Complexiteit — college 12

Reducties en SAT

Mark van den Bergh

9 mei 2025

Universiteit Leiden

Vorige keer

- 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

Opgave

Gegeven als invoer een array van n gehele getallen. Welke algoritmes zijn polynomiaal begrensd in de grootte van de invoer?

1. Bepalen van het maximum (door één keer door het array heen te lopen).
2. Bepalen welke paren getallen onderling ondeelbaar (copriem) zijn (door van elk paar getallen de delers te vergelijken).
3. De som van de getallen bepalen (door één keer door het array te lopen).

Antwoord

Algoritmes 1 en 3 zijn polynomiaal begrensd in het aantal getallen. Het vergelijken of optellen van getallen in het array duurt wellicht langer als de getallen groter zijn, maar wel polynomiaal in de lengte van de invoer (grootte van het getal).

Algoritme 2 is niet polynomiaal begrensd in het aantal getallen, omdat een groter getal hier voor exponentieel meer stappen zorgt.

Vandaag

- $\mathcal{NPC}$ en reducties: meer voorbeelden
- Satisfiability als oer-probleem

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

TSP $\in \mathcal{NPC}$

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 $\leq k$?

Bewijs: TSP $\in \mathcal{NPC}$ m.b.v. een reductie vanaf HC.

TSP $\in \mathcal{NP}$

Een polynomiaal begrensd niet-deterministisch algoritme A voor TSP:

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 met gewicht $\leq k$:

   (1) controleer of s een Hamiltonkring voorstelt (zie vorige week: HC $\in \mathcal{NP}$)

   (2) controleer of het totale gewicht van de Hamiltonkring (dat stelt s voor als is voldaan aan (1)–(3)) $\leq k$ is: $O(|s| \times |x|)$

   Als beide tests positief zijn wordt True geretourneerd. Anders wordt False teruggegeven (of ∞ loop).

TSP $\in \mathcal{NP}$

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

Voor het hierboven beschreven algoritme geldt:

- Het antwoord van A op invoer $x = (G, k)$ is "ja" $\iff$ er bestaat een string s waarvoor fase 2 True geeft $\iff$ er bestaat een string s die een Hamiltonkring voorstelt met gewicht $\leq k \iff$ G heeft een Hamiltonkring met gewicht $\leq k \iff$ x is een ja-instantie voor TSP.
- Het algoritme is polynomiaal, want voor een ja-instantie stelt s een (goede) Hamiltonkring voor, dus dan is $|s| \in O(|V|) \subseteq O(|x|)$ en derhalve is fase 2 dan $O(|x|^2)$.

HC $\leq_P$ TSP

Opgave 65:
Beeld een graaf $G = (V, E)$ af op de volledige graaf $G' = (V, E')$, met $E' = \{(i, j) : i, j \in V\}$ en definieer de gewichten op de takken van G' als:

$$c_{ij} = \begin{cases} 0 & \text{als } (i,j) \in E, \\ 1 & \text{als } (i,j) \notin E. \end{cases}$$

Neem $k = 0$.

Nog één keer: invoer

Terzijde: het maakt niet uit voor de polynomialiteit van het algoritme welke representatie is gekozen voor de invoer.

- Neem aan dat $V = \{1, 2, \ldots, n\}$. We nemen nu voor G de adjacency-list-representatie, en voor k de binaire of decimale representatie (bijvoorbeeld). Dan is zoeken van een tak met bijbehorend gewicht $O(|V|) \subseteq O(|x|)$.
- Hetzelfde als hierboven, maar nu nemen we voor G de adjacency-matrix-representatie. Dan is zoeken van een tak met bijbehorend gewicht $O(1)$.
- (denkend aan een Turingmachine) Representeer (G, k) als een rij van knopen, gevolgd door de takken met gewichten, gevolgd door k, alles binair of decimaal. Dan is zoeken van een tak met bijbehorend gewicht $O(|x|)$.

SUM en Partitie

Subset Sum (SUM)

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

Partitie*

Gegeven een (eindig) rijtje getallen $(a_1, \ldots, a_n) \in \mathbb{N}^n$ (m.a.w., $a_i \in \mathbb{N}$ voor alle $1 \leq i \leq n$). Zijn de getallen op te delen in twee deelrijtjes zodat de som van beide deelrijtjes hetzelfde is? M.a.w., bestaat er een deelverzameling $I \subseteq \{1, \ldots, n\}$ zodat $\sum_{i \in I} a_i = \sum_{i \notin I} a_i$?

