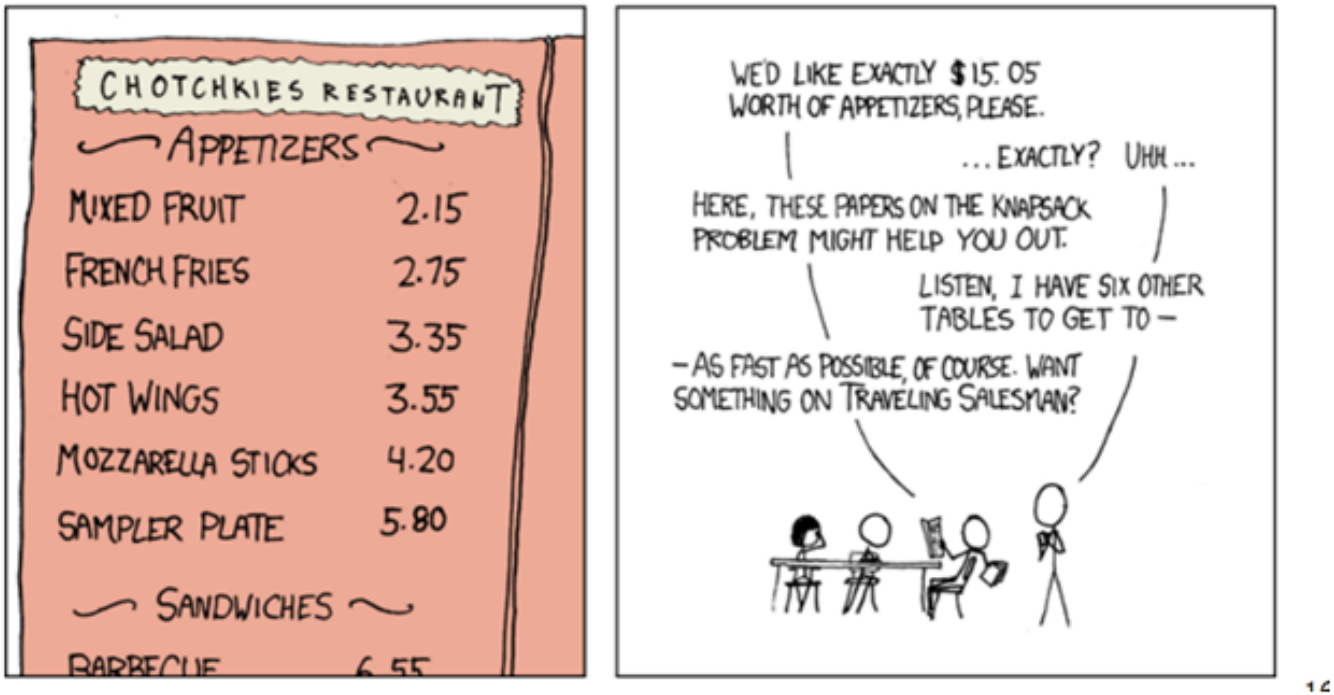
Beslissingsprobleem SUM

Gegeven een getal $t \in \mathbb{N}$ en een eindige verzameling $S \subset \mathbb{N}$.
Bestaat er een deelverzameling $S' \subseteq S$ met $\sum_{s \in S'} s = t$?

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 $S' = \{7, 3, 9, 11\}$.

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 $S' = \{1, 4, 16, 256, 1040, 1093, 1344\}$.

:)



https://xkcd.com/287/

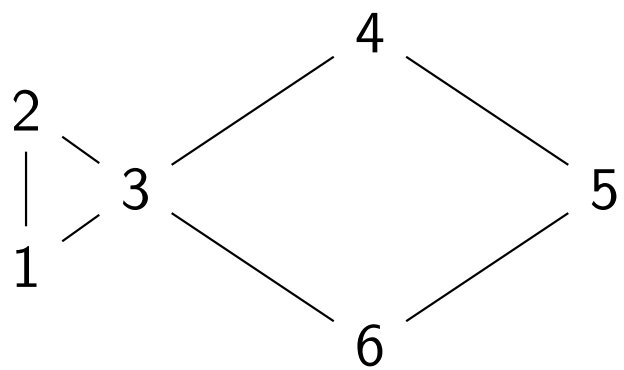
Hamiltonkring

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

Beslissingsprobleem HC

Gegeven een graaf $\mathcal{G} = (V, E)$. Heeft $\mathcal{G}$ een Hamiltonkring?

Voorbeeld



is een nee-instantie voor HC.

Handelsreizigersprobleem

Handelsreizigersprobleem of Travelling Salesperson Problem (TSP)

Optimalisatieprobleem

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

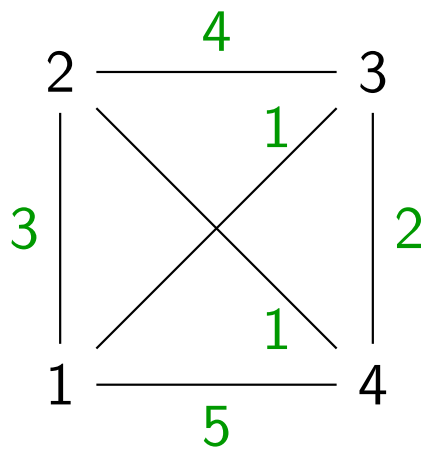
Beslissingsprobleem TSP

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

*tussen elk tweetal knopen van $\mathcal{G}$ 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.



