Negende college complexiteit

9 april 2019

NP-volledigheid I: introductie

N	10	50	100	300	1000
$\log_2 N$	3	5	6	8	9
$5N$	50	250	500	1500	5000
$N \cdot \log_2 N$	33	282	665	2469	9966
N^2	100	2500	10.000	90.000	7 cijfers
N^3	1000	125.000	7 cijfers	8 cijfers	10 cijfers
2^N	1024	16 cijfers	31 cijfers	91 cijfers	302 cijfers
$N!$	7 cijfers	65 cijfers	161 cijfers	623 cijfers	onvoorstelbaar
N^N	11 cijfers	85 cijfers	201 cijfers	744 cijfers	onvoorstelbaar

Ter vergelijking:

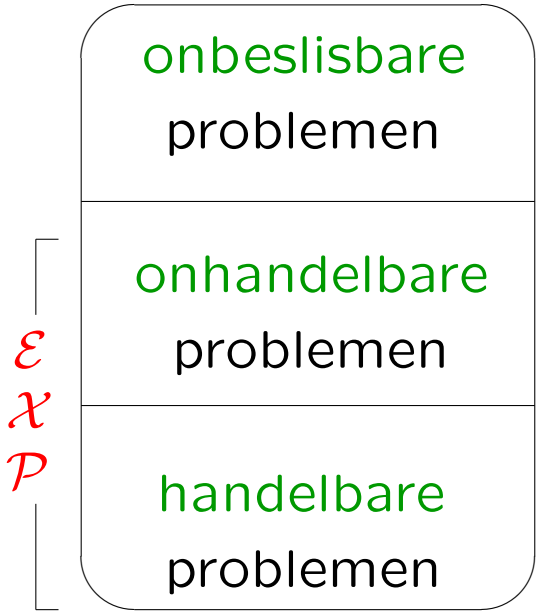
het aantal protonen in het heelal is een getal met 79 cijfers
het aantal microseconden sinds de Oerknal heeft 24 cijfers

Stel dat de computer 1 instructie doet per microseconde (10^{-6} sec).

N	10	20	50	100	300
N^2	$\frac{1}{10000}$ sec	$\frac{1}{2500}$ 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.

We kunnen problemen globaal in drie klassen indelen:



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

Problemen waarvoor geen polynomiaal algoritme mogelijk is

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

Beslisbare problemen: - in acceptabele tijd oplosbaar*

- niet in acceptabele tijd oplosbaar*

* met een deterministisch algoritme

$\mathcal{EXP}$ 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 $O(p(n))$.

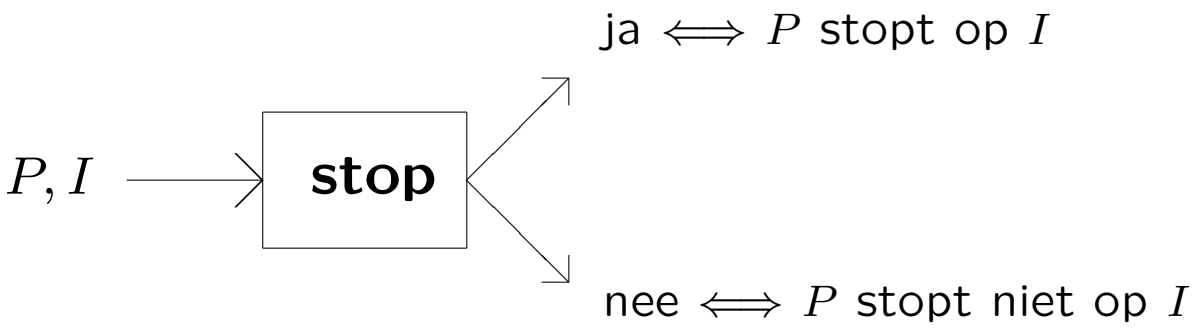
Onder $\mathcal{EXP}$ vallen uiteraard ook alle polynomiaal oplosbare problemen: $\mathcal{P} \subset \mathcal{EXP}$.

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

Stopprobleem:

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



Definitie

Een algoritme heet polynomiaal begrensd als zijn worst case complexiteit van boven begrensd is 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 $\ell \geq 0$.

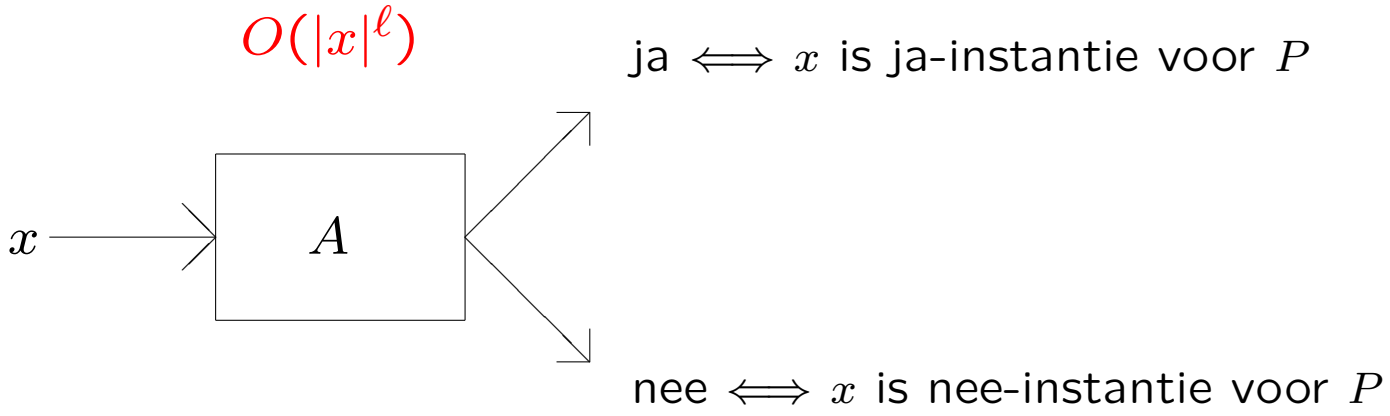
Definitie

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 $\mathcal{P}$.