Voorbeeld (Partitie): het rijtje $(3, 1, 1, 2, 2, 1)$ is op te delen in deelrijtjes $(3, 2)$ en $(1, 1, 2, 1)$, die beide sommeren tot 5.

*hertentamen 2023

SUM $\leq_P$ Partitie

Transformatie T

Gegeven een instantie (S, t) van SUM.* Neem alle elementen van S en voeg er twee aan toe: de som van alle elementen in S, en $2t$.

$$T((S,t)) = (s_1, \ldots, s_n, t_1, t_2), \text{ waar } t_1 = \sum_{s \in S} s \text{ en } t_2 = t$$

Voorbeeld: als $t = 16$ en $S = \{3, 1, 4, 5, 9, 2, 6\}$, dan $t_1 = 30$ en $t_2 = 32$, dus $T((S,t)) = \{3, 1, 4, 5, 9, 2, 6, 30, 32\}$.

*$t \in \mathbb{N}$, $S = \{s_1, \ldots, s_n\} \subset \mathbb{N}$

SUM $\leq_P$ Partitie

Polynomialiteit T*

$T((S,t))$ voegt twee getallen toe aan S: $t_1 = \sum_{s \in S} s$ en $t_2 = 2t$.

- $|t_1| \in O(|S|)$
- $|t_2| \in O(|t|)$

Dus:

$$T((S,t)) \in O(|S| + |t|) = O(|(S,t)|)$$

*hertentamen 2023: aangenomen, niet gevraagd

SUM $\leq_P$ Partitie

Correctheid $\Rightarrow$

Aan te tonen: als (S, t) een ja-instantie is van SUM, dan is $T((S, t))$ een ja-instantie van Partitie.

Merk op: $s_1 + \ldots + s_n + t_1 + t_2 = 2\sum_{s \in S} s + 2t$, dus beide deelrijtjes voor Partitie moeten sommeren tot $\sum_{s \in S} s + t$.

Laat $S' \subseteq S$ een oplossing zijn voor SUM, d.w.z., $\sum_{s \in S'} s = t$.

Neem dan als deelrijtjes voor Partitie: de elementen uit S' samen met t_1, en de resterende elementen uit S samen met t_2:

- $\sum_{s \in S'} s + t_1 = t + \sum_{s \in S} s$
- $\sum_{s \in S \setminus S'} s + t_2 = \sum_{s \in S} s - t + 2t = \sum_{s \in S} s + t$

SUM $\leq_P$ Partitie

Correctheid $\Leftarrow$

Aan te tonen: als $T((S, t))$ een ja-instantie is van Partitie, dan is (S, t) een ja-instantie van SUM.

$s_1 + \ldots + s_n + t_1 + t_2 = 2\sum_{s \in S} s + 2t$, dus beide deelrijtjes voor Partitie sommeren tot $\sum_{s \in S} s + t$.

Één van de deelrijtjes moet $t_1 = \sum_{s \in S} s$ bevatten. De resterende elementen in dit deelrijtje moeten dus sommeren tot t. $t_2 = 2t$ is hier te groot voor, dus moeten deze elementen allemaal uit S komen. M.a.w.: een deelverzameling van S sommeert tot t.

SAT en 3SAT

SAT

Gegeven een logische formule ϕ in CNF. Bestaat er een waardering die ϕ waar maakt?

Definitie

Een logische formule ϕ staat in CNF als ϕ een conjunctie is van clausules, waarin elke clausule een disjunctie is van literalen.

3SAT

Gegeven een logische formule ϕ in 3-CNF. Bestaat er een waardering die ϕ waar maakt?

Definitie

Een logische formule ϕ staat in 3-CNF als ϕ in CNF staat en elke clausule bestaat uit precies drie (doorgaans verschillende) literalen.

