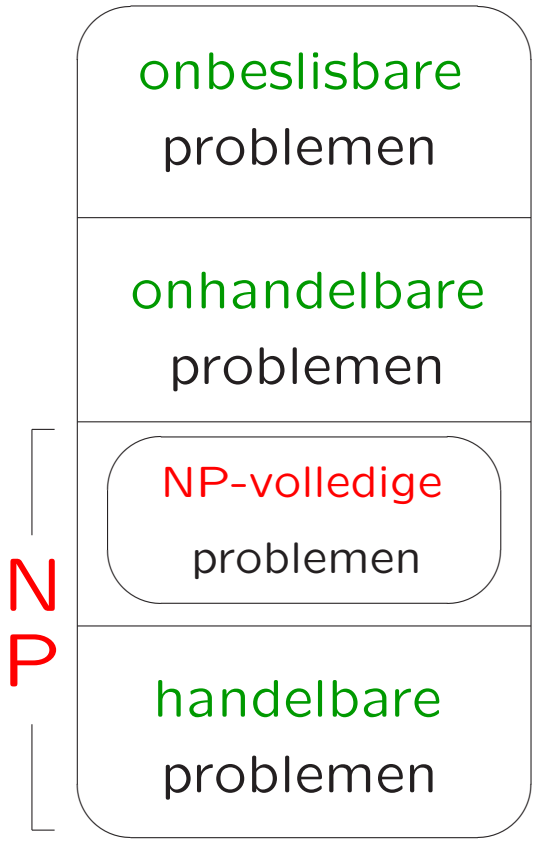
Elfde college complexiteit

8 april 2008

NP-volledigheid II

Indeling problemen

Problemen kunnen als volgt in klassen worden ingedeeld:



Problemen waarvoor geen algoritme mogelijk is (stop-probleem)

Problemen waarvoor geen polynomiaal algoritme mogelijk is

????

Problemen waarvoor een polynomiaal algoritme bestaat: P

De klasse van NP-volledige problemen $\mathcal{NPC}$ (ook wel: NP-complete problemen), 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.
   Omgekeerd: als er van één enkel NP-volledig probleem bewezen wordt dat het onhandelbaar is, dan zijn alle NP-volledige problemen onhandelbaar.

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.

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

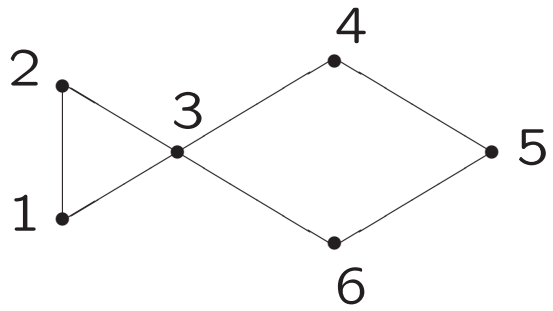
Voorbeeld.
Neem $S = \{1, 4, 16, 64, 256, 1040, 1093, 1284, 1344\}$ en $t = 3754$, dan is dit een ja-instantie voor het probleem. Immers $S' = \{1, 16, 64, 256, 1040, 1093, 1344\}$ voldoet.

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

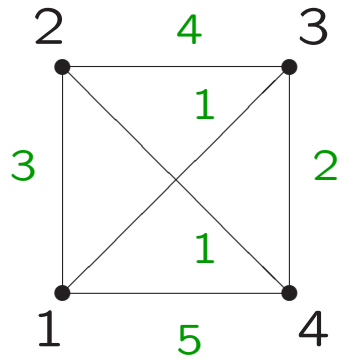
Gegeven een volledige *, ongerichte graaf $\mathcal{G} = (V, E)$ met gewichten op de takken. Geef een Hamiltonkring in $\mathcal{G}$ met minimaal totaalgewicht.

