Complexiteit — college 10

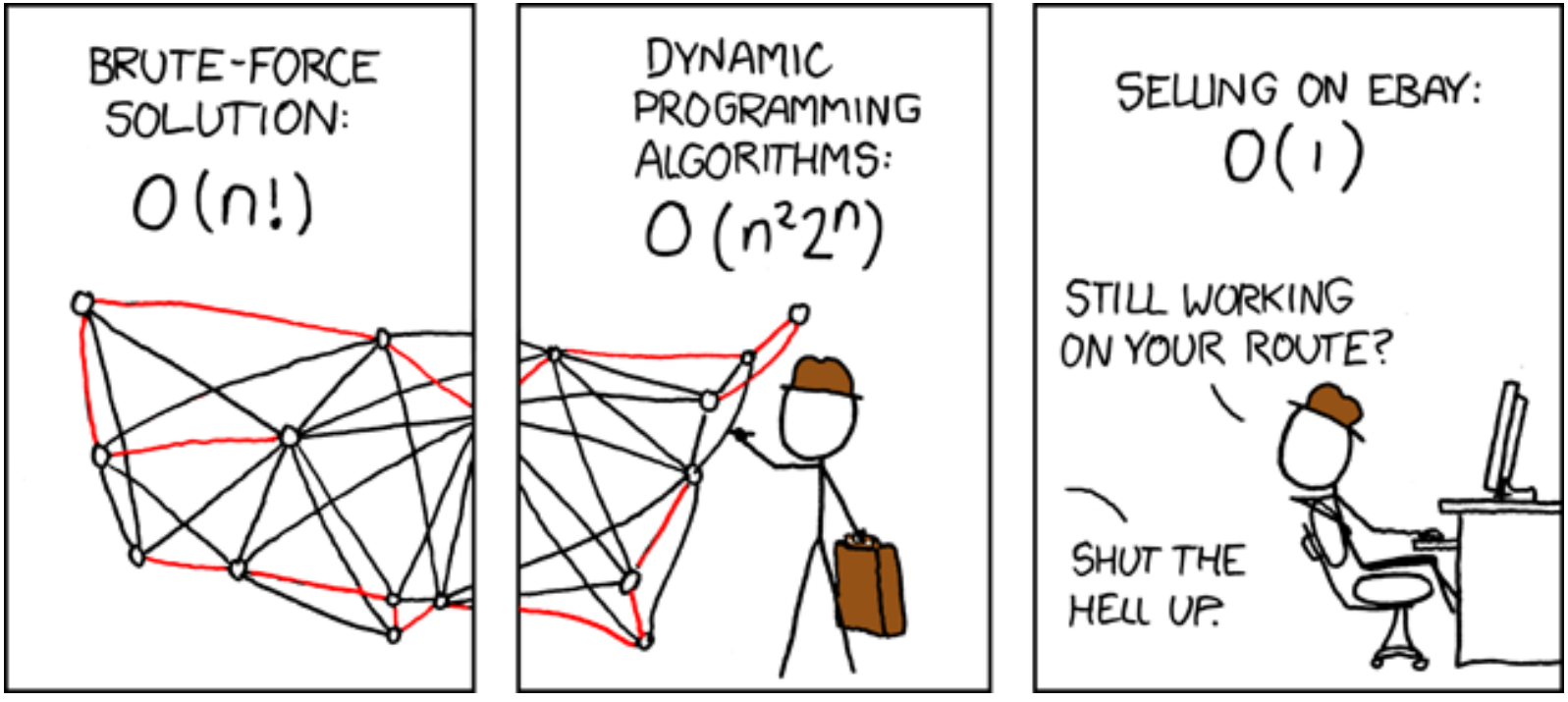
Complexiteitsklassen

Mark van den Bergh

25 April 2025

Universiteit Leiden

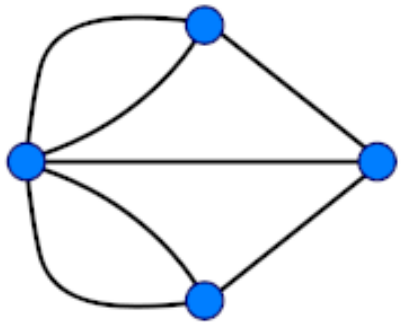
Complexiteit van algoritmen



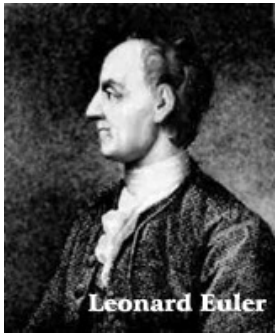
Makkelijk, moeilijk, of onmogelijk

- Computability: sommige problemen zijn onmogelijk (=onbeslisbaar) met een computer (eigenlijk: Turing machine) op te lossen.
  voorbeeld: Halting problem
- Sommige problemen zijn met een efficiënt (=polynomiaal) algoritme op te lossen. Deze problemen zijn kennelijk makkelijk.
  voorbeeld: Sorteren in $O(n \lg n)$
- Voor sommige problemen kennen we geen efficiënt algoritme. Zijn deze problemen "dus" moeilijk? Kunnen we dit formeler maken?

Koningsberger bruggenprobleem



Kun je een wandeling door de stad maken waarbij je elke brug precies één keer beloopt en je weer terugkeert in het beginpunt?



Leonard Euler, 1736*

*en.wikipedia.org/wiki/List_of_things_named_after_Leonhard_Euler

Definities

Gegeven een samenhangende, ongerichte graaf $G = (V, E)$.

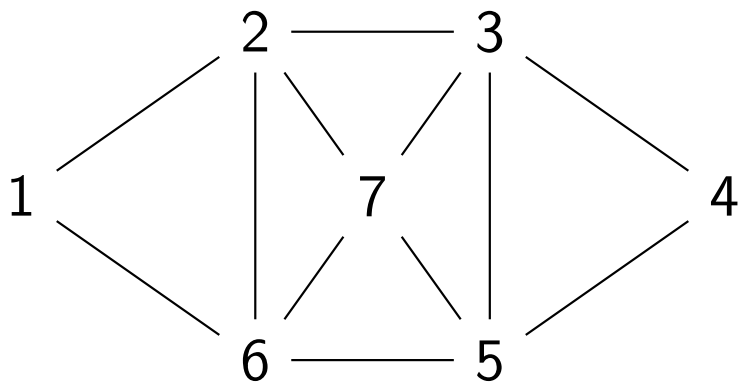
- een pad van u naar v is een rij knopen waarvoor geldt dat tussen elk tweetal opeenvolgende knopen uit die rij een tak (lijn) zit.
- de lengte van een pad is het aantal takken op dat pad is het aantal knopen $- 1$.
- een kring (FoCS: circuit) in een ongerichte graaf is een pad dat begint en eindigt in dezelfde knoop (een *gesloten* pad dus) en dat geen enkele tak meer dan één keer bevat. Een kring bestaat dus uit allemaal verschillende takken.
- een gesloten pad dat geen enkele *knoop* meer dan één keer bevat (FoCS: cykel) is een speciaal geval van een kring. Zo'n pad bestaat dus uit allemaal verschillende knopen.