SAT $\leq_P$ 3SAT

Er geldt: SAT $\leq_P$ 3SAT. Om dit aan te tonen moeten we een logische formule ϕ in CNF afbeelden op een logische formule ϕ' in 3-CNF. We gaan er voor het gemak van uit dat de $\ell_1, \ell_2, \ldots, \ell_k$ per clausule al verschillend zijn (kan anders worden bewerkstelligd in $O(|\phi|^2)$). Op clausuleniveau werkt de transformatie T als volgt:

$$\ell_1 \mapsto (\ell_1 \vee \widetilde{\ell_2} \vee \widetilde{\ell_3}) \wedge (\ell_1 \vee \widetilde{\ell_2} \vee \neg\widetilde{\ell_3}) \wedge (\ell_1 \vee \neg\widetilde{\ell_2} \vee \widetilde{\ell_3}) \wedge (\ell_1 \vee \neg\widetilde{\ell_2} \vee \neg\widetilde{\ell_3})$$
$$\ell_1 \vee \ell_2 \mapsto (\ell_1 \vee \ell_2 \vee \widetilde{\ell_3}) \wedge (\ell_1 \vee \ell_2 \vee \neg\widetilde{\ell_3})$$
$$\ell_1 \vee \ell_2 \vee \ell_3 \mapsto \ell_1 \vee \ell_2 \vee \ell_3$$
$$\ell_1 \vee \ell_2 \vee \ell_3 \vee \ell_4 \mapsto (\ell_1 \vee \ell_2 \vee \widetilde{\ell_5}) \wedge (\ell_3 \vee \ell_4 \vee \neg\widetilde{\ell_5})$$
$$\ell_1 \vee \ell_2 \vee \ell_3 \vee \ell_4 \vee \ell_5 \mapsto (\ell_1 \vee \ell_2 \vee \widetilde{\ell_6}) \wedge (\ell_3 \vee \neg\widetilde{\ell_6} \vee \widetilde{\ell_7}) \wedge (\ell_4 \vee \ell_5 \vee \neg\widetilde{\ell_7})$$

SAT $\leq_P$ 3SAT

En in het algemeen voor $k \geq 4$:

$$\ell_1 \vee \ell_2 \vee \ldots \vee \ell_{k-1} \vee \ell_k \mapsto (\ell_1 \vee \ell_2 \vee \widetilde{\ell_{k+1}}) \wedge (\ell_3 \vee \neg\widetilde{\ell_{k+1}} \vee \widetilde{\ell_{k+2}}) \wedge (\ell_4 \vee \neg\widetilde{\ell_{k+2}} \vee \widetilde{\ell_{k+3}}) \wedge \ldots \wedge (\ell_{k-2} \vee \neg\widetilde{\ell_{2k-4}} \vee \widetilde{\ell_{2k-3}}) \wedge (\ell_{k-1} \vee \ell_k \vee \neg\widetilde{\ell_{2k-3}})$$

Hierin zijn $\widetilde{\ell_{k+1}}, \widetilde{\ell_{k+2}}, \ldots, \widetilde{\ell_{2k-3}}$ steeds nieuwe, frisse logische variabelen.

Een clausule met k (verschillende) literalen wordt zo getransformeerd in een conjunctie van $k-2$ clausules met elk 3 verschillende literalen. Het beeld van een conjunctie van clausules definiëren we als een conjunctie van de beelden van de samenstellende clausules:

$$\phi = C_1 \wedge C_2 \wedge \ldots \wedge C_m \mapsto T(C_1) \wedge T(C_2) \wedge \ldots \wedge T(C_m) = T(\phi)$$

SAT $\leq_P$ 3SAT

Voor deze transformatie T geldt:

- de constructie van $T(\phi)$ uit ϕ kan met een polynomiaal begrensd ($O(|\phi|^q)$) algoritme.
- ϕ is een ja-instantie van SAT $\Leftrightarrow$ $T(\phi)$ is een ja-instantie van 3SAT.
- oftewel: er is een waardering die ϕ waarmaakt $\Leftrightarrow$ er is een waardering die $T(\phi)$ waarmaakt.
- conclusie uit de vorige punten: SAT $\leq_P$ 3SAT.

Vragen

We hebben nu: SAT $\leq_P$ 3SAT (*)

Verder is 1SAT: gegeven een logische formule ϕ in 1-CNF. Bestaat er een waardering die ϕ waar maakt? Een logische formule ϕ in 1-CNF heeft de volgende vorm: $\phi = \ell_1 \wedge \ell_2 \wedge \ldots \wedge \ell_n$.

1. Stel dat we weten dat 3SAT $\in \mathcal{NPC}$ en SAT $\in \mathcal{NP}$. Volgt dan uit (*) dat SAT $\in \mathcal{NPC}$?
2. Stel dat we weten dat SAT $\in \mathcal{NPC}$ en 3SAT $\in \mathcal{NP}$. Volgt dan uit (*) dat 3SAT $\in \mathcal{NPC}$?
3. Stel dat we weten dat 3SAT $\in \mathcal{NPC}$. Is 1SAT $\leq_P$ 3SAT?
4. Stel dat we weten dat 3SAT $\in \mathcal{NPC}$. Is 3SAT $\leq_P$ 1SAT?