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

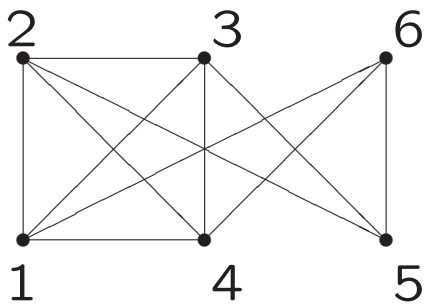
Gegeven een ongerichte graaf $\mathcal{G} = (V, E)$. Geef een kliek met zo veel mogelijk knopen (een maximale kliek).

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: $\{1, 2, 3, 4\}$.



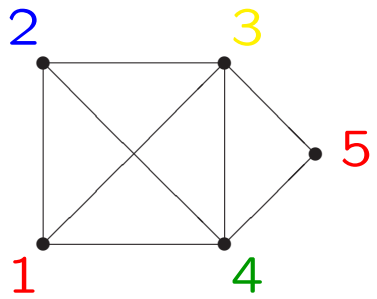
Een kleuring van (de knopen van) een ongerichte graaf $\mathcal{G} = (V, E)$ is een afbeelding $c : V \to S$, waarin S een eindige verzameling (van kleuren) is, en wel zó dat als $(v, w) \in E$ dan $c(v) \neq c(w)$. Met andere woorden: aangrenzende knopen hebben niet dezelfde kleur.

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.

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 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 een voorgestelde oplossing, maar dit zou iets anders kunnen zijn) dat gebruikt kan worden om te laten zien dat het antwoord inderdaad "ja" is.

6. Bovendien is dit certificaat kort (polynomiaal).

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

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

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.

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 $\mathcal{P}$.

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 van polynomiaal begrensde algoritmen is zelf ook weer polynomiaal begrensd (zie 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

Vraag: wat is eigenlijk de lengte van de invoer?

Voorbeeldprobleem
Gegeven een geheel getal $n > 0$. Heeft n echte delers, met andere woorden, is $n = a * b$ voor zekere $a, b > 1$?

Algoritme:

```
// gewoon alle mogelijke delers proberen
    gevonden := False;
    m := 2;
    while not gevonden and m < n do
        if n % m = 0 then
            gevonden := True;
        else
            m := m + 1;
        fi
    od
    // m is nu de kleinste deler van n
```

Lengte invoer: voorbeeld

De worst case complexiteit van dit algoritme is $O(n)$ (indien we de berekening van n % m voor 1 stap tellen, 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 $l > 1$, dan is het algoritme exponentieel in de lengte van de invoer (voor elke $l > 1$ dus).

Vraag: is het algoritme nu polynomiaal of exponentieel?