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.

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 begrensd is in het ene model, dan ook in een ander model (geldt voor alle redelijke/praktische berekeningsmodellen)

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 bewezen is. Bijvoorbeeld: gegeneraliseerd schaken op een $n \times n$ bord (is er een winnende strategie?), en een aantal problemen uit de formele talen theorie en logica.

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

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 dat ze onhandelbaar zijn, maar dat heeft tot dusver ook nog niemand kunnen bewijzen.

2. Als er een polynomiaal algoritme bestaat voor willekeurig welk NP-volledig probleem, dan is meteen elk NP-volledig probleem in polynomiale tijd oplosbaar.

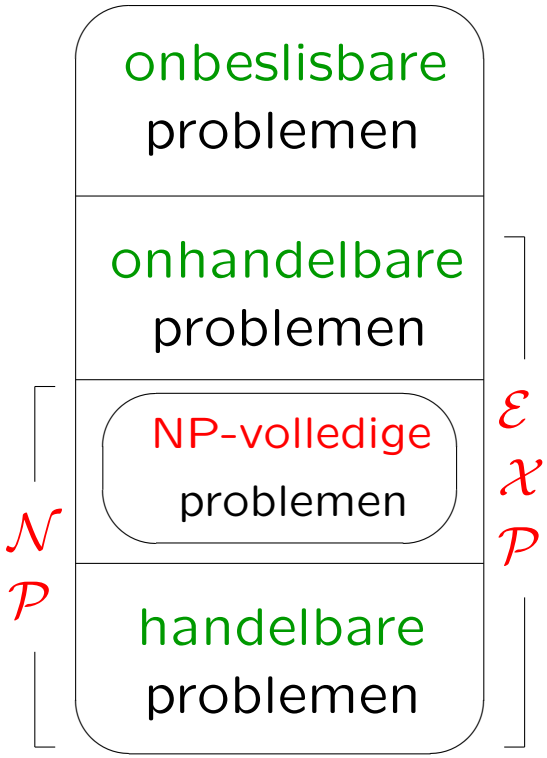
3. Omgekeerd: als er van één enkel NP-volledig probleem bewezen wordt dat het onhandelbaar is, dan zijn alle NP-volledige problemen onhandelbaar.

"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.



Problemen waarvoor geen algoritme mogelijk is (stop-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.

$\mathcal{P}$ 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 geverifieerd/gecontroleerd worden.

$\mathcal{P}$: 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 $\mathcal{NP}$.

$\mathcal{P}$: oplossing in polynomiale tijd te *vinden*

$\mathcal{NP}$: oplossing in polynomiale tijd te *controleren*

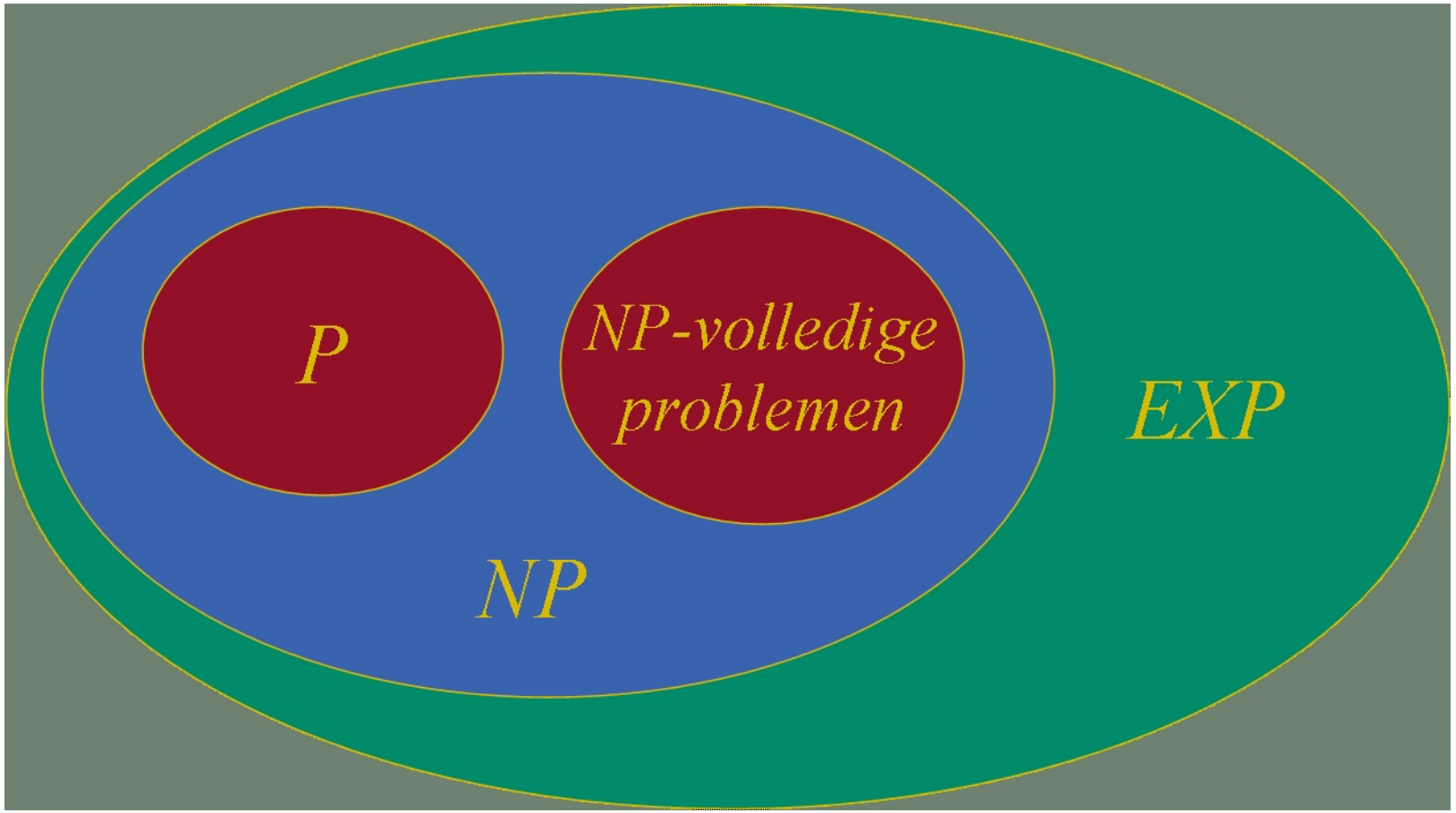
We weten dat $\mathcal{P} \subseteq \mathcal{NP}$ en $\mathcal{NPC} \subseteq \mathcal{NP}$. In theorie kan het zijn dat $\mathcal{P} = \mathcal{NP}$, maar dat is onwaarschijnlijk (zie later).

