

POD I

Hoofdstuk 2: De uitvoeringstijd v.e. algoritme

Inhoud

1 Theoretische benadering

- Voorbeeld 1
- Voorbeeld 2
- Voorbeeld 2
- Voorbeeld 3
- Voorbeeld 4

2 Asymptotische analyse

- Voorbeeld 1
- Voorbeeld 2
- Voorbeeld 3
- Voorbeeld 4
- Voorbeeld 5

3 Oefeningen

Inhoud

1 Theoretische benadering

- Voorbeeld 1
- Voorbeeld 2
- Voorbeeld 2
- Voorbeeld 3
- Voorbeeld 4

2 Asymptotische analyse

- Voorbeeld 1
- Voorbeeld 2
- Voorbeeld 3
- Voorbeeld 4
- Voorbeeld 5

3 Oefeningen

Principe

Het aantal instructies van het algoritme exact tellen.

Assumptie : Random Access Machine

- 1) Instructies zijn sequentieel uitgevoerd
- 2) Elke primaire instructie = 1 tijdsenheid
- 3) Grenzige beschikbaarheid/gehanger dat toegankelijk is in c^k tijds

Algoritme Bereken het kwadraat van een gegeven waarde n .

berekenKwadraat (I : n : geheel getal) : *kwadraat*: geheel getal

- Preconditie: n is een geheel getal.
- Postconditie: n^2 werd berekend.
- Gebruikt: /

Algoritme Bereken het kwadraat van een gegeven waarde n .berekenKwadraat ($I : n$: geheel getal) : *kwadraat*: geheel getal

- Preconditie: n is een geheel getal.
- Postconditie: n^2 werd berekend.
- Gebruikt: /

$$\begin{matrix} \text{Product} + \text{Toenjizing} + \text{Retour} \\ , + , + , + , \end{matrix}$$

$$T(n) = 3$$

BEGIN

- $kwadraat \leftarrow n \cdot n$
- RETOUR (kwadraat)

EINDE

Algoritme Bereken het kwadraat van een gegeven waarde n .

berekenKwadraat ($l : n$: geheel getal) : *kwadraat*: geheel getal

- Preconditie: n is een geheel getal.
- Postconditie: n^2 werd berekend.
- Gebruikt: /

BEGIN

1: *kwadraat* $\leftarrow n \cdot n$

2: RETOUR (*kwadraat*)

EINDE

De uitvoeringstijd is constant:

$$T(n) = 3.$$

Bereken

$$\sum_{i=1}^n i^2 = 1^2 + 2^2 + \cdots + n^2 .$$

Algoritme Bereken de som van de eerste n kwadraten.

maakSom ($l : n$: geheel getal) : som : geheel getal

- Preconditie: n is een geheel getal.
- Postconditie: de som van de eerste n gehele kwadraten werd berekend.
- Gebruikt: /

Algoritme Bereken de som van de eerste n kwadraten.

maakSom ($l : n$: geheel getal) : som : geheel getal

- Preconditie: n is een geheel getal.
- Postconditie: de som van de eerste n gehele kwadraten werd berekend.
- Gebruikt: /

BEGIN

- 1: $som \leftarrow 0$
- 2: VOOR $i = 1$ TOT n DOE
- 3: $som \leftarrow som + i \cdot i$
- 4: EINDE VOOR
- 5: RETOUR (som)

EINDE

Algoritme Bereken de som van de eerste n kwadraten.maakSom ($l : n$: geheel getal) : som : geheel getal

- Preconditie: n is een geheel getal.
- Postconditie: de som van de eerste n gehele kwadraten werd berekend.
- Gebruikt: /

BEGIN

```

1: som ← 0
2: VOOR i = 1 TOT n DOE
3:   som ← som + i · i
4: EINDE VOOR
5: RETOUR (som)

```

EINDE

$$T(n) = 2 + 2 + 5n$$

De uitvoeringstijd is lineair: $T(n) = 5n + 4$.

2 losse instructies
loselijke toewijzingen en + en vgl: +

is n keer uitgevoerd &
bestaat uit 5 instructies
- Toewijzen i + vergelijken
+ omsluit + som + toewijzing

Algoritme Zoek het grootste element.

bepaalGrootste ($I : a : \text{array[] v. gehele getallen}$) : $\text{grootste: geheel getal}$

- Preconditie: a is een array van positieve gehele getallen, de lengte van a is n .
- Postconditie: het grootste element v.d. array werd berekend.
- Gebruikt: /

Algoritme Zoek het grootste element.

bepaalGrootste ($I : a : \text{array[] v. gehele getallen}$) : $\text{grootste: geheel getal}$

- Preconditie: a is een array van positieve gehele getallen, de lengte van a is n .
- Postconditie: het grootste element v.d. array werd berekend.
- Gebruikt: /