Eulerkringen

Gegeven een samenhangende, ongerichte graaf $G = (V, E)$.

Definitie: een kring in G die alle takken van G bevat heet een Eulerkring. Deze bevat dus *elke* tak precies één keer.

Voorbeeld:



Voor deze graaf is 1 2 3 4 5 3 7 5 6 7 2 6 1 een Eulerkring.

Eulerkringprobleem

Eulerkringprobleem. Gegeven een samenhangende ongerichte graaf $G = (V, E)$. Heeft G een Eulerkring?

Stelling
Een samenhangende ongerichte graaf heeft een Eulerkring $\Leftrightarrow$ de graad (d.w.z. het aantal buren) van iedere knoop is even.

Het is dus heel gemakkelijk (d.w.z. polynomiaal) na te gaan of een graaf een Eulerkring heeft: $O(|E|)$.

Het bewijs van "$\Leftarrow$" is constructief: het geeft je meteen een $O(|E| + |V|)$ algoritme om zo'n Eulerkring te vinden.

Bewijs "⇒"

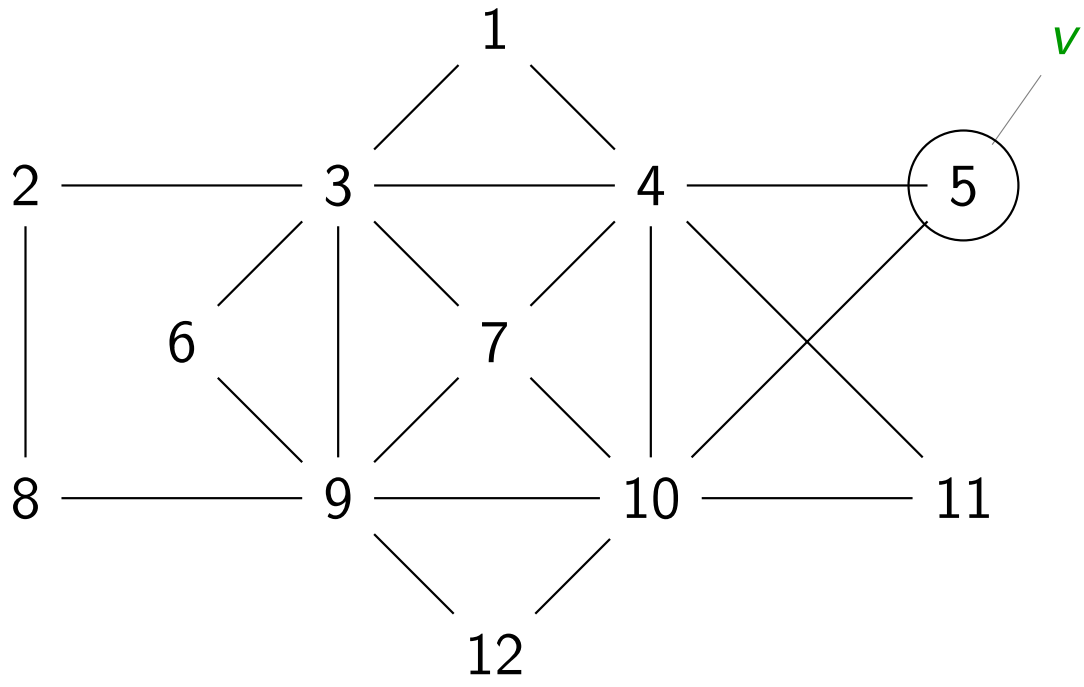
Als een graaf een Eulerkring heeft, wordt daarin elke knoop even vaak betreden als verlaten. Een Eulerkring bevat elke tak precies één keer, dus elke knoop heeft een even aantal takken (en dus aantal buren, en dus graad).

Constructie "$\Leftarrow$"

Algoritme voor constructie Eulerkring (Hierholzer, 1873):

```
1  controleer de samenhangendheid en controleer of elke knoop
     even graad heeft;
2  if controle positief then
3  |   kies een knoop v;
4  |   construeer een kring C (van v naar v);
   |   // gewoon lopen en steeds andere, ongebruikte
   |      takken bewandelen tot je terug bent in v
5  |   while er nog onbewandelde takken zijn do
6  |   |   zoek/onthoud een knoop w van C die nog
   |   |     onbewandelde takken heeft;
7  |   |   construeer een kring (van w naar w) uit ongebruikte
   |   |     takken;
8  |   |   voeg deze kring in in C op de plek van w;
9  |   od
```