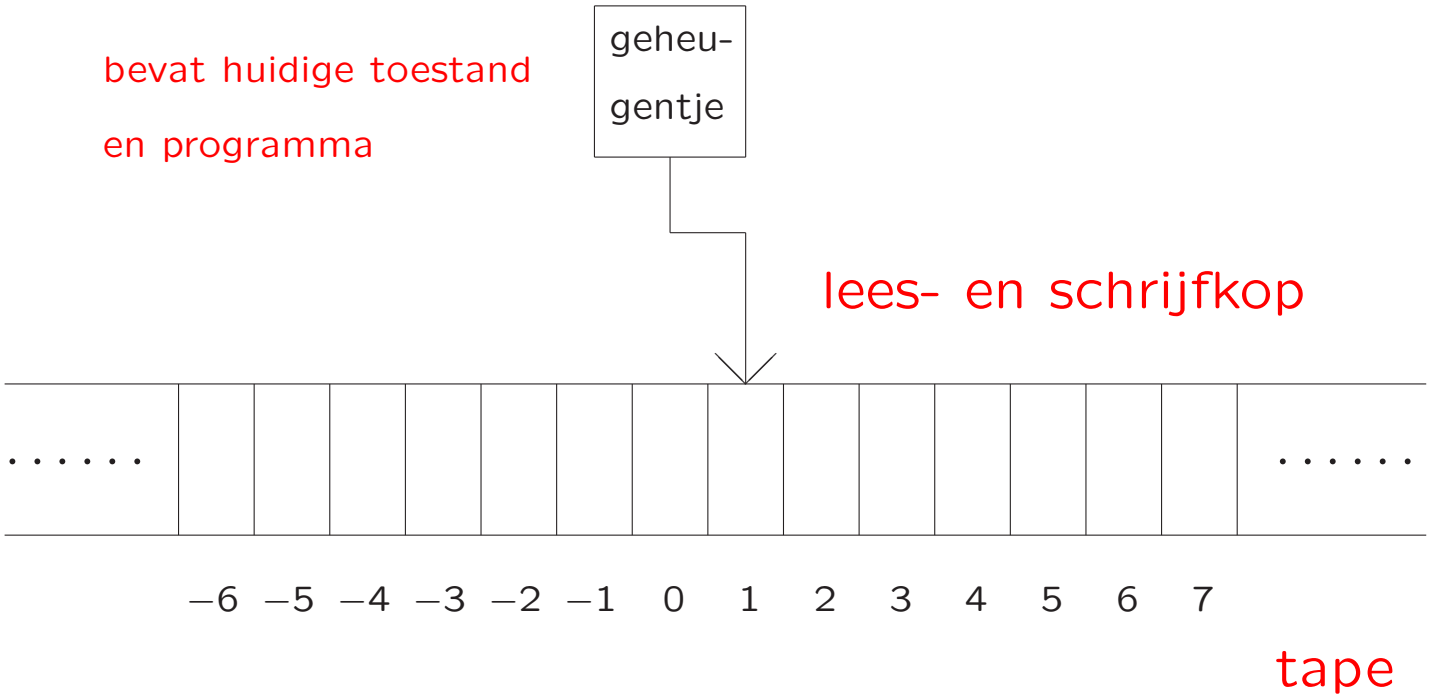
Knapzakprobleem

Gegeven een knapzak met capaciteit S (een geheel getal > 0) en n objecten met gewicht $s_1, s_2, \ldots s_n$ 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 $O(n * S)$ (dynamisch programmeren, zie Algoritmiek).

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

Deterministische one-tape Türing machine (DMT)



Een DTM-programma bevat:

- Γ: een eindige verzameling tape-symbolen (inputsymbolen 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: $Q = \{q_0, q_1, q_2, q_3, q_Y, q_N\}$.

- $\delta : Q - \{q_Y, q_N\} \times \Gamma \longrightarrow Q \times \Gamma \times \{-1, 0, 1\}$ een transitiefunctie die bepaalt wat er gebeurt als in een bepaalde toestand een bepaald karakter wordt gelezen.

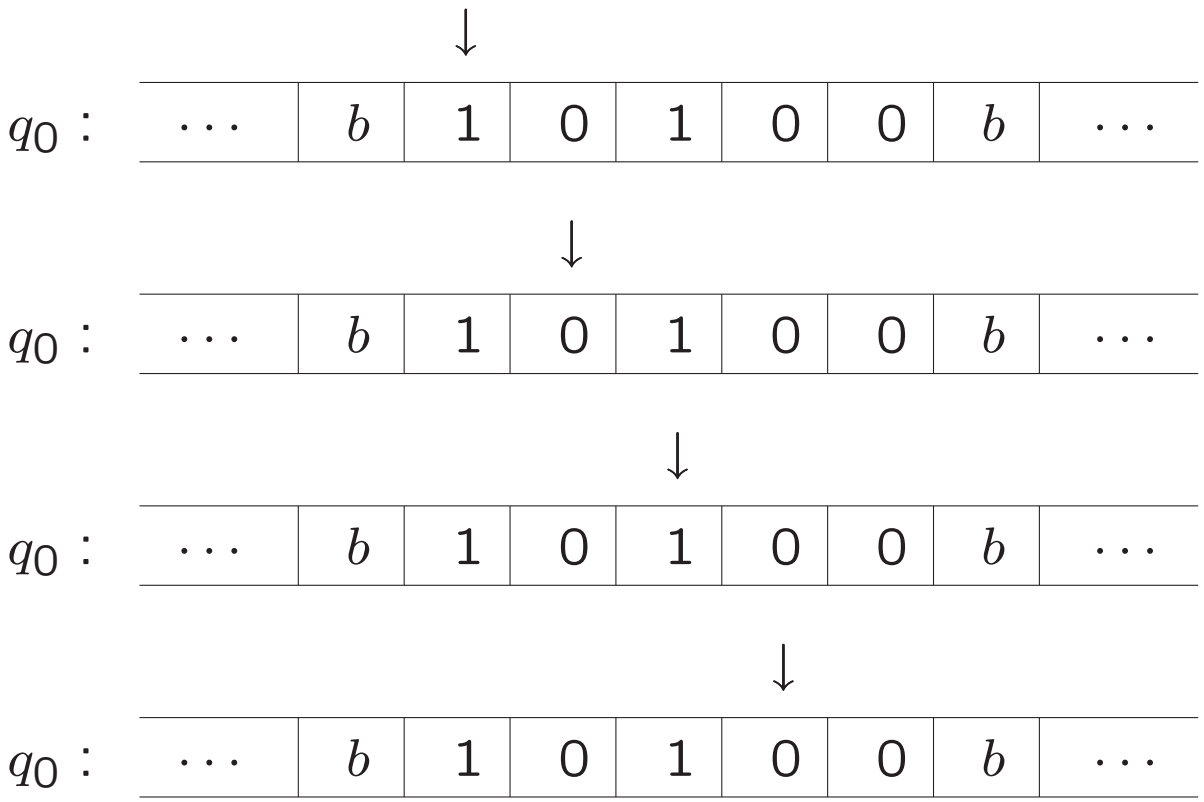
In de begintoestand staat de invoerstring x op plek 1 t/m $|x|$ en de rest van de tape is blanco. Het programma start in toestand q_0 met de lees- en schrijfkop op positie 1 en stopt als de toestand q_y (yes) of q_N (no) bereikt wordt.