BEGIN

- $\text{grootste} \leftarrow 0$
- VOOR $i = 0$ TOT $n - 1$ DOE
 - ALS $a[i] \geq \text{grootste}$ DAN |
 - $\text{grootste} \leftarrow a[i]$ | → n keer 1 of 2 instructies
- EINDE ALS vooraanstaand uitgesloten → c_n
- EINDE VOOR
- RETOUR (grootste) ←

2 lage instructies
controle $i > n - 1 \rightarrow n + 1$
heer uitvoeren van 2
instructie (toesnijding
+ vgl) $\rightarrow 2(n + 1)$
 n keer 1 of 2 instructies
uitgevoerd $\rightarrow cn$

$$\tau(n) = 2 + 1(n+1) + cn = 4 + (2+c)n$$

Bereken

$$\sum_{i=1}^n \left(\sum_{j=1}^n i.j \right) .$$

Algoritme Bereken $\sum_{i=1}^n (\sum_{j=1}^n i.j)$.

berekenSommatie (l : n: geheel getal) : som: geheel getal

- Preconditie: n is een geheel getal.
- Postconditie: de som $\sum_{i=1}^n (\sum_{j=1}^n i.j)$ werd berekend.
- Gebruikt: /

Algoritme Bereken $\sum_{i=1}^n (\sum_{j=1}^n i \cdot j)$.berekenSommatie ($l : n$: geheel getal) : som : geheel getal

- Preconditie: n is een geheel getal.
- Postconditie: de som $\sum_{i=1}^n (\sum_{j=1}^n i \cdot j)$ werd berekend.
- Gebruikt: /

BEGIN

1: $som \leftarrow 0$ 2: VOOR $i = 1$ TOT n DOE3: VOOR $j = 1$ TOT n DOE4: $som \leftarrow som + i \cdot j$

5: EINDE VOOR

6: EINDE VOOR

7: RETOUR (som)

EINDE

Test $j \leq n$ wordt ook in heel uitgevoerd: 4 instructies (toevoegen & controleren j + toevoegen en controleren $i \leq n$)

n heen uitgevoerd (5 instructies)

2 lege instructies
+ laatste toekenning
 $i = n \rightarrow 4$ instructies

$$T(n) = sm^2 + 4n + 4$$

Algoritme Bereken $\sum_{i=1}^n (\sum_{j=1}^n i \cdot j)$.berekenSommatie ($l : n$: geheel getal) : som : geheel getal

- Preconditie: n is een geheel getal.
- Postconditie: de som $\sum_{i=1}^n (\sum_{j=1}^n i \cdot j)$ werd berekend.
- Gebruikt: /

BEGIN

- 1: $som \leftarrow 0$
- 2: VOOR $i = 1$ TOT n DOE
- 3: VOOR $j = 1$ TOT n DOE
- 4: $som \leftarrow som + i \cdot j$
- 5: EINDE VOOR
- 6: EINDE VOOR
- 7: RETOUR (som)

EINDE

1 Theoretische benadering

- Voorbeeld 1
- Voorbeeld 2
- Voorbeeld 2
- Voorbeeld 3
- Voorbeeld 4

2 Asymptotische analyse

- Voorbeeld 1
- Voorbeeld 2
- Voorbeeld 3
- Voorbeeld 4
- Voorbeeld 5

3 Oefeningen

Motivatie

Motivatie

- Voor grote n wordt $T(n)$ bijna uitsluitend bepaald door de leidende term.

Motivatie

- Voor grote n wordt $T(n)$ bijna uitsluitend bepaald door de leidende term.
- Voor kleine n zijn de verschillen op $T(n)$ kleiner.

Motivatie

- Voor grote n wordt $T(n)$ bijna uitsluitend bepaald door de leidende term.
- Voor kleine n zijn de verschillen op $T(n)$ kleiner.
- De exacte waarde van $T(n)$ is afhankelijk van de pc.