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.

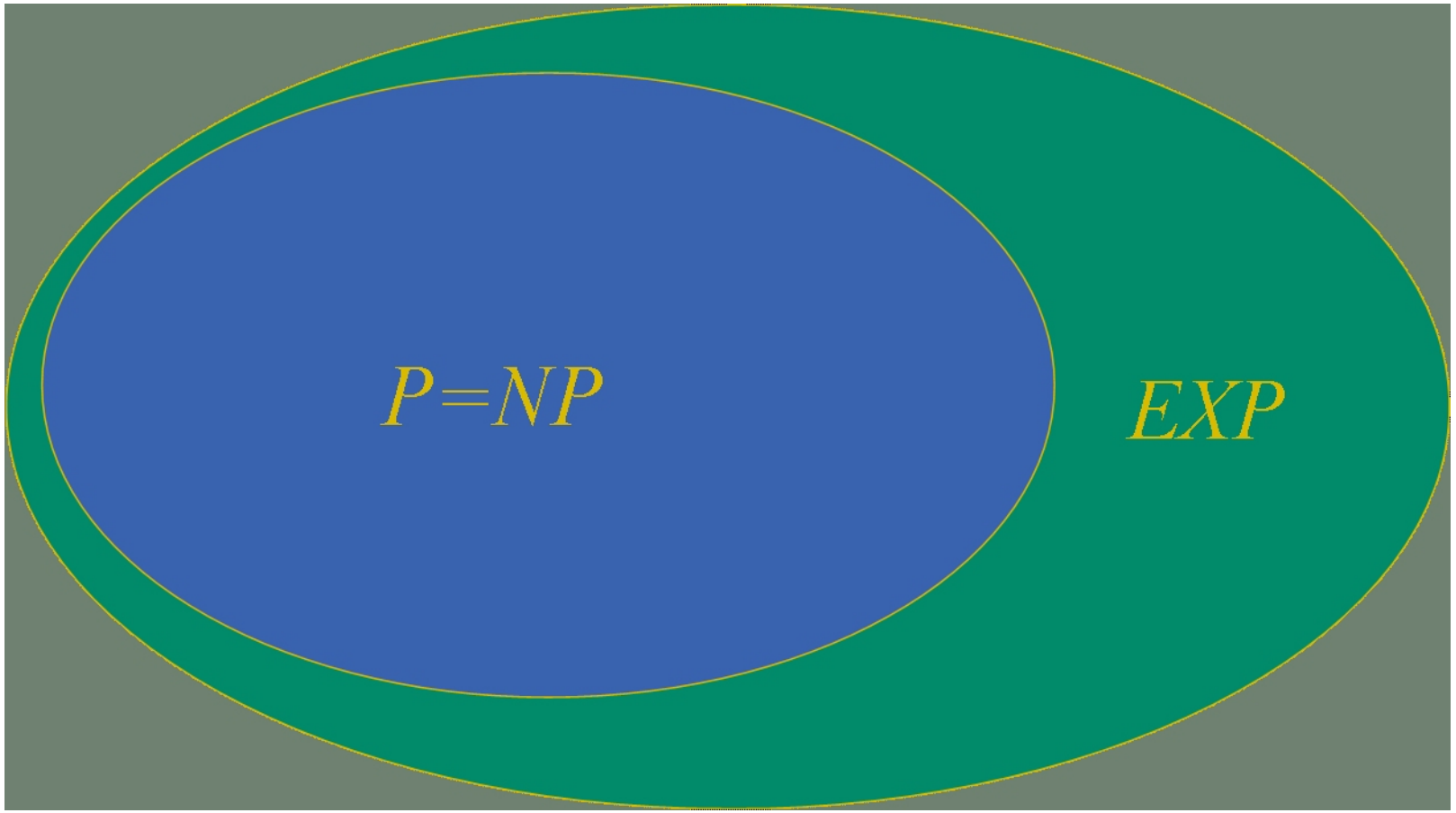
Het bestaan van $\mathcal{NPI}$-problemen is hypothetisch. Ze bestaan onder de aanname dat $\mathcal{P} \neq \mathcal{NP}$, dat weten we zeker, maar we hebben nog geen probleem kunnen identificeren. Er zijn wel enkele veelbelovende kandidaten, zoals het graafisomorfisme probleem*.