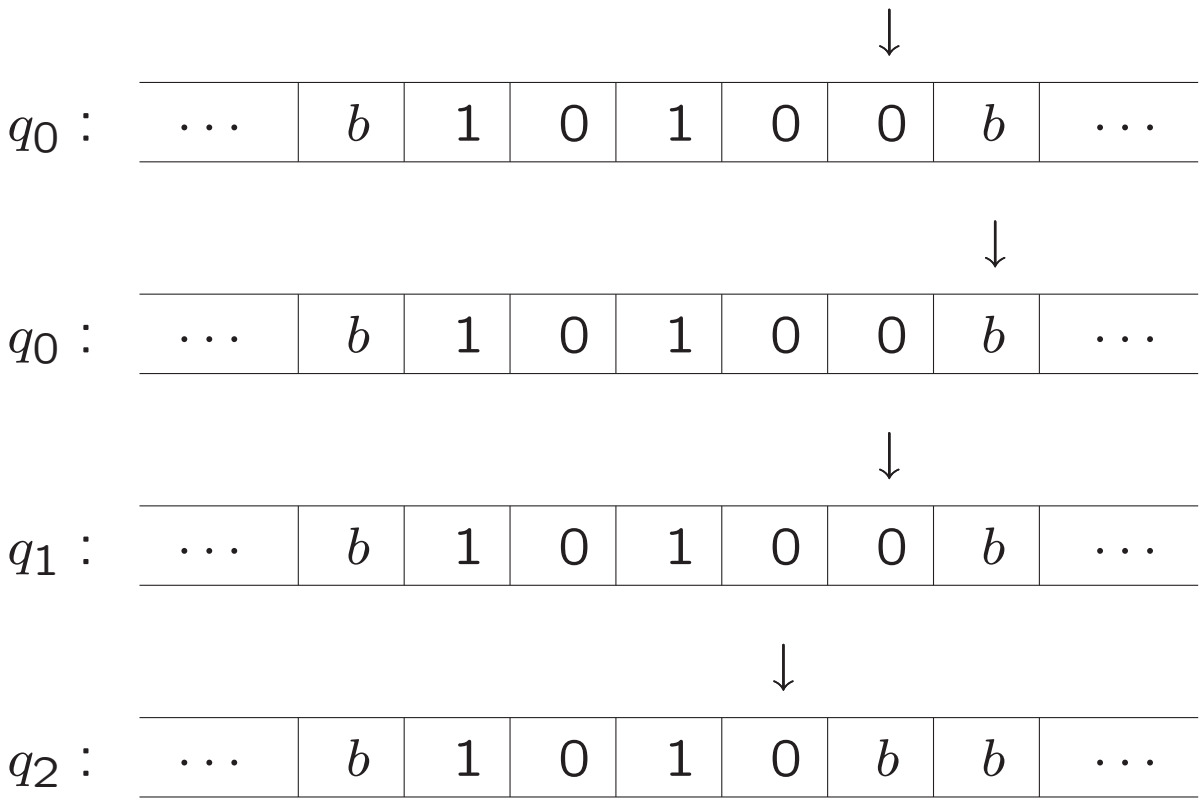
$$\Gamma = \{0, 1, b\}, \Sigma = \{0, 1\}$$