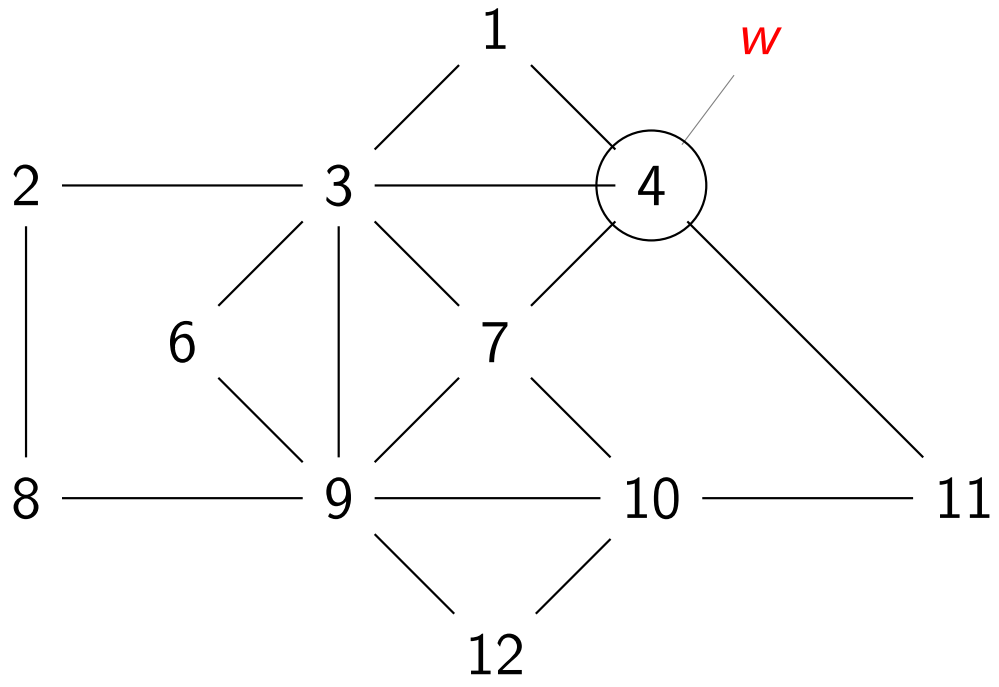
Opbouw Eulerkring



Kring vanuit v: 5, 4, 10, 5

C = 5, 4, 10, 5

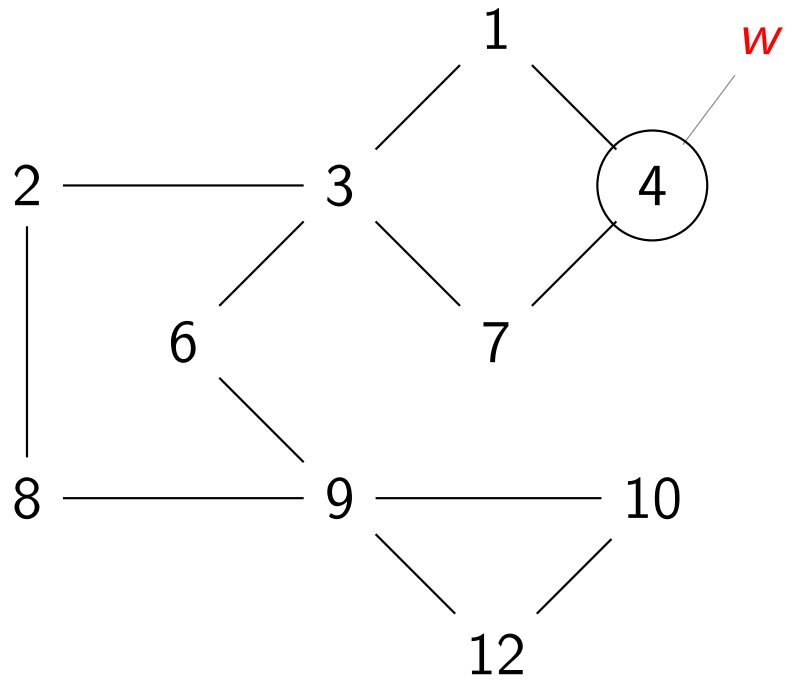
Opbouw Eulerkring



Kring vanuit w: 4, 11, 10, 7, 9, 3, 4

C = 5, 4, 11, 10, 7, 9, 3, 4, 10, 5

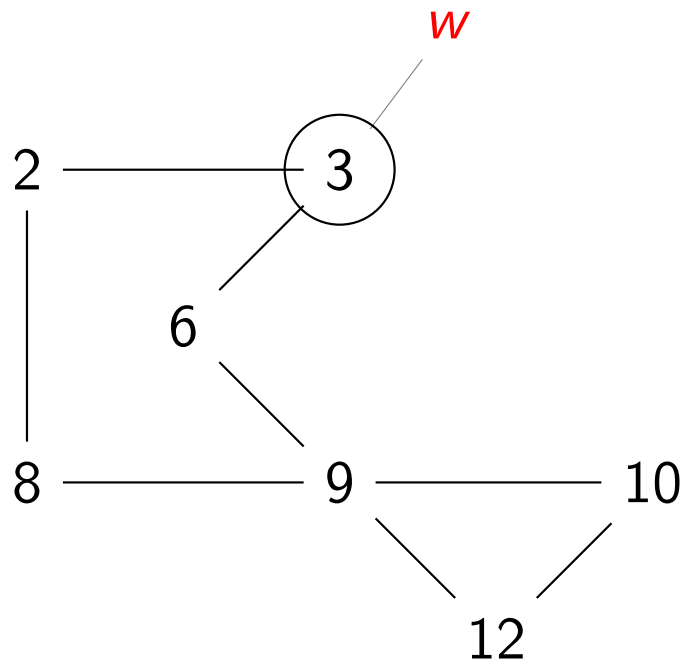
Opbouw Eulerkring



Kring vanuit w: 4, 1, 3, 7, 4

C = 5, 4, 1, 3, 7, 4, 11, 10, 7, 9, 3, 4, 10, 5

Opbouw Eulerkring



Kring vanuit w: 3, 2, 8, 9, 10, 12, 9, 6, 3
(of eerst 3, 2, 8, 9, 6, 3 en dan 9, 10, 12, 9)

C = 5, 4, 1, 3, 2, 8, 9, 10, 12, 9, 6, 3, 7, 4, 11, 10, 7, 9, 3, 4, 10, 5

Correctheid

- "gewoon lopen en steeds andere, ongebruikte takken bewandelen tot je terug bent in v" $\rightarrow$ dit gaat altijd goed, want de graad van elke knoop is even (en de graaf is eindig)
- "zoek/onthoud een knoop w van C die nog onbewandelde takken heeft" $\rightarrow$ dit gaat altijd goed, want G is samenhangend

Complexiteit

Complexiteit van dit algoritme: $O(|V| + |E|)$

Stap 1: samenhangendheid controleren met behulp van DFS en tegelijk de graden bepalen: $O(|V| + |E|)$.

Stap 4 t/m 8: kan in $O(|V| + |E|)$ stappen

- gebruik een kopie van G ($O(|V| + |E|)$ om die te maken).
- haal tijdens de constructie van C een tak weg (uit de kopie) zodra die is gebruikt.
- houd de kring C bij als enkelverbonden lijst, met een pointer naar de eerste knoop die nog onbewandelde takken heeft.

Zie Opgave 50 voor meer details.