*László Babai (2015–17): oplosbaar in $O(2^{p(\lg n)})$, quasi-polynomiaal

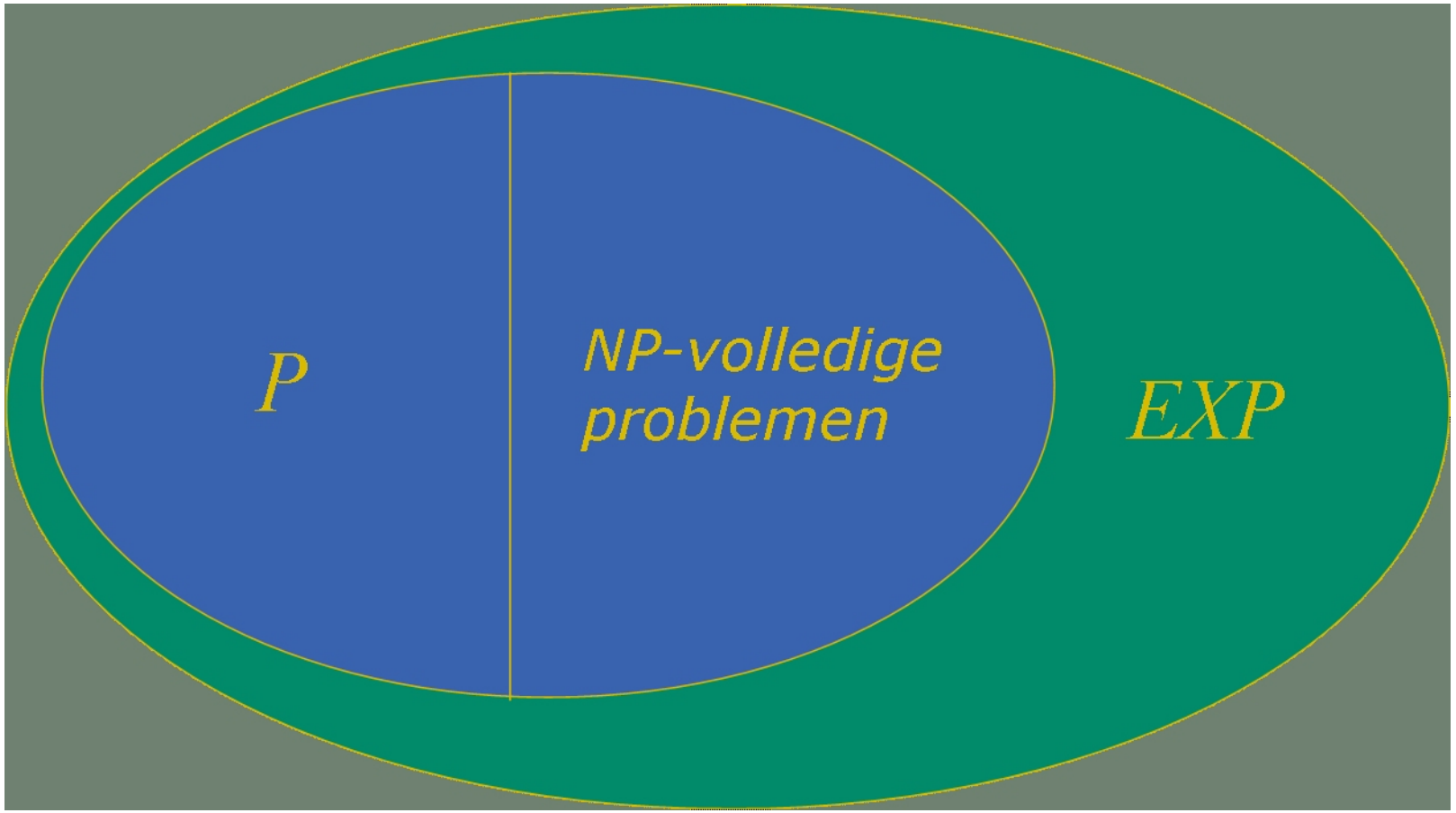
De meest waarschijnlijke relatie



Onwaarschijnlijk, maar mogelijk



Elegant, maar onmogelijk



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.**

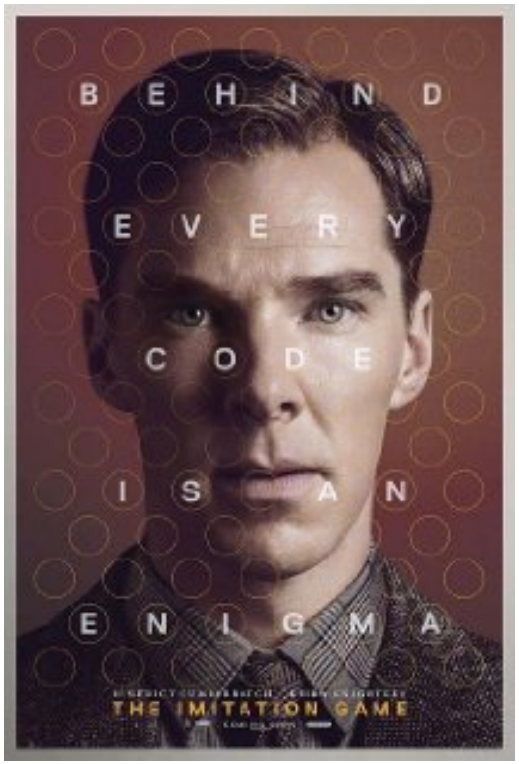
De klassen $\mathcal{P}$ en $\mathcal{NP}$ worden formeel gedefinieerd met behulp van Turing machines. Een Turing machine is een uiterst simpel, maar krachtig model van een 'computer'.

Elk probleem dat met een 'normaal' computerprogramma in polynomiale tijd op te lossen is, 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 weten over de werking van Turing machines. Zie ook Fundamentele informatica 3.