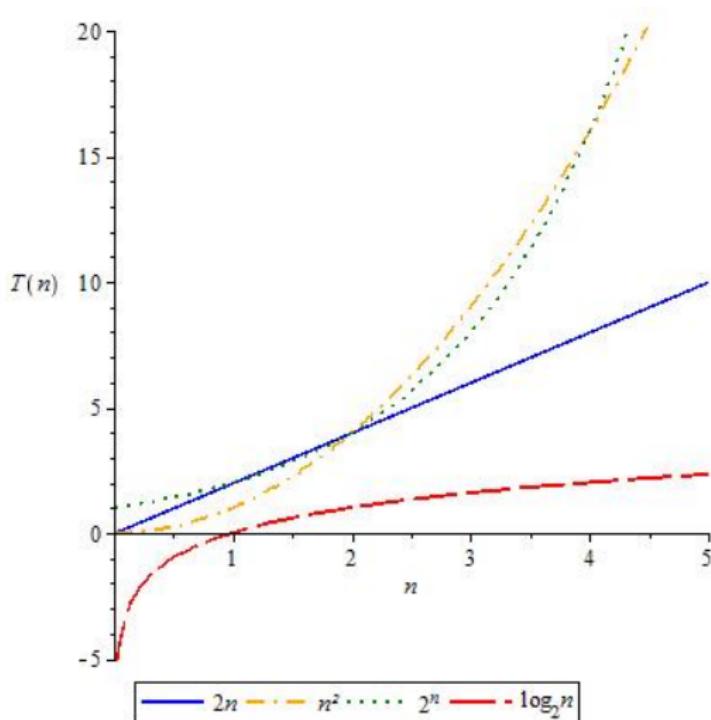
Motivatie

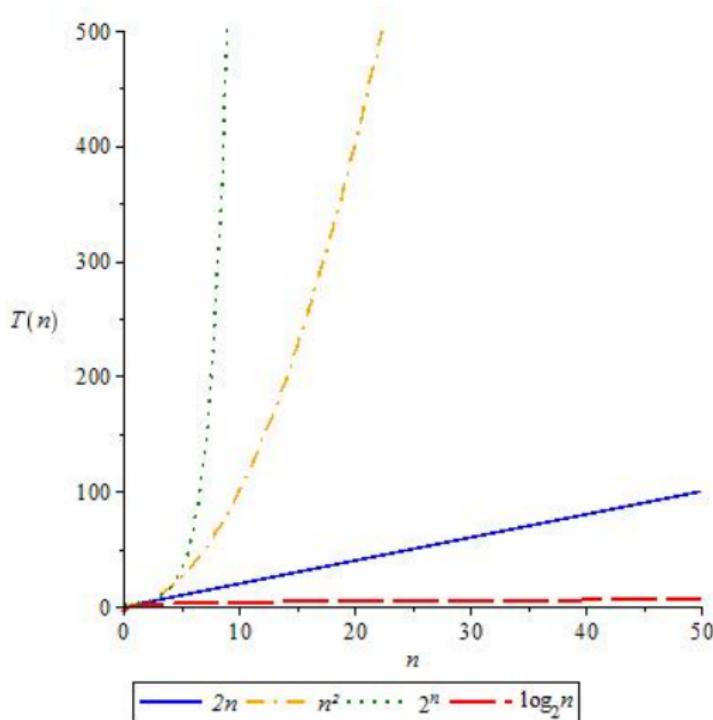
- Voor grote n wordt $T(n)$ bijna uitsluitend bepaald door de leidende term.
- Voor kleine n zijn de verschillen op $T(n)$ kleiner.
- De exacte waarde van $T(n)$ is afhankelijk van de pc.

Motivatie

- Voor grote n wordt $T(n)$ bijna uitsluitend bepaald door de leidende term.
- Voor kleine n zijn de verschillen op $T(n)$ kleiner.
- De exacte waarde van $T(n)$ is afhankelijk van de pc.

→ Θ -notatie = 'big O' notatie (*Order of growth*)
i.e. in $T(n)$ worden constanten en lagere-ordetermen genegeerd.

Kleine inputgrootte n 

Grote inputgrootte n 

Voorbeeld 1

-
- 1: $som \leftarrow 0$
 - 2: VOOR $i = 1$ TOT n DOE
 - 3: $som \leftarrow som + i$
 - 4: EINDE VOOR
-

Voorbeeld 1

-
- 1: $som \leftarrow 0$
 - 2: VOOR $i = 1$ TOT n DOE
 - 3: $som \leftarrow som + i$
 - 4: EINDE VOOR
-

Er geldt:

$$T(n) = c_2 + c_1 n \quad (c_1, c_2 \in \mathbb{N})$$

c_1 const. t.o.v. n = *marginal*

Voorbeeld 1

-
- 1: $som \leftarrow 0$
 - 2: VOOR $i = 1$ TOT n DOE
 - 3: $som \leftarrow som + i$
 - 4: EINDE VOOR
-

Er geldt:

$$\begin{aligned}T(n) &= c_2 + c_1 n \quad (c_1, c_2 \in \mathbb{N}) \\&= \Theta(n).\end{aligned}$$

↳ dominante term

Voorbeeld 2

-
- 1: $som \leftarrow 0$
 - 2: VOOR $i = 1$ TOT n DOE
 - 3: VOOR $j = 1$ TOT i DOE
 - 4: $som \leftarrow som + j$
 - 5: EINDE VOOR
 - 6: EINDE VOOR
-

Voorbeeld 2

-
- 1: $som \leftarrow 0$
 - 2: VOOR $i = 1$ TOT n DOE
 - 3: VOOR $j = 1$ TOT i DOE
 - 4: $som \leftarrow som + j$
 - 5: EINDE VOOR
 - 6: EINDE VOOR