Toepassing (DNA)

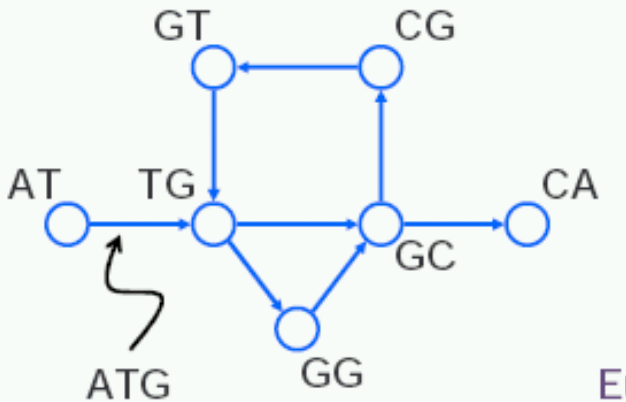
Uit: Molecular Computational Biology (oud mastercollege)

SBH example

we can do better with same problem:

$\ell = 3$

{ ATG, TGG, TGC, GTG, GGC, GCA, GCG, CGT }



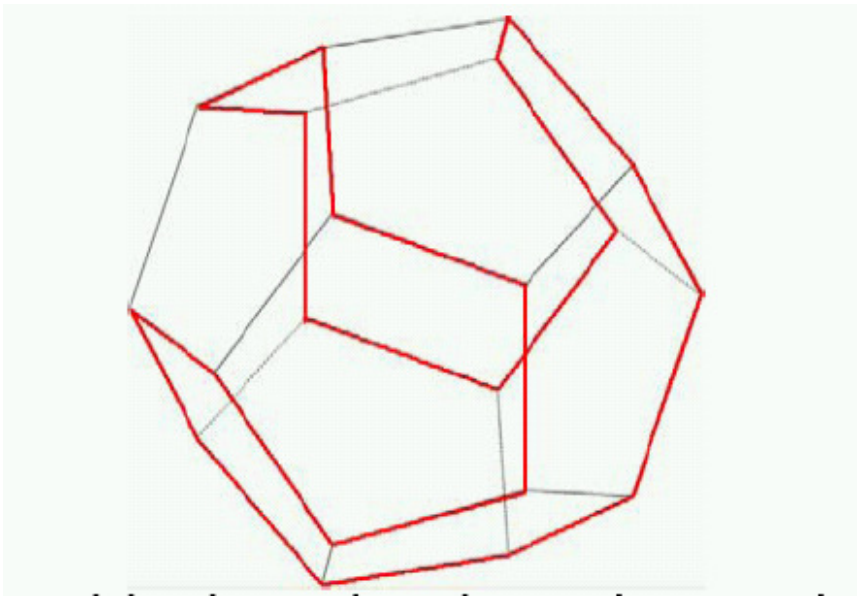
ATGGCGTGCA

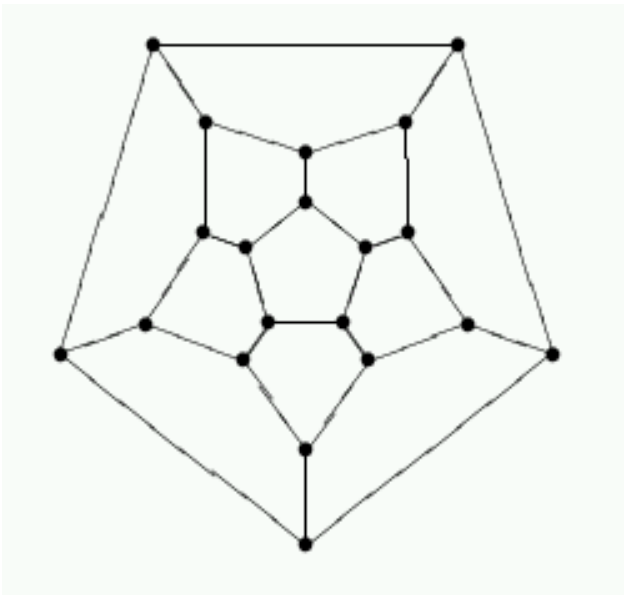
Euler approach: edges
(overlap ℓ-1 = node)
linear ☺

triplet=edge

Hamilton

Kun je een wandeling door de graaf rechtsonder maken waarbij elke *knoop* precies één keer wordt bezocht en je weer eindigt in het startpunt?





Sir William Rowan Hamilton, 1859*

*https://en.wikipedia.org/wiki/List_of_things_named_after_William_Rowan_Hamilton

Hamiltonkringprobleem

Gegeven een ongerichte (of gerichte) graaf $G = (V, E)$.

Definitie: een Hamiltonkring in G is een kring die elke knoop precies één keer bevat.

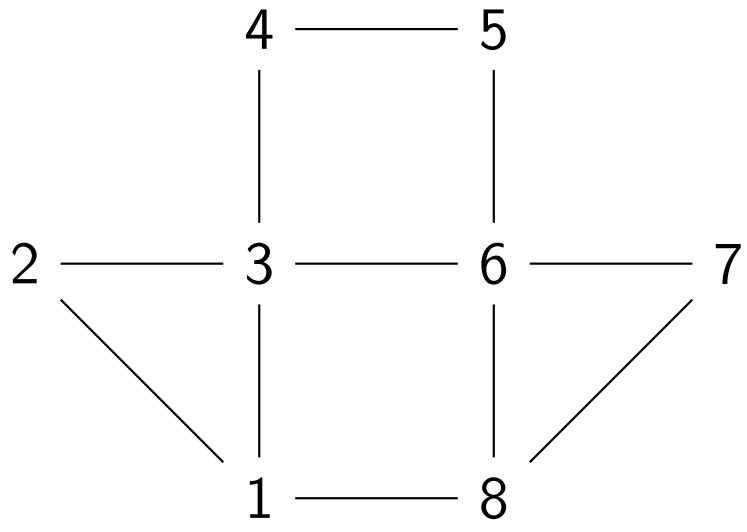
Hamiltonkringprobleem (HC) voor ongerichte grafen*. Gegeven een ongerichte graaf $G = (V, E)$. Heeft G een Hamiltonkring?

Analoog: **Hamiltonpadprobleem**. Vervang 'kring' door 'pad'.

*analoog voor gerichte grafen

Voorbeeld 1

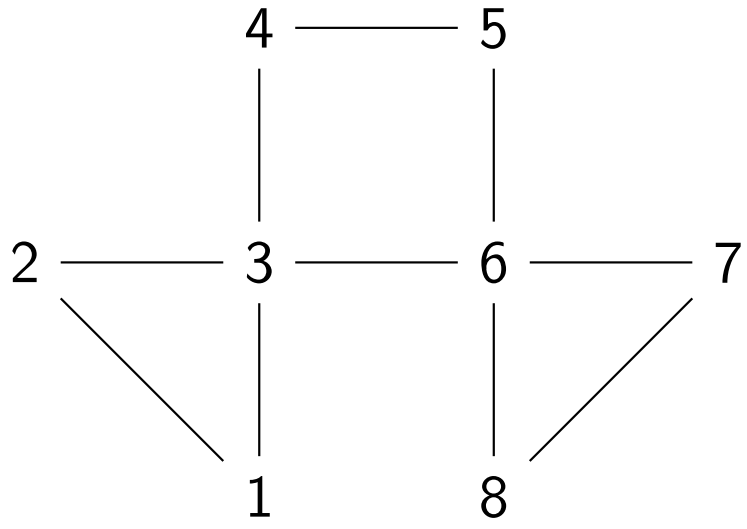
Voorbeeld 1:



Deze graaf heeft een Hamiltonkring: 1, 2, 3, 4, 5, 6, 7, 8, 1.