3SAT en Kliek

3SAT

Gegeven een logische formule ϕ in 3-CNF. Bestaat er een waardering die ϕ waar maakt?

Kliek

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

Definitie

Een kliek in een ongerichte graaf $G = (V, E)$ is een deelverzameling $V' \subseteq V$ zodanig dat voor elk tweetal knopen $u, v \in V'$ $(u \neq v)$ geldt dat $(u, v) \in E$. (Oftewel: V' is volledig.)

3SAT $\leq_P$ Kliek

Er geldt: 3SAT $\leq_P$ Kliek. Om dit aan te tonen moeten we een logische formule in 3-CNF afbeelden op een invoer voor Kliek, dus op een ongerichte graaf en een geheel getal.

Zij ϕ een logische formule in 3-CNF, met zeg m clausules: $\phi = C_1 \wedge C_2 \wedge \ldots \wedge C_m$. Hierin is $C_r = \ell_1^r \vee \ell_2^r \vee \ell_3^r$ $(r = 1, \ldots, m)$ en ℓ_1^r, ℓ_2^r en ℓ_3^r steeds verschillend bij vaste r.

Construeer nu een ongerichte graaf $G_\phi = (V, E)$ als volgt.

Voor elke clausule C_r uit ϕ doen we drie knopen v_1^r, v_2^r en v_3^r in V (deze corresponderen met ℓ_1^r, ℓ_2^r en ℓ_3^r). $\mathcal{G}_\phi$ heeft dus $3m$ knopen.

3SAT $\leq_P$ Kliek

Er komt een tak tussen twee knopen v_i^r en v_j^s als:

- v_i^r en v_j^s in verschillende drietallen zitten (dus $r \neq s$), en
- de bijbehorende ℓ_i^r en ℓ_j^s zó zijn dat $\ell_i^r \neq \neg\ell_j^s$, met andere woorden: ℓ_i^r en ℓ_j^s zijn niet elkaars negatie.

Definieer nu de transformatie T als: $T(\phi) = (G_\phi, m)$. Dan geldt:

- De constructie van $T(\phi)$ uit ϕ kan in $O(|\phi|^q)$ stappen voor zekere $q \geq 0$ (dus polynomiaal).
- Er is een waardering die ϕ waarmaakt $\Leftrightarrow$ G_ϕ heeft een kliek ter grootte m.

3SAT $\leq_P$ Kliek: voorbeeld

Laat $\phi = C_1 \wedge C_2 \wedge C_3$, met $C_1 = x_1 \vee \neg x_2 \vee \neg x_3$,
$C_2 = \neg x_1 \vee x_2 \vee x_3$ en $C_3 = x_1 \vee x_2 \vee x_3$. Hier is dus $m = 3$.

Dan $v_1^1 = x_1$, $v_2^1 = \neg x_2$ en $v_3^1 = \neg x_3$; alle uit clausule C_1, enz.

- een waardering w die ϕ waarmaakt is bijvoorbeeld: $w(x_1) = w(x_2) = \text{False}$ en $w(x_3) = \text{True}$. Een bijbehorende kliek in G_ϕ ter grootte 3 is dan $\{v_2^1, v_3^2, v_3^3\}$*.
- een kliek ter grootte 3 in G_ϕ is bijvoorbeeld $\{v_1^1, v_2^2, v_2^3\}$. Een bijbehorende waardering is $w(x_1) = w(x_2) = w(x_3) = \text{True}$. Deze maakt ϕ waar.

*De bovenindex komt overeen met de clausule. Uit elk drietal (clausule) één knoop (literaal).

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**:

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

SAT is NP-volledig

Vroege jaren '70, Stephen Cook, Richard Karp en Leonid Levin: SAT $\in \mathcal{NPC}$.

The complexity of theorem proving procedures, `https://doi.org/10.1145/800157.805047`.

Stelling

Gegeven een *arbitrair* probleem $P \in \mathcal{NP}$. Dan is P reduceerbaar tot SAT: $P \leq_P$ SAT.