 $c_1 \rightarrow$ enkele instructies

\sim een herhaaldoop

$c_2 \sim$

voor elk i
 i -heen herhaaldoop

$$\sum_{i=1}^n i = \frac{n(n+1)}{2}$$

$c_3 \sim \frac{(n+1)}{2}$

$$T(m) = c_1 + c_2 m + c_3 \frac{n(n+1)}{2}$$

Er geldt:

$$T(n) = \Theta(n^2).$$

\downarrow
leidende term

Voorbeeld 3

-
- 1: $som \leftarrow 0$
 - 2: VOOR $i = 1$ TOT n DOE
 - 3: VOOR $j = 1$ TOT n DOE
 - 4: $som \leftarrow som + j$
 - 5: EINDE VOOR
 - 6: EINDE VOOR
- c, ene intuïtie*
- $c_3 n^2$ heel*
- $c_1 n$ heel*

$$T(n) = c_1 + c_2 n + c_3 n^2$$

Voorbeeld 3

-
- 1: $som \leftarrow 0$
 - 2: VOOR $i = 1$ TOT n DOE
 - 3: VOOR $j = 1$ TOT n DOE
 - 4: $som \leftarrow som + j$
 - 5: EINDE VOOR
 - 6: EINDE VOOR
-

Er geldt:

$$T(n) = \Theta(n^2).$$

Voorbeeld 4

Stel $n = 2^k$ ($k \in \mathbb{N}$).

Voorbeeld 4

Stel $n = 2^k$ ($k \in \mathbb{N}$).

* Lees $\log_2 n$ als 'de macht welke je aan 2 moet toekennen om n te bereiken. Aangezien $n = 2^k$ is dit dan k .

1: $som \leftarrow 0$ 2: $i \leftarrow 1$ 3: ZOLANG $i \leq n$ DOE4: $i \leftarrow i \cdot 2 \rightarrow$ olnr opeenvolgens $2, 4, \dots, n = 2^k$
en $2^k = 2^{k+1}$ 5: VOOR $j = 1$ TOT n DOE6: $som \leftarrow som + j$

7: EINDE VOOR

8: EINDE ZOLANG

$k+1$
heer
uitgewereld

$(k+1) \cdot n$
heer
uitgewereld

$\overline{\overline{j}}$
hier pastelt
bruteerde
lens

$$\text{met } n = 2^k \rightarrow k = \log_2 n$$

$$\begin{aligned}
 T(n) &= c_1 + \underline{c_2 (k+1)} + \underline{c_3 (k+1) \cdot n} \\
 &= c_1 + c_2 (\log_2 n + 1) + \underline{c_3 (\log_2 n + 1) \cdot n} \\
 &= \Theta(n \log n)
 \end{aligned}$$

Voorbeeld 4

Stel $n = 2^k$ ($k \in \mathbb{N}$).

- 1: *som* $\leftarrow 0$
 - 2: $i \leftarrow 1$
 - 3: ZOLANG $i \leq n$ DOE
 - 4: $i \leftarrow i \cdot 2$
 - 5: VOOR $j = 1$ TOT n DOE
 - 6: *som* \leftarrow *som* + j
 - 7: EINDE VOOR
 - 8: EINDE ZOLANG
-

Er geldt:

$$T(n) = \Theta(n \lg n).$$

Voorbeeld 5

Stel $n = 2^k$ ($k \in \mathbb{N}$).

Voorbeeld 5

Stel $n = \underline{2^k}$ ($k \in \mathbb{N}$).

- 1: $som \leftarrow 0$
- 2: $i \leftarrow 1$
- 3: ZOLANG $i \leq n$ DOE
- 4: VOOR $j = 1$ TOT i DOE ?!
- 5: $som \leftarrow som + j$
- 6: EINDE VOOR
- 7: $i \leftarrow i \cdot 2$
- 8: EINDE ZOLANG

<i>voori =</i>	<i>doorgang j waarbij</i>	<i>#x o.a.eenstukken i doorlopen</i>
$1 = 2^0$	1	
$2 = 2^1$	1, 2	
$4 = 2^2$	1, 2, 3, 4	
\vdots	\vdots	
$n = 2^k$	$1, 2, \dots, n$	

Binnerke eins voorstel

$$1 + 2 + 4 + \dots + \cancel{n} = 1 + 2 + 4 + \dots + \cancel{2^k} = \sum_{l=0}^k 2^l$$

heer herhaald, met

$$\sum_{l=0}^k 2^l = \frac{1 - 2^{k+1}}{1 - 2}$$

$$T(n) = \frac{1 - 2^{k+1}}{1 - 2} = 2^{k+1} - 1 = 2 \cdot \cancel{2^k} - 1 = \cancel{2n} - 1$$

** Formule gegeven*

$$= \Theta(n)$$

Voorbeeld 5

Stel $n = 2^k$ ($k \in \mathbb{N}$).