Voorbeeld 2

Voorbeeld 2:



Deze graaf heeft geen Hamiltonkring, maar wel een Hamiltonpad:
1, 2, 3, 4, 5, 6, 7, 8.

Toepassingen

Vermomde Hamiltonproblemen:

- kan een paard over een $n \times n$-schaakbord een serie sprongen maken zodanig dat het alle n^2 velden precies één keer bezoekt en ten slotte weer terugkeert in zijn beginveld?
- gegeven een groep van n mensen. Van elk tweetal is bekend of ze elkaar wel of niet kennen. Kunnen deze mensen zo aan een ronde tafel worden geplaatst, dat iedereen twee bekenden heeft als buren?
- (Gray-code) er zijn 2^n binaire getallen van n bits. Kunnen deze zo (cyclisch) achter elkaar worden geplaatst dat opeenvolgende getallen slechts op één plaats verschillen?

Moeilijkheid HC

- Een (super)exponentieel algoritme is snel gevonden: genereer alle $n!$ permutaties van knopen van G en controleer voor elke permutatie of het een Hamiltonkring is.
- Er is geen polynomiaal algoritme bekend voor dit probleem; zelfs niet voor het beslissingsprobleem HC, dat alleen vraagt *of* G een Hamiltonkring heeft.
- Er is echter ook niet bewezen dat een exponentieel algoritme *nodig* is om HC op te lossen.

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

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 $\approx 2^{34}$)

Terminologie (1/2)

- Een probleem is een abstracte vraag met een collectie instanties en bijbehorende (mogelijk lege) oplossingsverzamelingen.
  voorbeeld: **Sorteren.** Gegeven een rijtje gehele getallen (instantie), gevraagd het gesorteerde rijtje (bijbehorende oplossing).
- Een instantie van een probleem is een concrete invoer voor het probleem.
  voorbeeld: Een instantie van het probleeem Sorteren is een concreet rijtje getallen $a_1, \ldots, a_n$.

Terminologie (2/2)

- Een probleem heet een beslissingsprobleem als de uitvoer alleen "ja" of "nee" kan zijn.
  voorbeelden: Eulerkring, HC.
- Een instantie van een beslissingsprobleem heet een ja-instantie als de oplossing "ja" is, en anders een nee-instantie.
  voorbeeld: een graaf G is een ja-instantie voor Eulerkring $\Leftrightarrow$ elke knoop in G heeft even graad.

De klasse $\mathcal{P}$

Definitie
Een probleem heet polynomiaal begrensd als er een *deterministische* Turing machine bestaat met tapealfabet $\{0, 1\}$ die voor iedere gegeven instantie van het probleem in een aantal stappen dat hoogstens polynomiaal is in de grootte van de invoer een oplossing genereert.

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

De klasse $\mathcal{P}$ — alternatief

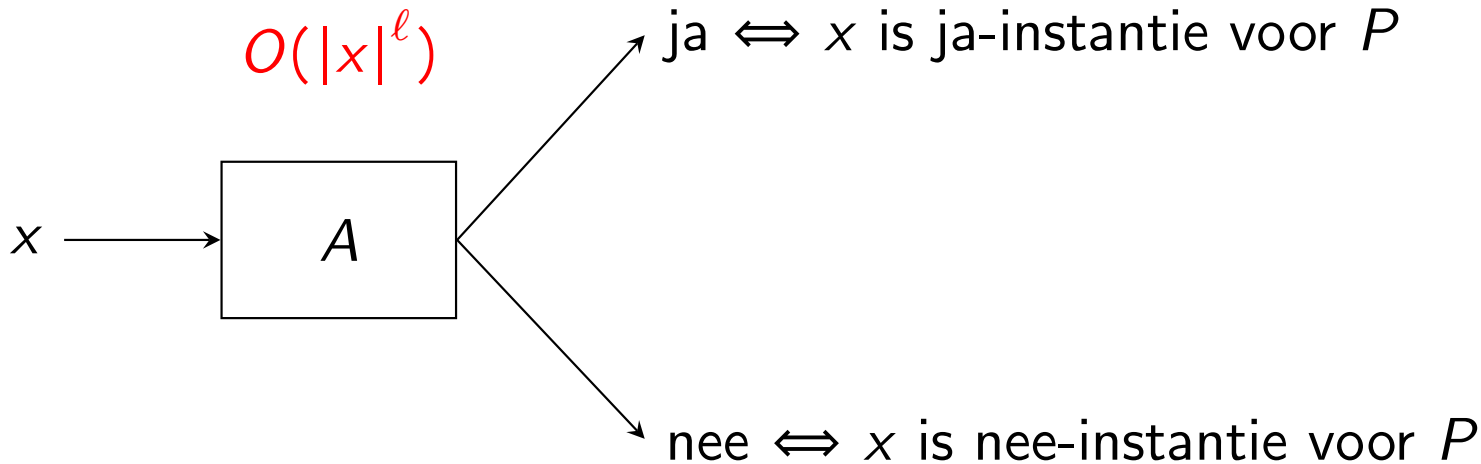
Definitie
Een algoritme heet polynomiaal begrensd als zijn worst case-complexiteit van boven is begrensd door een functie die polynomiaal is in de lengte van de invoer. Dus: worst case is $O(p(n))$ voor een zeker polynoom p, en met n (een maat voor) de lengte van de invoer.
Eenvoudiger geformuleerd: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

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.



Grootte van de invoer

Formeel bestaat de invoer uit een reeks nullen en enen, d.w.z. binaire representatie. De grootte van de invoer is het aantal nullen en enen.

Merk op: een getal n heeft $\Theta(\lg n)$ cijfers in binaire notatie.

In de praktijk kunnen we dus de grootte van een getal, of het aantal getallen, of het aantal knopen of takken in een graaf, of ... nemen als grootte van de invoer.

Hier komen we volgende week nog even op terug.