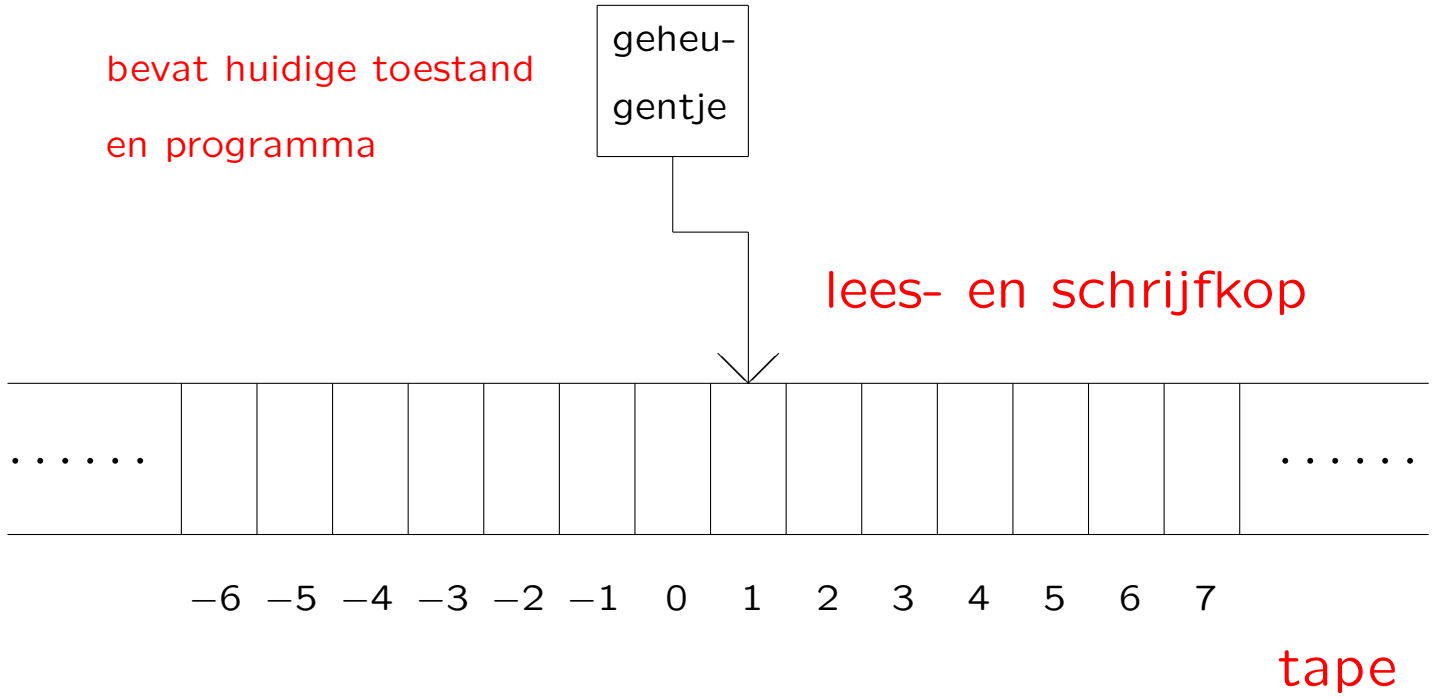
Alan Turing

1912–1954



2015

Deterministische one-tape Turing machine (DTM)



Een DTM-programma bevat:

- Γ: een eindige verzameling tape-symbolen (waaronder een invoeralfabet Σ en blanco). Voorbeeld: $\Gamma = \{0, 1, b\}$.
- Q: een eindige verzameling toestanden, waaronder een begintoestand q_0 en twee eindtoestanden q_Y en q_N. Voorbeeld: $Q = \{q_0, q_1, q_2, q_3, q_Y, q_N\}$.
- $\delta: Q \setminus \{q_Y, q_N\} \times \Gamma \longrightarrow Q \times \Gamma \times \{-1, 0, 1\}$ een transitiefunctie die bepaalt wat er gebeurt als in een bepaalde toestand een bepaald karakter wordt gelezen.