- 1: *som* $\leftarrow 0$
 - 2: $i \leftarrow 1$
 - 3: ZOLANG $i \leq n$ DOE
 - 4: VOOR $j = 1$ TOT i DOE
 - 5: *som* \leftarrow *som* + j
 - 6: EINDE VOOR
 - 7: $i \leftarrow i \cdot 2$
 - 8: EINDE ZOLANG
-

Er geldt:

$$T(n) = \Theta(n).$$

1 Theoretische benadering

- Voorbeeld 1
- Voorbeeld 2
- Voorbeeld 2
- Voorbeeld 3
- Voorbeeld 4

2 Asymptotische analyse

- Voorbeeld 1
- Voorbeeld 2
- Voorbeeld 3
- Voorbeeld 4
- Voorbeeld 5

3 Oefeningen

Oefeningen

Oefening 1

Oefening 1

Onderstaande tabel geeft voor een aantal waarden van n de uitvoeringstijd corresponderend met de gegeven Θ -notatie. Vul de tabel verder aan:

$T(n)$	$n =$	8	128	1 024	1 000 000
$\Theta(n)$		8	128	1 024	1 000 000
$\Theta(n^2)$					
$\Theta(n^{\frac{1}{2}})$					
$\Theta(\log_2 n)$					
$\Theta(n \log_2 n)$					

Theoretische benadering
oooooo

Asymptotische analyse
ooooo

Oefeningen
○●oooooooooooo

Oefeningen

Oefening 2

Oefening 2

Veronderstel dat een zeker algoritme een tijdscomplexiteit $T(n) = 2^n$ heeft en dat het uitvoeren van een implementatie ervan op een bepaalde computer T_0 seconden duurt voor een inputgrootte n_0 . Veronderstel vervolgens dat we het programma kunnen uitvoeren op een machine die 64 keer sneller is dan de eerste computer. Wat is de inputgrootte die op de nieuwe machine kan verwerkt worden in T_0 seconden? (Bron [Fack, 2004])

Theoretische benadering
oooooo

Asymptotische analyse
ooooo

Oefeningen
oo●oooooooooooo

Oefeningen

Oefening 3

Oefeningen

Oefening 3

Bepaal de Θ -notatie voor de volgende uitdrukkingen:

a) $6n + 1$

Oefening 3

Bepaal de Θ -notatie voor de volgende uitdrukkingen:

- a) $6n + 1$
- b) $6n^3 + 12n^2 + 1$

Oefening 3

Bepaal de Θ -notatie voor de volgende uitdrukkingen:

- a) $6n + 1$
- b) $6n^3 + 12n^2 + 1$
- c) $2\log n + 4n + 3n\log n$

Oefening 3

Bepaal de Θ -notatie voor de volgende uitdrukkingen:

- a) $6n + 1$
- b) $6n^3 + 12n^2 + 1$
- c) $2\log n + 4n + 3n\log n$
- d) $2 + 4 + 6 + \dots + 2n$

Oefening 3

Bepaal de Θ -notatie voor de volgende uitdrukkingen:

- a) $6n + 1$
- b) $6n^3 + 12n^2 + 1$
- c) $2\log n + 4n + 3n\log n$
- d) $2 + 4 + 6 + \dots + 2n$
- e) $(6n + 4)(1 + \log n)$

Oefening 3

Bepaal de Θ -notatie voor de volgende uitdrukkingen:

- a) $6n + 1$
- b) $6n^3 + 12n^2 + 1$
- c) $2\log n + 4n + 3n\log n$
- d) $2 + 4 + 6 + \dots + 2n$
- e) $(6n + 4)(1 + \log n)$
- f) $2n^2 + 1$

Oefening 3

Bepaal de Θ -notatie voor de volgende uitdrukkingen:

- a) $6n + 1$
- b) $6n^3 + 12n^2 + 1$
- c) $2\log n + 4n + 3n\log n$
- d) $2 + 4 + 6 + \dots + 2n$
- e) $(6n + 4)(1 + \log n)$
- f) $2n^2 + 1$
- g) $3n^2 + 2n\log n$

Oefening 3

Bepaal de Θ -notatie voor de volgende uitdrukkingen:

- a) $6n + 1$
- b) $6n^3 + 12n^2 + 1$
- c) $2\log n + 4n + 3n\log n$
- d) $2 + 4 + 6 + \dots + 2n$
- e) $(6n + 4)(1 + \log n)$
- f) $2n^2 + 1$
- g) $3n^2 + 2n\log n$
- h) $6n^6 + n + 4$