De klasse $\mathcal{EXP}$

$\mathcal{EXP}$ 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.

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.

Handelbaar $\leftrightarrow \mathcal{P}$?

Geldt 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)

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

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. Deze problemen blijken een klasse op zich te vormen.

De klasse $\mathcal{NP}$

Definitie
Een probleem heet niet-deterministisch polynomiaal begrensd als er een *niet-deterministische* Turing machine bestaat met tapealfabet $\{0, 1\}$ die voor iedere gegeven instantie van het probleem in een aantal stappen dat hoogstens polynomiaal is in de grootte van de invoer een oplossing *kan* genereren.

Definitie
De klasse van beslissingsproblemen die niet-deterministisch polynomiaal begrensd zijn noteren we als $\mathcal{NP}$.

LET OP! $\mathcal{NP}$ staat dus NIET voor *niet-polynomiaal*.

De klasse $\mathcal{NP}$ — alternatief

Definitie
Gegeven een ja-instantie van een beslissingsprobleem. Een oplossing van deze instantie heet een certificaat.
voorbeeld: gegeven een graaf G die een Hamiltonkring bevat (dus een ja-instantie van HC). Een certificaat is een ordening van de knopen die een Hamiltonkring vormt.

Definitie
Een probleem heet niet-deterministisch polynomiaal begrensd als er voor iedere ja-instantie een polynomiaal begrensd certificaat bestaat, en er een polynomiaal begrensd algoritme bestaat dat controleert of een gegeven certificaat inderdaad een oplossing is.

Merk op: het algoritme hoeft de oplossing dus niet te vinden, maar alleen te controleren. Het hoeft tevens niets te kunnen zeggen over nee-instanties.

Voorbeeld: HC $\in \mathcal{NP}$

Invoer: Een graaf $G = (V, E)$, zeg $|V| = n$.

Een certificaat is een ordening S van de n knopen en dus van lengte $\Theta(n)$.

Het verifiëren of een gegeven oplossing inderdaad een certificaat is, kan als volgt:

1. Controleer of er precies n getallen staan: $O(|S|)$.
2. Controleer of elk getal tussen 1 en n is: $O(|S|)$.
3. Controleer dat alle knopen uit S verschillen: $O(|S|^2)$ of sneller.
4. Controleer dat tussen opeenvolgende knopen uit S een tak zit in E: $O(|S| \cdot |E|)$.

Verifieerbaarheid is voldoende

Als een certificaat polynomiaal verifieerbaar is, dan bestaat er ook een niet-deterministische Turing machine die het probleem in polynomiale tijd *kan* oplossen. Deze werkt als volgt:

1. Genereer een willekeurige mogelijke oplossing. Deze is in lengte polynomiaal begrensd, dus dit kan in polynomiale tijd.
2. Ga na of de oplossing inderdaad een certificaat is voor de instantie. Dit kan per aanname ook in polynomiale tijd.
3. Zo ja, accepteer. Zo nee, probeer een volgende willekeurige oplossing.

De klasse $\mathcal{NPC}$

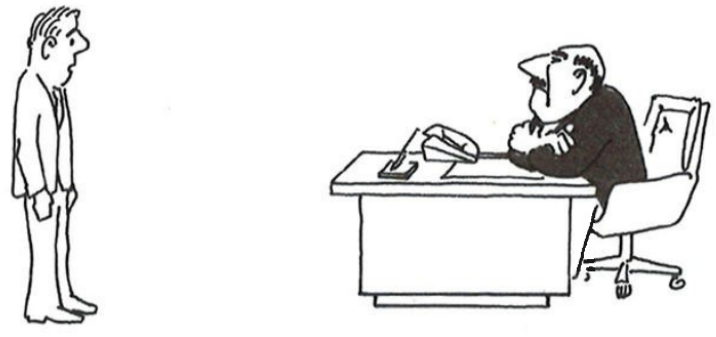
De klasse $\mathcal{NPC}$ van NP-volledige problemen (Engels: NP-complete) bestaat uit alle problemen in $\mathcal{NP}$ die minstens net zo moeilijk zijn als alle andere problemen in $\mathcal{NP}$. Het zijn dus de "moeilijkste" problemen in $\mathcal{NP}$. We maken dit volgende week preciezer. We zullen dan ook zien dat HC $\in \mathcal{NPC}$.

Eigenschappen van $\mathcal{NPC}$

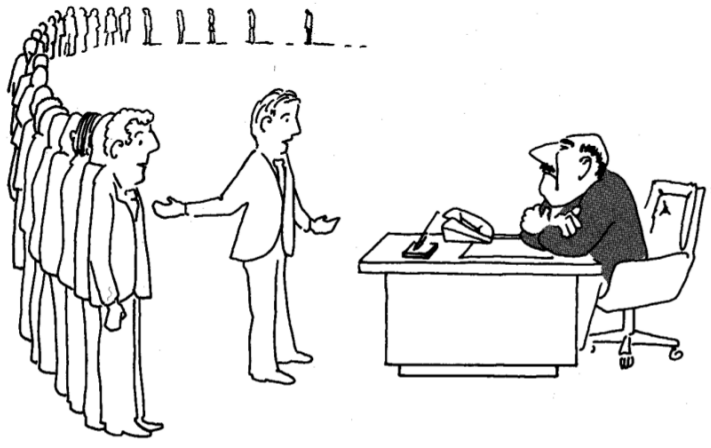
De klasse $\mathcal{NPC}$ heeft enkele interessante eigenschappen, zoals:

- 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.
- als er een polynomiaal algoritme bestaat voor ook maar één NP-volledig probleem, dan is meteen elk NP-volledig probleem in polynomiale tijd oplosbaar.
- omgekeerd: als er van één enkel NP-volledig probleem wordt bewezen dat het onhandelbaar is, dan zijn alle NP-volledige problemen onhandelbaar.

