Negende college complexiteit

11 april 2025

Polynoomevaluatie

Matrixvermenigvuldiging

- Shellsort
  - sorteert deelrijtjes in rondes aan de hand van een serie (afnemende) stapgroottes: h_i-sorteren
  - complexiteit sterk afhankelijk van de serie stapgroottes; Shell: $O(n^2)$ en Hibbard: $O(n\sqrt{n})$ in de worst case

- Merge Insertion sort
  - bereikt de theoretische ondergrens voor enkele kleine waarden van n en is optimaal voor nog wat meer kleine n
  - niet optimaal voor grotere waarden van n, maar wel relatief dicht bij de theoretische ondergrens

- Counting sort
  - gebruikt geen arrayvergelijkingen
  - maakt gebruik van begrensd bereik van de $A[i]$
  - tellen en in de juiste volgorde neerzetten: $O(n)$ (als ...)
  - stabiele sorteermethode: volgorde gelijke elementen behouden

- Radix sort
  - sorteert cijfer voor cijfer, van het minst significante naar het meest significante cijfer
  - per cijfer is een stabiele sorteermethode nodig
  - $O(n)$ (als ...)

Zij $p(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0$ een polynoom van graad $n \geq 1$, met alle a_i reële getallen ($a_i \in \mathbb{R}$).

Probleem:

Gegeven: $a_0, a_1, \ldots, a_{n-1}, a_n$ en x
Gevraagd: $p(x)$

Van links naar rechts de termen $a_i x^i$ berekenen en optellen levert een $\Theta(n^2)$-algoritme.

Als we het polynoom daarentegen van rechts naar links evalueren kunnen we eenvoudig een ordeverbetering bereiken.

Algoritme 1: "gewoon"

$pol := a_0 + a_1 \times x$; // $n \geq 1$
$macht := x$;
for $i := 2$ **to** n **do**
    $macht := macht \times x$;
    // berekent $x^2, x^3, \ldots$
    $pol := pol + a_i \times macht$;
od

Basisoperatie: $\times$ en $+, -$

Complexiteit: aantal $\times$: $2n - 1$
aantal $+, -$: n

Algoritme 2: methode van Horner*

$pol := a_n$;
for $i := n - 1$ **downto** 0 **do**
    $pol := pol \times x + a_i$;
od

Gebaseerd op:

$$p(x) = a_0 + x \times (a_1 + x \times (a_2 + \cdots + x \times (a_{n-1} + x \times a_n) \cdots))$$

Complexiteit: aantal $\times$: n
aantal $+, -$: n

Vraag: kan het met minder $\times$ en $+, -$?
Antwoord: nee !

*gepopulariseerd door W. Horner (1819); Chinese bronnen bekend van eeuwen eerder

Algoritmen gebaseerd op het doen van vergelijkingen konden we beschrijven met **beslissingsbomen**.
Een model om rekenkundige algoritmen (algoritmen die zijn gebaseerd op $+, -, \times$ en $/$) mee te beschrijven: schema's.
Een schema

- is een eindige serie stappen van de vorm $s_i := q \circ r$;
- hierin is $\circ$ een rekenkundige operatie: $\times, /, +$ of $-$
- q en r zijn constanten (bijvoorbeeld $1, -1, \pi^2, \ldots$) of invoerwaarden (hier a_i's of x) of tussenresultaten van eerdere stappen
- de laatste stap uit het schema berekent het eindresultaat (hier dus $p(x)$)

Horner met $n = 2$: $s_1 := a_2 \times x$; $s_2 := s_1 + a_1$; $s_3 := s_2 \times x$; $s_4 := s_3 + a_0$;

Stelling. Elk schema (dat alleen $+, -$ en $\times$ gebruikt) om $a_0 + a_1x + a_2x^2 + \ldots + a_nx^n$ te berekenen moet ten minste n $(+,-)$-stappen doen en n $\times$-stappen.
Bewijs voor $(+,-)$-stappen volgt (vervang x door 1) uit:

Lemma. Een schema (dat alleen $+, -$ en $\times$ gebruikt) om $a_0 + a_1 + a_2 + \ldots + a_{n-1} + a_n$ te berekenen moet ten minste n $(+,-)$-stappen doen.

Het bewijs gaat met inductie naar n, met $n = 0$ als flauw basisgeval. Vervang overal a_n door 0. Dit geeft een schema dat $a_0 + a_1 + a_2 + \ldots + a_{n-1}$ berekent. Kijk naar de eerste $(+,-)$-stap die a_n gebruikt (die bestaat zeker): $s_i := q \pm 0$ of $s_i := 0 + r$ of $s_i := 0 - r$; laat de twee eerste weg en vervang overal s_i door q danwel r, of $s_i := -1 \times r$ vervangt zonodig de derde. We hebben zo een $(+,-)$-stap geëlimineerd uit het schema. Gebruik nu inductie.