Sindsdien is met behulp van de reductiemethode van veel bekende problemen aangetoond dat ze NP-volledig zijn. Bijvoorbeeld:

$$\text{SAT} \leq_P \text{3SAT} \leq_P \text{Kliek} \leq_P \text{VC}$$
$$\text{3SAT} \leq_P \text{HC2} \leq_P \text{TSP}$$
$$\text{SAT} \leq_P \text{3Kleur} \leq_P \text{4Kleur}$$

Schets bewijs Cook

1. Omdat $P \in \mathcal{NP}$, is er een niet-deterministische Turingmachine M voor P; polynomiaal begrensd op ja-instanties.
2. Deze machine zal voor elke invoer x van P worden gemodelleerd als een logische formule $\phi = T(x)$ in CNF: deze ϕ beschrijft de berekening van M, werkend op x.
3. De formule ϕ is weliswaar lang, maar kan worden geconstrueerd in hooguit $O(|x|^{\ell})$ stappen.
4. Voor een ja-instantie x vergt de executie van M hooguit $c|x|^{k}$ stappen.
5. Een waarmakende waardering voor ϕ correspondeert precies met een executie van M die een "ja" produceert. Dus er is een waardering die ϕ waarmaakt $\Leftrightarrow$ x is een ja-instantie voor P.

Eindige automaat naar SAT

We bekijken hoe een niet-deterministische eindige automaat en diens invoer naar een formule in CNF kan worden omgezet. De omzetting van een Turingmachine is gelijkaardig, maar meer boekhoudwerk.

Zij n het aantal karakters in de invoer. We voegen een "sink" toestand F toe aan de toestandsruimte Q.

Benodigde Boolese variabelen:

Q_i^q : op tijdstip i is de automaat in toestand $q \in Q \cup \{F\}$; $0 \leq i \leq n$

L_i^σ : het i-de karakter in de invoer is σ; $0 \leq i < n$, $\sigma \in \Sigma$

Clausules

- $Q_0^{q_0}$

  op tijdstip 0 begint de berekening in de begintoestand

- $\bigvee_{q \in A} Q_n^q$

  op tijdstip n stopt de berekening in een ja-toestand

- $(\bigvee_{q \in Q} Q_i^q), \qquad (\neg Q_i^p \vee \neg Q_i^q) \qquad \begin{array}{c} 0 \leq i \leq n \\ p, q \in Q, p \neq q \end{array}$

  altijd precies één toestand

- $L_i^{x_i}, \qquad \neg L_i^{\sigma} \qquad \begin{array}{c} 0 \leq i < n \\ \sigma \in \Sigma, \sigma \neq x_i \end{array}$

  invoer = x

Clausules

- $Q_i^p \wedge L_i^\sigma \rightarrow \bigvee_{q \in \delta(q,\sigma)} Q_{i+1}^q$
  $0 \leq i < n, p \in Q, \sigma \in \Sigma$

  elke stap van de automaat verloopt volgens de transitierelatie δ
- $Q_i^q \wedge L_i^\sigma \rightarrow Q_{i+1}^F$
  $0 \leq i < n, q \in Q, \sigma \in \Sigma, \delta(q,\sigma) = \varnothing$

  crash waar nodig
- $Q_i^F \rightarrow Q_{i+1}^F \qquad 0 \leq i < n$

  eenmaal gecrasht blijft gecrasht

Een waarmakende waardering komt zo precies overeen met een echte executie van de niet-deterministische automaat die de invoer accepteert na een polynomiaal (namelijk n) aantal stappen.

SAT in de praktijk

- Hoewel SAT $\in \mathcal{NPC}$, vallen veel instanties in de praktijk nog behoorlijk snel op te lossen $\rightsquigarrow$ SAT-solvers, volgende week meer
- Toepassingen: hardware-, software- en modelverificatie, planning, scheduling, spellen

Het belangrijkste van vandaag

- Voorbeelden van reducties:
  - HC $\leq_P$ TSP
  - SUM $\leq_P$ Partitie
  - SAT $\leq_P$ 3SAT
  - 3SAT $\leq_P$ Kliek
- SAT is het "oer-$\mathcal{NPC}$-probleem"

(Werk)college

Volgende college: vrijdag 16 mei, 11u00 – 12u45, BW.0.40

Werkcollege: vanmiddag, 13u15 – 15u00, BW.0.29, BW.0.30
Opgaven: 65, 66, 67

Huiswerk: deadline huiswerk 4 op maandag 19 mei

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