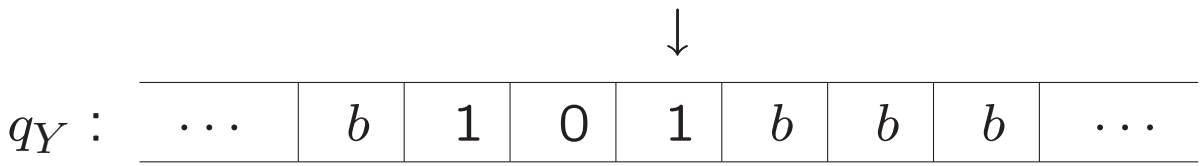
$$Q = \{q_0, q_1, q_2, q_3, q_Y, q_N\}$$

q	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?

Een certificaat kun je gebruiken om aan te tonen dat een ja-instantie van een probleem inderdaad een ja-instantie is.

Veel beslissingsproblemen zijn geformuleerd als "er-is-een"-vragen. In dat geval kun je bij een certificaat denken aan een kandidaat-oplossing (voorgestelde oplossing) van het probleem. Daarvan moet dan onder andere geverifieerd worden dat hij aan de criteria van het probleem voldoet, dus een *echte* oplossing is.

Probleem	Certificaat
HC	Hamiltonkring
SAT	waarmakende waardering
Kliek	kliek met k knopen
Kleur	kleuring met $\leq k$ kleuren
Subset Sum	deelverzameling met som $= t$
TSP	Hamiltonkring met totaalgewicht $\leq k$

Een niet-deterministisch algoritme bestaat uit 3 fasen:

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 het certificaat, het is een soort gok
   // van de oplossing van het probleem;
   // s kan echter ook een onzinstring zijn.

2. Deterministische verificatiefase
   Zowel de invoer van het probleem als de string s mogen hier gebruikt worden. Er wordt True of False geretourneerd of het programma gaat in een oneindige loop en stopt nooit.

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

3. Uitvoerstap
   Als fase 2 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 nodig om s te schrijven (dus het aantal karakters waaruit s bestaat) + het aantal stappen dat gedaan wordt in de verificatiefase (+1 voor fase 3).

Definitie
Het antwoord van een niet-deterministisch algoritme A voor invoer x is "ja" $\Longleftrightarrow$ er is een executie* van A die "ja" als uitvoer geeft.
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 een niet-deterministisch algoritme A voor invoer x is "ja" $\Longleftrightarrow$ er bestaat een string s (certificaat) waarmee je kunt aantonen dat x een ja-instantie is.

* dus een string s

Definitie
Een niet-deterministisch algoritme heet polynomiaal begrensd als er een polynoom p bestaat zodat voor elke invoer x ter grootte n (dus $n = |x|$) *waarvoor het antwoord "ja" is*, er *een executie* van het algoritme is die "ja" oplevert in hooguit $p(n)$ stappen.
Dientengevolge mag de string s in dat geval niet te lang zijn (polynomiaal in $|x|$), en het verificatie-algoritme uit fase 2 moet polynomiaal begrensd zijn in $|x|$ (en $|s|$).

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

$\mathcal{NP}$ betekent: $\mathcal{N}$on-deterministic $\mathcal{P}$olynomial time.