De methode van Horner berekent $p(x)$ met n vermenigvuldigingen en n optellingen ($+/-$). Er bestaat geen algoritme dat het probleem voor algemene p en x met minder $\times$'s en $+/-$'s kan oplossen. De methode van Horner is derhalve optimaal.*

Maar misschien kan het wel beter voor polynomen die een speciale vorm hebben.

*de schets van het bewijs van de ondergrens (vorige twee slides) is geen tentamenstof

Polynoomevaluatie met preprocessing: bewerk het polynoom tot een polynoom in een speciale vorm waarop een nieuw evaluatie-algoritme sneller werkt.

Het polynoom

Laat $p(x) = x^n + a_{n-1}x^{n-1} + \ldots + a_1 x + a_0$, met $n = 2^k - 1$.

$p(x)$ is dus een monisch polynoom, dat wil zeggen dat $a_n = 1$. We kunnen zonder beperking der algemeenheid aannemen dat het polynoom dat we willen evalueren monisch is*. Hetzelfde geldt voor de aanname dat $n = 2^k - 1$.

*zie toelichting volgende slide

We kunnen zonder beperking der algemeenheid aannemen dat het polynoom dat we willen evalueren monisch is. Hetzelfde geldt voor de aanname dat $n = 2^k - 1$.

Laat $p(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0$. Dan:

(i) $p(x) = a_n \cdot (x^n + \frac{a_{n-1}}{a_n}\ x^{n-1} + \cdots + \frac{a_1}{a_n}\ x + \frac{a_0}{a_n}) = a_n \cdot q(x)$
Nu is q een monisch polynoom, en als we q in x kunnen evalueren hoeven we de uitkomt alleen nog maar met a_n te vermenigvuldigen.

(ii) Als $2^{k-1} - 1 < n < 2^k - 1$, dan is $q(x) = x^{2^k-1} + p(x)$ monisch en $p(x) = q(x) - x^{2^k-1}$. Dus als we q in x kunnen evalueren, hoeven we van de uitkomst alleen nog de –snel te berekenen– term x^{2^k-1} af te trekken.

De speciale vorm (*recursief* geformuleerd):

$$p(x) = (x^j + b) \times q(x) + r(x),$$

waarin q en r ook weer monisch zijn en in de speciale vorm staan, beide van graad $2^{k-1} - 1$ zijn, en $j = 2^{k-1}$.

Voorbeeld (met $n = 7$)

$$p(x) = x^7 + 6x^6 + 5x^5 + 4x^4 + 3x^3 + 2x^2 + x + 1 =$$

$$(x^4+2)[(x^2+4)(x+6)+(x-20)]+[(x^2-10)(x-10)+(x-107)]$$

Om dit polynoom in x te evalueren gebruikt de methode van Horner 7 vermenigvuldigingen en 7 optellingen ($+/-$), maar het kan (ga dit na) met 5 resp. 10.

Een gegeven monisch polynoom p van graad $n = 2^k - 1$ is eenvoudig om te zetten naar de speciale vorm.

We willen p schrijven als: $p(x) = (x^j + b) \times q(x) + r(x)$ met q en r monisch, beide van graad $2^{k-1} - 1$, en $j = 2^{k-1}$. De waarde van b en de coëfficiënten van q en r zijn hieruit simpel af te lezen.
Immers: als $q(x) = x^{j-1} + q_{j-2}x^{j-2} + \ldots + q_0$ en $r(x) = x^{j-1} + r_{j-2}x^{j-2} + \ldots + r_0$, dan geldt:

$$b + 1 = a_{j-1}, q_\ell = a_{\ell+j}, b \times q_\ell + r_\ell = a_\ell,$$

voor $\ell = 0, 1, \ldots, j-2$ en de a_i de coëfficiënten van p.

Vervolgens kunnen q en r op dezelfde manier in de speciale vorm gebracht worden, etc. Algoritme met complexiteit: zie opgave 48.

Als het polynoom p in de juiste vorm* staat kan $p(x)$ als volgt geëvalueerd worden:

1. evalueer $q(x)$ en $r(x)$ recursief

2. bereken de x^j's: nodig hiervoor zijn $x, x^2, x^4, \ldots, x^{2^{k-1}}$. Bereken deze alle van tevoren: $k-1$ vermenigvuldigingen

3. vermenigvuldig $(x^j + b)$ met $q(x)$ en tel er $r(x)$ bij op: 1 vermenigvuldiging en 2 optellingen

*Merk op: als $k = 1$ (dus $n = 1$) en p is monisch, dan is $p(x) = x + a_0$, en dan staat p dus al in de juiste vorm

Zij $M(k)$ = het aantal vermenigvuldigingen dat gedaan wordt om een monisch polynoom (in de speciale vorm) van graad $2^k - 1$ te evalueren, zonder de berekening van de x^j mee te tellen.