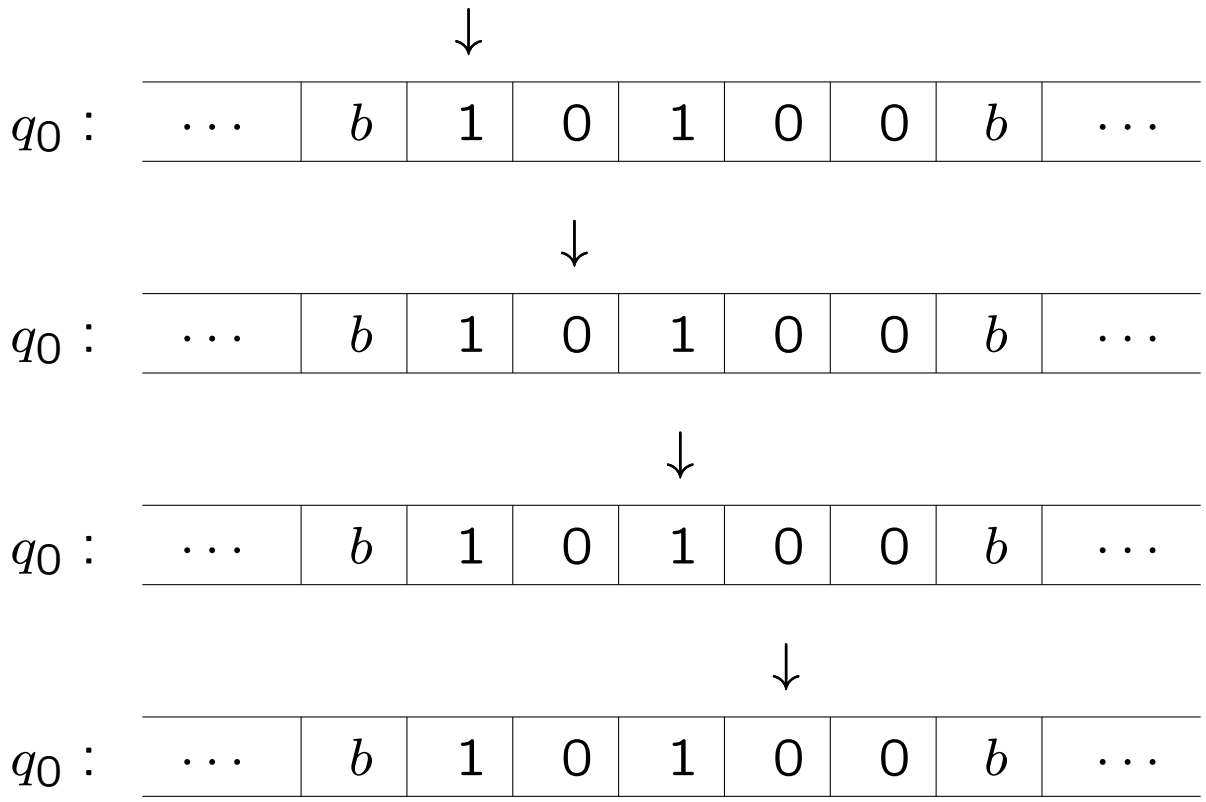
In de begintoestand staat de invoerstring x op plek 1 t/m $|x|$ en de rest van de tape is blanco. Het programma start in toestand q_0 met de lees- en schrijfkop op positie 1 en stopt als de toestand q_y (yes) of q_N (no) bereikt wordt. (En "deterministisch" betekent: δ is een functie.)

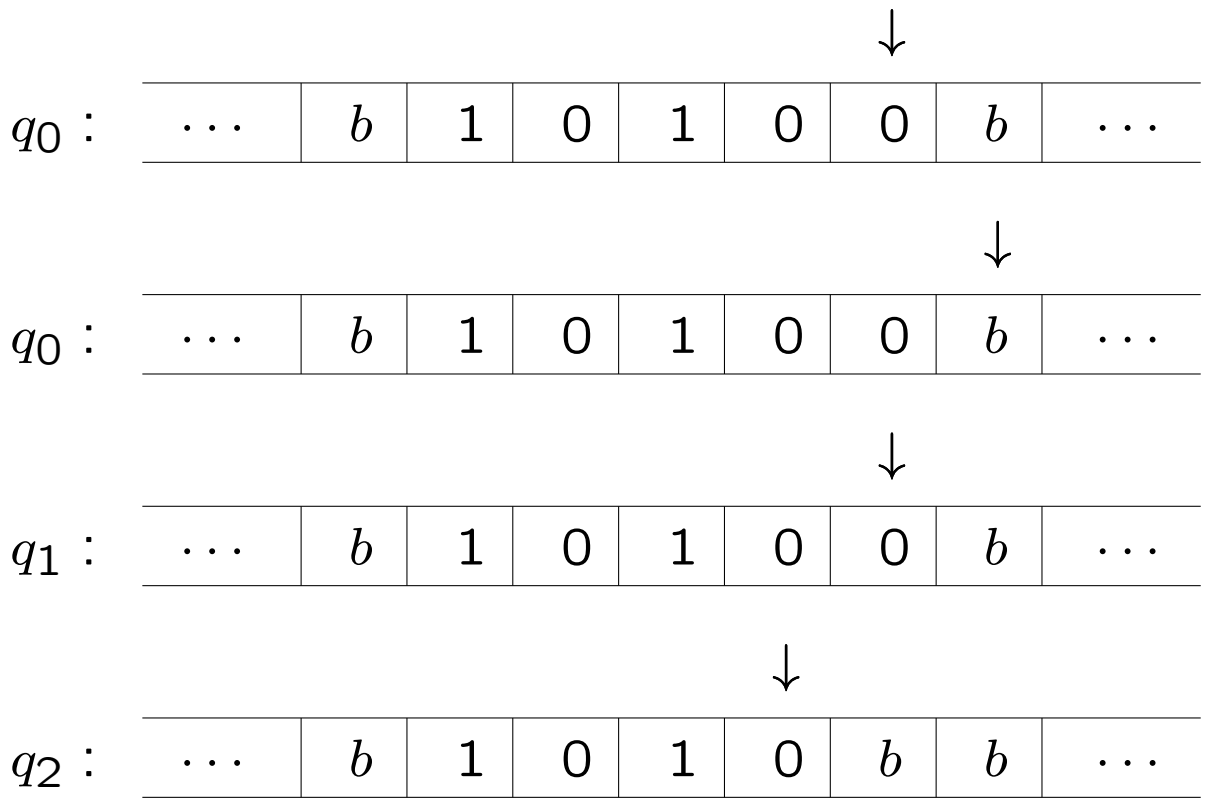
$$\Gamma = \{0, 1, b\}, \Sigma = \{0, 1\}$$