Oefeningen

Oefening 3

Bepaal de Θ -notatie voor de volgende uitdrukkingen:

a) $6n + 1 \rightarrow \Theta(n)$

b) $6n^3 + 12n^2 + 1 \rightarrow \Theta(n^3)$

c) $2\log n + 4n + 3n\log n \rightarrow \Theta(n \log n)$

d) $2 + 4 + 6 + \dots + 2n \rightarrow 2 \cdot (1 + 2 + 3 + \dots + n) = 2 \cdot \frac{n(n+1)}{2} \rightarrow \Theta(n^2)$

e) $(6n + 4)(1 + \log n) \sim \Theta(n \log n)$

f) $2n^2 + 1 \rightarrow \Theta(n^2)$

g) $3n^2 + 2n\log n \rightarrow \Theta(n^2)$

h) $6n^6 + n + 4 \rightarrow \Theta(n^6)$

i) $\frac{(6n + 1)^2}{2} + 12n + 1 \rightarrow \Theta(n^2)$

Oefening 3

Bepaal de Θ -notatie voor de volgende uitdrukkingen:

- a) $6n + 1$
- b) $6n^3 + 12n^2 + 1$
- c) $2\log n + 4n + 3n\log n$
- d) $2 + 4 + 6 + \dots + 2n$
- e) $(6n + 4)(1 + \log n)$
- f) $2n^2 + 1$
- g) $3n^2 + 2n\log n$
- h) $6n^6 + n + 4$
- i) $(6n + 1)^2$
- j) $1 + 2 + 4 + 8 + 16 + \dots + 2^n.$

Oefeningen

Oefening 4

Oefening 4

Tel in de gegeven programmafragmenten het aantal keer dat de uitdrukking $x \leftarrow x + 1$ wordt uitgevoerd. Geef vervolgens de Θ -notatie voor de uitvoeringstijd van deze fragmenten.

a) VOOR $i = 1$ TOT n

$x \leftarrow x + 1$

EINDE VOOR

Oefening 4

Tel in de gegeven programmafragmenten het aantal keer dat de uitdrukking $x \leftarrow x + 1$ wordt uitgevoerd. Geef vervolgens de Θ -notatie voor de uitvoeringstijd van deze fragmenten.

a) VOOR $i = 1$ TOT n | $\sim \bar{c} n$ keer oorlog
 $x \leftarrow x + 1$ $T(n) = c_1 + c_2 n \sim \Theta(n)$
 EINDE VOOR

b) VOOR $i = 1$ TOT $2n$
 VOOR $j = 1$ TOT n | $\sim \bar{c} n$ keer oorlog
 $x \leftarrow x + 1$ $\left. \begin{array}{l} i \\ j \end{array} \right\} \sim 2n$ keer oorlog
 EINDE VOOR
 EINDE VOOR

$$T(n) = c_1 + c_2 \cdot 2 \cdot n + c_3 \cdot 2 \cdot n \cdot n \sim \Theta(n^2)$$

Oefening 4

c) VOOR $i = 1$ TOT n VOOR $j = 1$ TOT i VOOR $k = 1$ TOT j $x \leftarrow x + 1$

EINDE VOOR

EINDE VOOR

EINDE VOOR

$$T(n) = 1 + (1+2) + (1+2+3) + \dots + \sum_{i=1}^n i$$

$$= \sum_{m=1}^n \left(\sum_{i=1}^m i \right) = \sum_{m=1}^n \frac{m(m+1)}{2} = \frac{1}{2} m \cdot \frac{(m+1)(m+2)}{3}$$

formule bewerken en later

i	j	k	$\# k'$
1	1	1	1
2	1	1	1
	2	1, 2	2
3	1	1	1
	2	1, 2	2
	3	1, 2, 3	3
\vdots	\vdots	\vdots	\vdots
n	1	1	1
	2	1, 2	2
	\vdots	\vdots	\vdots
n	$1, 2, \dots, n$	$\sum_{i=1}^n i$	$\sum_{i=1}^n i$

$$\text{Totaal} = 1 + 3 + \dots + \sum_{i=1}^n i$$

Oefeningen

Oefening 4

d) VOOR $i = 1$ TOT n VOOR $j = 1$ TOT i VOOR $k = 1$ TOT j $x \leftarrow x + 1$

$$\begin{array}{c|c} i & j \\ \hline 1 & 1 \\ 2 & 1, 2 \\ 3 & 1, 2, 3 \\ 4 & 1, 2, 3, 4 \\ \vdots & \vdots \\ m & 1, 2, \dots, m \end{array}$$

i	j	k	$\#k'$
1	1	1	1
2	1, 2		2
3	1, 2, 3		3
4	1, 2, 3, 4		4
\vdots	\vdots		
m	$1, 2, \dots, m$		m
$?$	$1, 2, \dots, m$		m
m	$1, 2, \dots, m$		m

Totaal = $1 + 2.2 + 3.3 + \dots + m.m$

$$\begin{aligned}
 T(m) &= 1 + 2.2 + 3.3 + \dots + m.m \\
 &= 1^2 + 2^2 + 3^2 + \dots + m^2 \\
 &= \sum_{i=1}^m i^2 = \frac{m(m+1)(2m+1)}{6} \rightsquigarrow \Theta(m^3)
 \end{aligned}$$

Oefening 4

e) stel $n = 2^k$

$i \leftarrow n$

ZOLANG ($i \geq 1$) DOE

$x \leftarrow x + 1$

$i \leftarrow i/2$

EINDE ZOLANG

?