:)



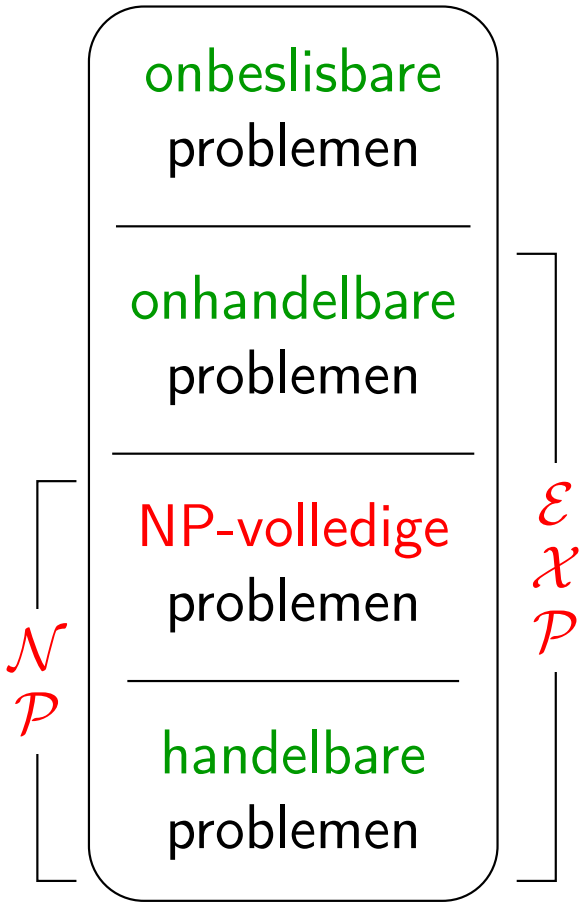
"I can't find an efficient algorithm, I guess I'm just too dumb."



"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

Complexiteitsklassen



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

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

???

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

Het grote vraagstuk

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

Het is logisch dat $\mathcal{P} \subseteq \mathcal{NP}$, maar het is nog onbekend of $\mathcal{NP} \subseteq \mathcal{P}$.

Zie ook: `https://www.claymath.org/millennium/p-vs-np/`

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

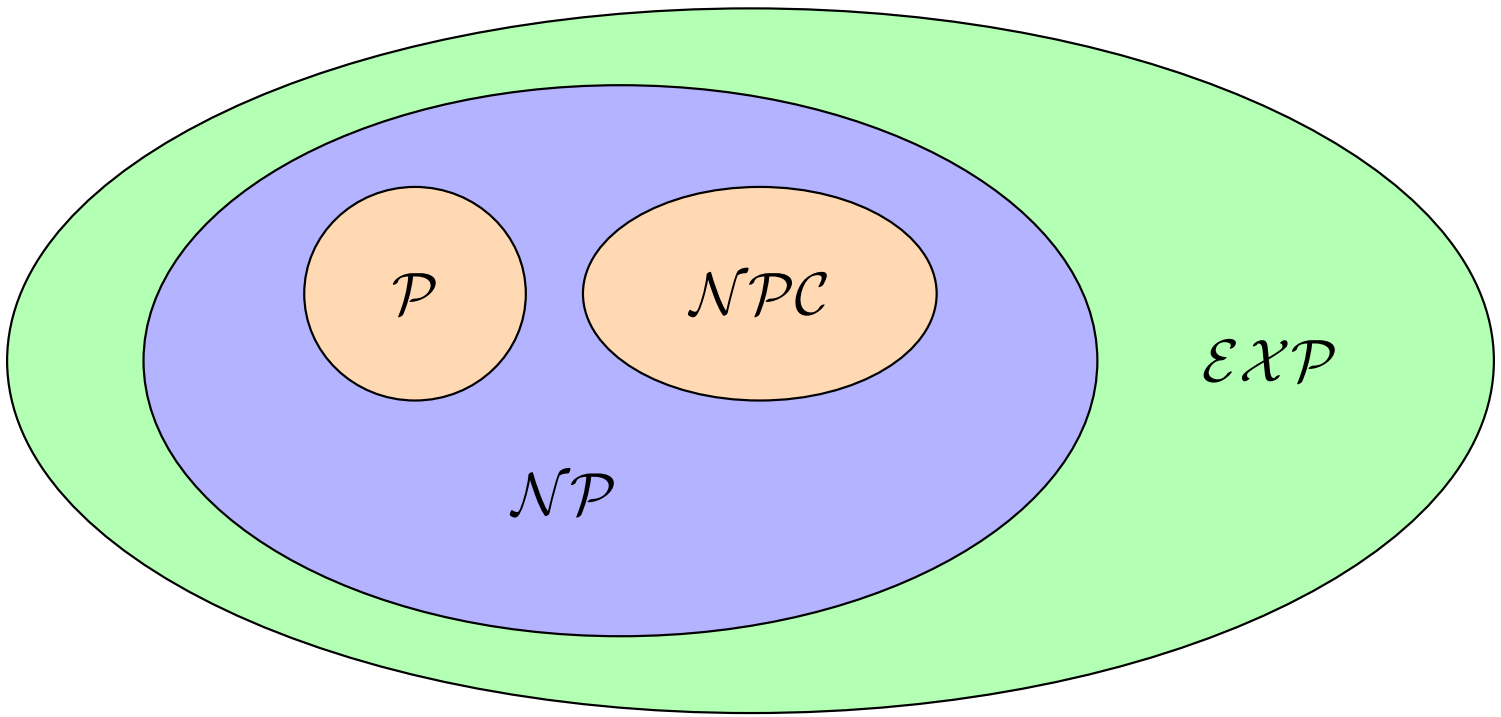
We weten dat $\mathcal{P} \subseteq \mathcal{NP}$ en $\mathcal{NPC} \subseteq \mathcal{NP}$. In theorie kan het zijn dat $\mathcal{P} = \mathcal{NP}$, maar dat is onwaarschijnlijk (zie later).

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.