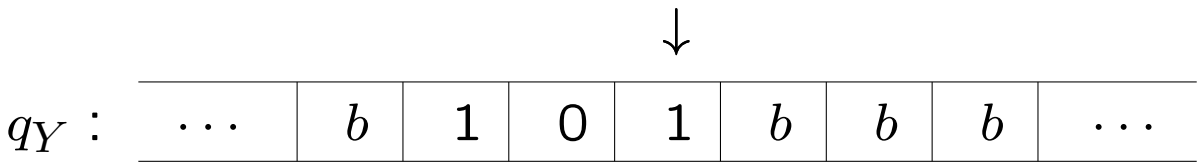
$$Q = \{q_0, q_1, q_2, q_3, q_Y, q_N\}$$

$q \backslash s$	0	1	b
q_0	$(q_0, 0, +1)$	$(q_0, 1, +1)$	$(q_1, b, -1)$
q_1	$(q_2, b, -1)$	$(q_3, b, -1)$	$(q_N, b, -1)$
q_2	$(q_Y, b, -1)$	$(q_N, b, -1)$	$(q_N, b, -1)$
q_3	$(q_N, b, -1)$	$(q_N, b, -1)$	$(q_N, b, -1)$

$$\delta(q, s)$$


q_0 : ⋯ b 1 0 1 0 0 b ⋯
q_0 : ⋯ b 1 0 1 0 0 b ⋯
q_0 : ⋯ b 1 0 1 0 0 b ⋯
q_0 : ⋯ b 1 0 1 0 0 b ⋯


q_0 : ⋯ b 1 0 1 0 0 b ⋯
q_0 : ⋯ b 1 0 1 0 0 b ⋯
q_1 : ⋯ b 1 0 1 0 0 b ⋯
q_2 : ⋯ b 1 0 1 0 b b ⋯

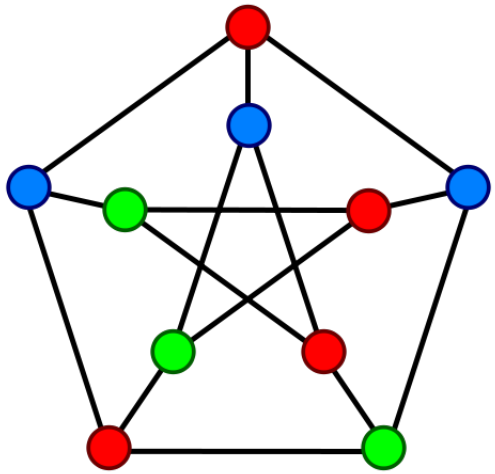


Vraag: welk beslissingsprobleem lost dit programma op?

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



- 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.

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

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

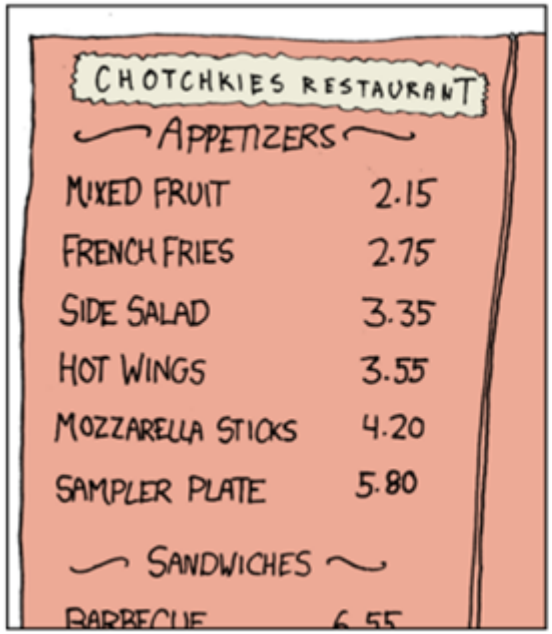
waar. Deze ϕ is dus een ja-instantie voor SAT.

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 $\Sigma_{s \in S'} s = t$?