Oefeningen

Oefening 4

f) stel $n = 3^k$ $j \leftarrow n$ ZOLANG ($j \geq 1$) DOEVOOR $i = 1$ TOT j $x \leftarrow x + 1$

EINDE VOOR

 $j \leftarrow j/3$

EINDE ZOLANG

 $T_{ij}:$ gebruik formule $\sum_{i=0}^{\infty} \alpha^i = \frac{1-\alpha}{1-\alpha}$

voor j	andere i	# $i \geq j$
$\frac{n}{3^k} = 3^{k-j}$	$, \dots, 3^k$	3^{k-j}
$\frac{n}{3^j} = 3^{k-j-1}$	$, \dots, 3^{k-1}$	3^{k-j-1}
$\frac{n}{3^2} = 3^{k-j-2}$	$, \dots, 3^{k-2}$	3^{k-j-2}
\vdots		
$\frac{n}{3^k} = 3^{k-k} = 0$	1	1

Total = ...

Oefening 4

g) Stel $n = 2^k$

$i \leftarrow n$

ZOLANG ($i \geq 1$) DOE

VOOR $j = 1$ TOT n  ?
 $x \leftarrow x + 1$

EINDE VOOR

$i \leftarrow i/2$

EINDE ZOLANG

Theoretische benadering
oooooo

Asymptotische analyse
ooooo

Oefeningen
oooooooo●○

Oefeningen

Oefening 5

Oefening 5

Stel n een natuurlijk getal. Beschouw de onderstaande programmafragmenten.

Algoritme A

```
1.som ← 0
2.VOOR i = 1 TOT n DOE
3.    VOOR j = 1 TOT i DOE
4.        som ← som + j
5.    EINDE VOOR
6.EINDE VOOR
7.RETOUR(som)
```

Algoritme B

```
1.som ← 0
2.VOOR i = 1 TOT n DOE
3.    VOOR j = 1 TOT i DOE
4.        som ← som + i
5.    EINDE VOOR
6.EINDE VOOR
7.RETOUR(som)
```

Oefening 5

Stel n een natuurlijk getal. Beschouw de onderstaande programmafragmenten.

Algoritme A

```

1.som ← 0
2.VOOR i = 1 TOT n DOE
3.    VOOR j = 1 TOT i DOE
4.        som ← som + j
5.    EINDE VOOR
6.EINDE VOOR
7.RETOUR(som)

```

Algoritme B

```

1.som ← 0
2.VOOR i = 1 TOT n DOE
3.    VOOR j = 1 TOT i DOE
4.        som ← som + i
5.    EINDE VOOR
6.EINDE VOOR
7.RETOUR(som)

```

- a) Hoeveel wordt er in beide gevallen in de volledige binnenste lus (die beschreven is van regel 3 tot en met regel 5) aan de voorlopige som toegevoegd
- als $i = 2$?
 - als $i = 4$?
 - als $i = n$?

Oefening 5

Stel n een natuurlijk getal. Beschouw de onderstaande programmafragmenten.

Algoritme A

```

1.som ← 0
2.VOOR i = 1 TOT n DOE
3.    VOOR j = 1 TOT i DOE
4.        som ← som + j
5.    EINDE VOOR
6.EINDE VOOR
7.RETOUR(som)

```

Algoritme B

```

1.som ← 0
2.VOOR i = 1 TOT n DOE
3.    VOOR j = 1 TOT i DOE
4.        som ← som + i
5.    EINDE VOOR
6.EINDE VOOR
7.RETOUR(som)

```

- a) Hoeveel wordt er in beide gevallen in de volledige binnenste lus (die beschreven is van regel 3 tot en met regel 5) aan de voorlopige som toegevoegd
 - i. als $i = 2$?
 - ii. als $i = 4$?
 - iii. als $i = n$?
- b) Wat is in beide gevallen de retourwaarde van de som
 - i. als $n = 2$?
 - ii. als $n = 4$?
 - iii. voor willekeurige n ?