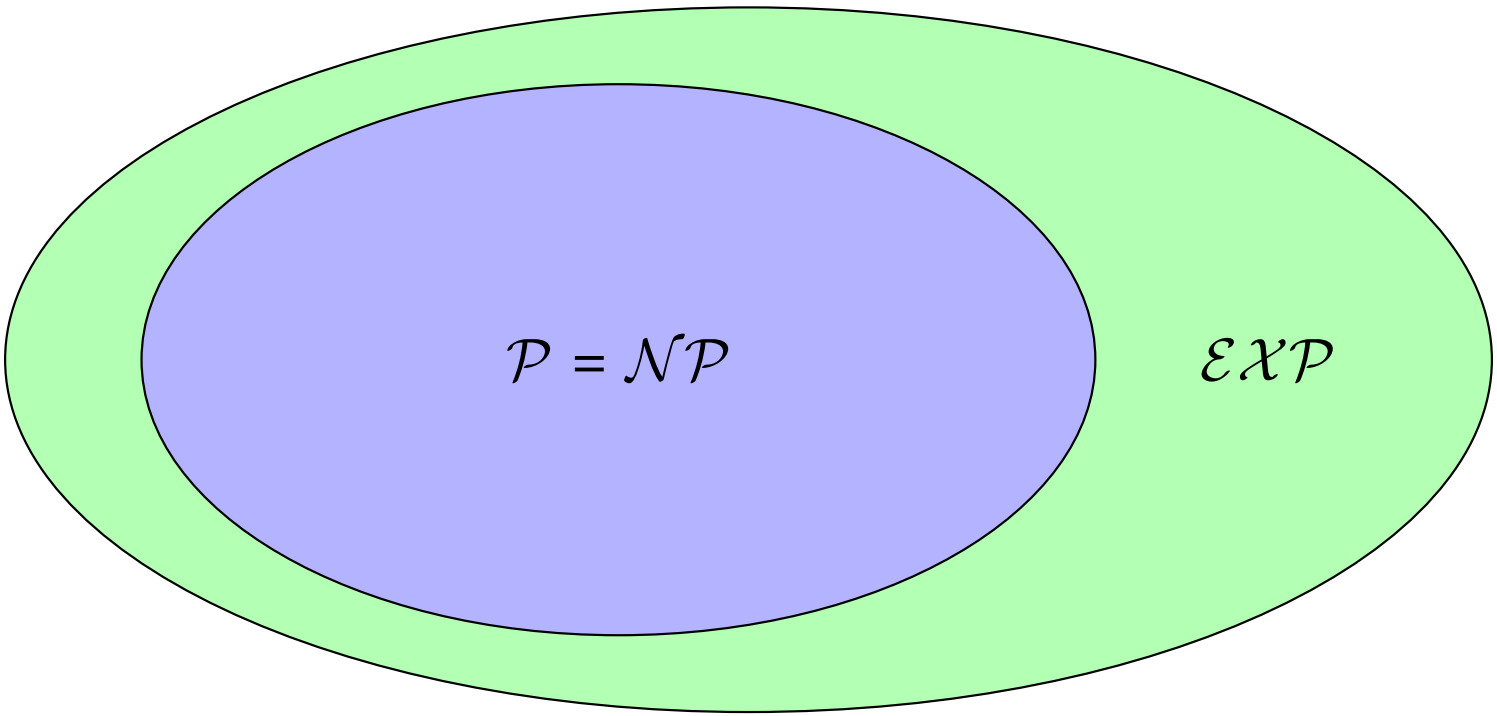
Het bestaan van $\mathcal{NPI}$-problemen is hypothetisch. Ze bestaan onder de aanname dat $\mathcal{P} \neq \mathcal{NP}$, dat weten we zeker, maar we hebben nog geen probleem kunnen identificeren. Er zijn wel enkele veelbelovende kandidaten, zoals het graafisomorfismeprobleem*.

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

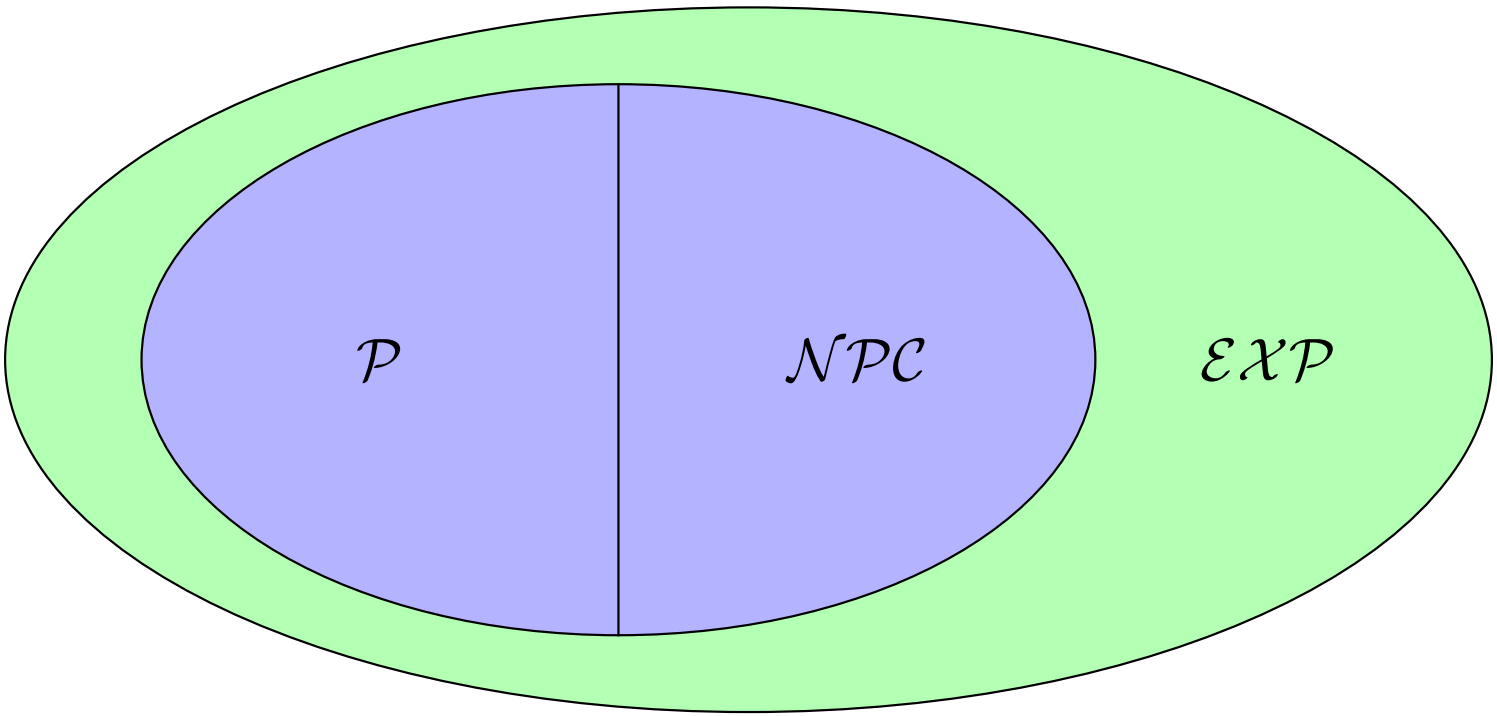
De meest waarschijnlijke relatie



Mogelijk, maar onwaarschijnlijk



Elegant, maar onmogelijk



Het belangrijkste van vandaag

- 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

(Werk)college

Volgende college: vrijdag 2 mei, 11u00 – 12u45, BW.0.40

Werkcollege: vandaag, 13u15 – 15u00, BW.0.29 en BW.0.30
Opgaven: 49, 55, 61, 58ac, 51

Huiswerk: huiswerk 3 staat online, uiterlijk inleveren komende maandag 28 april

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