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

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


CHOTCHKIES RESTAURANT
APPETIZERS
MIXED FRUIT 2.15
FRENCH FRIES 2.75
SIDE SALAD 3.35
HOT WINGS 3.55
MOZZARELLA STICKS 4.20
SAMPLER PLATE 5.80
SANDWICHES
BARBECUE 6.55

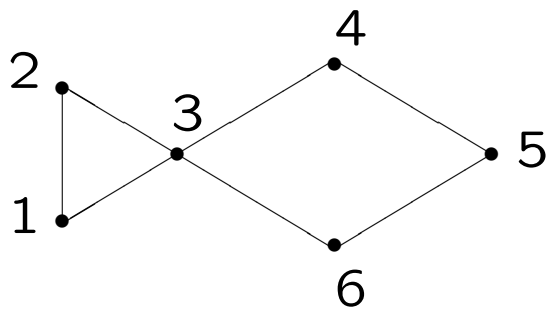

WE'D LIKE EXACTLY $15.05 WORTH OF APPETIZERS, PLEASE.
...EXACTLY? UHH...
HERE, THESE PAPERS ON THE KNAPSACK PROBLEM MIGHT HELP YOU OUT.
LISTEN, I HAVE SIX OTHER TABLES TO GET TO—
—AS FAST AS POSSIBLE, OF COURSE. WANT SOMETHING ON TRAVELING SALESMAN?

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.



Nevenstaande graaf is een nee-instantie voor HC.

Handelsreizigersprobleem of Travelling Salesperson Problem

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

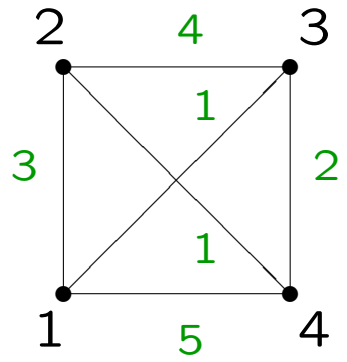
Gegeven een volledige*, ongerichte graaf $\mathcal{G} = (V, E)$ met gewichten op de takken, en een geheel getal $k \geq 0$. Bestaat er in $\mathcal{G}$ een Hamiltonkring met totaalgewicht $\leq k$?

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

Voorbeeld.

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

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



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

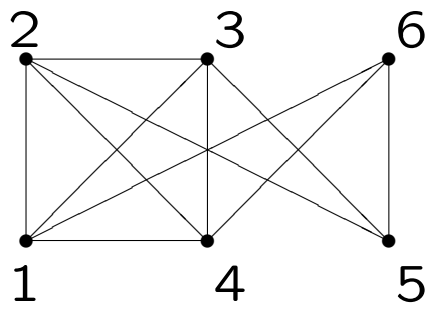
Gegeven een ongerichte graaf $\mathcal{G} = (V, E)$. Geef een kliek met zo veel mogelijk knopen (een maximale kliek).

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?

Opmerking.
Het is equivalent om te vragen naar een kliek ter grootte gelijk aan k. Immers elke deelverzameling van een kliek is 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: $\{1, 2, 3, 4\}$.



Een kleuring van (de knopen van) een ongerichte graaf $\mathcal{G} = (V, E)$ is een afbeelding $c: V \to S$, 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

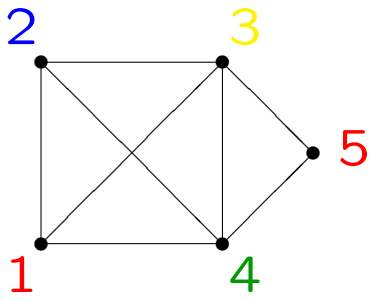
Gegeven een ongerichte graaf $\mathcal{G} = (V, E)$. Vind een kleuring van $\mathcal{G}$ met zo weinig mogelijk kleuren.

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) ?

Opmerking.
Het is equivalent om te vragen naar een kleuring met precies k kleuren die overigens niet allemaal gebruikt hoeven te worden.

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



1. Voor alle zes voorbeeldproblemen is eenvoudig een exponentieel algoritme op te schrijven.

2. In alle gevallen kan in polynomiale tijd gecontroleerd worden of een kandidaatoplossing een echte oplossing (= "dat wat je zoekt") is.

3. Voor al deze problemen geldt: het lijkt extreem moeilijk (exponentieel) 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.

5. Voor ja-instanties bestaat er een zogenaamd certificaat (in de voorbeelden steeds de gezochte oplossing d.w.z. dat wat de invoer zou moeten bezitten) dat gebruikt kan worden om te laten zien dat het antwoord inderdaad "ja" is.

6. Bovendien is dit certificaat kort (polynomiaal) en kan het verifiëren/controleren ervan in polynomiale tijd.

7. Eigenschap 2 t/m 6: de genoemde problemen zitten in $\mathcal{NP}$. Later wordt alles wat formeler gemaakt.

8. Ja-instanties zijn eenvoudig te verifiëren met de juiste hint (certificaat). Hoe zit het met nee-instanties?

- Volgende college:
  dinsdag 16 april, 11.00 – 12.45, zaal 174

- Eerstvolgende werkcollege:
  dinsdag 9 april, 13.30 – 15.15, zaal 174
  Opgaven 38, 39, 40, 41

- Derde huiswerkopgave (van de vier):

  * deadline: dinsdag 23 april; LaTeX; print → college

  * www.liacs.leidenuniv.nl/~graafjmde/COMP/