Oefeningen

Oefening 5

Algoritme A

1. som $\leftarrow 0$
2. VOOR $i = 1$ TOT n DOE
 3. VOOR $j = 1$ TOT i DOE
 4. som \leftarrow som + j
 5. EINDE VOOR
6. EINDE VOOR
7. RETOUR(som)

a) Als $i = 2 \rightarrow j$ van 1 tot 2

$$\begin{array}{lll} -j=1 & \text{som} = +j & \text{som} = +1 \\ j=2 & \text{som} = +j & \text{som} = +2 \end{array}$$

Dus, als $i = 2 \rightarrow \text{som} = 3$ Als $i = 4$

$$\begin{array}{lll} -j=1 & \rightarrow \text{som} = +1 \\ j=2 & \text{som} = +2 \\ j=3 & \text{som} = +3 \\ j=4 & \text{som} = +4 \end{array} \left. \begin{array}{l} \text{som} = +10 \\ (\text{1+2+3+4}) \end{array} \right\}$$

Als $i = n \rightarrow \text{som} = +(\sum_{j=1}^n j) \rightarrow +\frac{n(n+1)}{2}$

Algoritme B

1. som $\leftarrow 0$
2. VOOR $i = 1$ TOT n DOE
 3. VOOR $j = 1$ TOT i DOE
 4. som \leftarrow som + i
 5. EINDE VOOR
6. EINDE VOOR
7. RETOUR(som)

Als $i = 2 \rightarrow j$ van 1 tot 2

$$\begin{array}{lll} -j=1 & \text{som} = +i & \text{som} = +2 \\ j=2 & \text{som} = +i & \text{som} = +2 \end{array} \left. \begin{array}{l} \\ \end{array} \right\} +2^2$$

Als $i = 4 \rightarrow j$ van 1 tot 4

$$\begin{array}{lll} j=1 & \text{som} = +i & \text{som} = +4 \\ j=2 & " & " \\ j=3 & " & " \\ j=4 & " & " \end{array} \left. \begin{array}{l} \\ \\ \\ \end{array} \right\} +4^2$$

Als $i = n \rightarrow +n^2$

Oefening 5

Algoritme A

```
1.som ← 0  
2.VOOR  $i = 1$  TOT  $n$  DOE  
3.    VOOR  $j = 1$  TOT  $i$  DOE  
4.        som ← som +  $j$   
5.    EINDE VOOR  
6.EINDE VOOR  
7.RETOUR(som)
```

Algoritme B

```
1.som ← 0  
2.VOOR  $i = 1$  TOT  $n$  DOE  
3.    VOOR  $j = 1$  TOT  $i$  DOE  
4.        som ← som +  $i$   
5.    EINDE VOOR  
6.EINDE VOOR  
7.RETOUR(som)
```

- c) Hoeveel keer wordt voor algoritme A de opdracht 'som ← som + j ' uitgevoerd en voor algoritme B de opdracht 'som ← som + i ' voor een willekeurige inputgrootte n ?

Oefening 5

Algoritme A

```
1.som ← 0  
2.VOOR  $i = 1$  TOT  $n$  DOE  
3.    VOOR  $j = 1$  TOT  $i$  DOE  
4.        som ← som +  $j$   
5.    EINDE VOOR  
6.EINDE VOOR  
7.RETOUR(som)
```

Algoritme B

```
1.som ← 0  
2.VOOR  $i = 1$  TOT  $n$  DOE  
3.    VOOR  $j = 1$  TOT  $i$  DOE  
4.        som ← som +  $i$   
5.    EINDE VOOR  
6.EINDE VOOR  
7.RETOUR(som)
```

- c) Hoeveel keer wordt voor algoritme A de opdracht 'som ← som + j ' uitgevoerd en voor algoritme B de opdracht 'som ← som + i ' voor een willekeurige inputgrootte n ?
- d) Geef voor beide algoritmen de asymptotische notatie voor de uitvoeringstijd.

Oefening 5

Algoritme A

```
1.som ← 0
2.VOOR i = 1 TOT n DOE
3.    VOOR j = 1 TOT i DOE
4.        som ← som + j
5.    EINDE VOOR
6.EINDE VOOR
7.RETOUR(som)
```

Algoritme B

```
1.som ← 0
2.VOOR i = 1 TOT n DOE
3.    VOOR j = 1 TOT i DOE
4.        som ← som + i
5.    EINDE VOOR
6.EINDE VOOR
7.RETOUR(som)
```

- c) Hoeveel keer wordt voor algoritme A de opdracht 'som ← som + j' uitgevoerd en voor algoritme B de opdracht 'som ← som + i' voor een willekeurige inputgrootte n ?
- d) Geef voor beide algoritmen de asymptotische notatie voor de uitvoeringstijd.
- e) Volg het verloop van beide programmafragmenten voor $n = 3$. Doe dit door de waarden van de variabelen i , j en som te noteren in een voortgangstabel: telkens één van de variabelen een waarde wordt toegewezen moet deze waarde op een nieuwe regel van de tabel genoteerd worden.

-  T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein,
Introduction to Algorithms, The MIT Press, 2003.
-  V. Fack, *Datastructuren en Algoritmen I*, Academia Press, 2004.
-  R. Johnsonbaugh, M. Schaefer, *Algorithms*, Pearson Prentice Hall, 2004.
-  N. Wirth, *Algorithms + Data Structures = Programs*, Prentice Hall, 1976.