Kliek

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

Optimalisatieprobleem

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

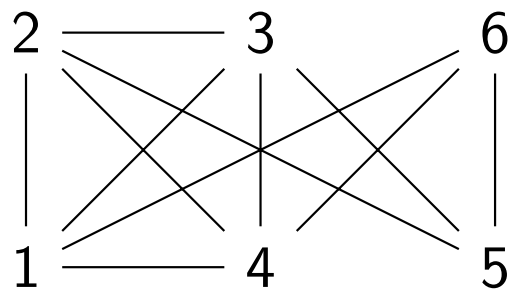
Beslissingsprobleem Kliek

Gegeven een ongerichte graaf $\mathcal{G} = (V, E)$ en een geheel getal k $(0 \leq k \leq |V|)$. Bestaat er in $\mathcal{G}$ 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: $\{1, 2, 3, 4\}$.



Graafkleuring

Een kleuring van (de knopen van) een ongerichte graaf $\mathcal{G} = (V, E)$ is een afbeelding $c : V \mapsto S$, 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 $\mathcal{G} = (V, E)$. Vind een kleuring van $\mathcal{G}$ met zo weinig mogelijk kleuren.

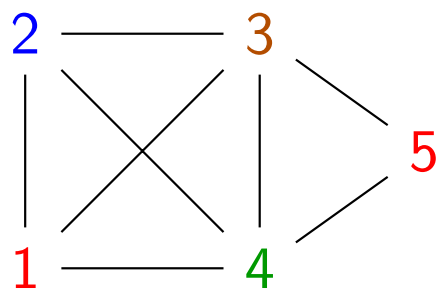
Beslissingsprobleem Kleur

Gegeven een ongerichte graaf $\mathcal{G} = (V, E)$ en een geheel getal $k > 0$. Bestaat er een kleuring van $\mathcal{G}$ die hooguit k kleuren gebruikt (ofwel: is $\mathcal{G}$ 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.



Gemeenschappelijk

1. voor alle zes voorbeeldproblemen is eenvoudig een exponentieel algoritme op te schrijven.
2. in alle gevallen kan in polynomiale tijd worden gecontroleerd of een kandidaatoplossing een echte oplossing (= "dat wat je zoekt") is.
3. voor al deze problemen geldt: het lijkt extreem moeilijk (exponentieel) te zijn om voor gegeven invoer x te bepalen of het antwoord "ja" of "nee" moet zijn.
4. echter, *als* x een ja-instantie is, dan is er een eenvoudige (polynomiale) manier om iemand daarvan te overtuigen.

Gemeenschappelijk

5. voor ja-instanties bestaat er een zogenaamd certificaat dat kan worden gebruikt om te laten zien dat het antwoord inderdaad "ja" is. In de voorbeelden is dit steeds de gezochte oplossing, d.w.z. dat wat de invoer zou moeten bezitten.
6. bovendien is dit certificaat kort (polynomiaal) en kan verifiëren / controleren ervan in polynomiale tijd.
7. eigenschap 2 t/m 6: de genoemde problemen zitten in $\mathcal{NP}$. Later wordt alles formeler gemaakt.
8. ja-instanties zijn eenvoudig te verifiëren met de juiste hint (certificaat). Hoe zit het met nee-instanties?

Om te onthouden van vandaag

- handelbare en onhandelbare problemen: wel/geen *deterministisch* polynomiaal algoritme
- $\mathcal{P}$: oplossing in polynomiale tijd te *vinden*, handelbaar; $\mathcal{EXP} \setminus \mathcal{P}$: oplossing niet in polynomiale tijd te vinden; onhandelbaar
- $\mathcal{NP}$: *niet-deterministisch* polynomiaal algoritme; oplossing in polynomiale tijd te *controleren*
- $\mathcal{NPC}$: moeilijkste problemen in $\mathcal{NP}$; aantal welbekende voorbeelden; handelbaarheid onbekend
- beslissingsproblemen en optimalisatieproblemen: zelfde handelbaarheid

De komende weken

- de klasse $\mathcal{NP}$ formeler gemaakt
- polynomiale reducties en de klasse $\mathcal{NPC}$
- de stelling van Cook, iets over MIP- en SAT-solvers
- benaderingsalgoritmen
- gastcollege Hendrik Jan Hoogeboom: complexiteit en spellen*

*details onder voorbehoud

(Werk)college

Volgende college: dinsdag 11 april, 9u00–10u45, zaal 412 (Snellius)

Werkcollege: zodadelijk van 11u00 tot 12u45, computerzalen 302–304 en 303 (Snellius, onderaan de trap)
Opgaven uit het dictaat: 55, 10, 45, 46, 33