Dan voldoet $M(k)$ aan de volgende recurrente betrekking:

$$M(k) = \begin{cases} 0 & k = 1 \\ 2M(k-1) + 1 & k > 1 \end{cases}$$

Oplossing: $M(k) = 2^{k-1} - 1 \longrightarrow \frac{n-1}{2}$

Zij $A(k)$ = het aantal optellingen dat gedaan wordt om een monisch polynoom (in de speciale vorm) van graad $2^k - 1$ te evalueren.

Dan voldoet $A(k)$ aan de volgende recurrente betrekking:

$$A(k) = \begin{cases} 1 & k = 1 \\ 2A(k-1) + 2 & k > 1 \end{cases}$$

Oplossing: $A(k) = 3 \cdot 2^{k-1} - 2 \longrightarrow \frac{3n-1}{2}$

Gegeven een monisch polynoom van graad $2^k - 1$ dat in de speciale vorm staat. We berekenen $M(k)$ en $A(k)$.

- Basisgeval $k = 1$, dus $n = 1$.
  Dan geldt dat $p(x) = x + a_0$. Om p in x te evalueren zijn dus 0 vermenigvuldigingen en 1 optelling nodig

- Als $k > 1$ is $p(x) = (x^j + b) \times q(x) + r(x)$, met $j = 2^{k-1}$ en q en r zijn ook monisch en hebben de speciale vorm
  - we moeten nu 2 polynomen van hetzelfde soort, maar van graad $2^{k-1} - 1$ evalueren in x. Dat kost $2M(k-1)$ vermenigvuldigingen en $2A(k-1)$ optellingen
  - als we q en r in x hebben geëvalueerd doen we nog 1 vermenigvuldiging en 2 optellingen
  - samen dus $2M(k-1) + 1$ vermenigvuldigingen en $2A(k-1) + 2$ optellingen

Zij A en B twee $n \times n$ matrices met elementen a_{ij} en b_{ij} ($1 \leq i, j \leq n$). De elementen c_{ij} van het (matrix-)product $C = A \cdot B$ zijn dan als volgt gedefinieerd: $c_{ij} = \sum_{k=1}^{n} a_{ik} \times b_{kj}$

Een standaard $\Theta(n^3)$ algoritme hiervoor is recht-toe recht-aan uit definitie:

for $i := 1$ **to** n **do**
  for $j := 1$ **to** n **do**
    $c_{ij} := 0$;
    for $k := 1$ **to** n **do**
      $c_{ij} := c_{ij} + a_{ik} \times b_{kj}$;
    od
  od
od

Voorbeeld: product van twee 2×2 matrices algemeen.

$$\underbrace{\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}}_{A} \underbrace{\begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix}}_{B} = \underbrace{\begin{pmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{pmatrix}}_{C}$$

$$\begin{aligned} c_{11} &= a_{11} \times b_{11} + a_{12} \times b_{21} \\ c_{12} &= a_{11} \times b_{12} + a_{12} \times b_{22} \\ c_{21} &= a_{21} \times b_{11} + a_{22} \times b_{21} \\ c_{22} &= a_{21} \times b_{12} + a_{22} \times b_{22} \end{aligned}$$

Het product van twee matrices is overigens ook gedefinieerd voor niet-vierkante matrices A en B, mits maar geldt dat aantal kolommen van A = aantal rijen van B.

Voorbeeld: product van twee 2×2 matrices.
We berekenen c_{21}

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \begin{pmatrix} 5 & 6 \\ 7 & 8 \end{pmatrix} = \begin{pmatrix} 19 & 22 \\ 43 & 50 \end{pmatrix}$$

A B C

via $c_{21} = a_{21} \times b_{11} + a_{22} \times b_{21}$.

We kunnen c_{21} ook berekenen als:

$$(a_{21} + a_{22}) \times b_{11} + a_{22} \times (b_{21} - b_{11})$$

Dit lijkt weinig zinvol, maar toch ...

In totaal kost de recht-toe recht-aan methode (links) voor dit voorbeeld 8 vermenigvuldigingen van array-elementen. Het kan echter door 'herschrijven' met 7 (rechts):

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \begin{pmatrix} 5 & 6 \\ 7 & 8 \end{pmatrix} = \begin{pmatrix} 19 & 22 \\ 43 & 50 \end{pmatrix}$$

$$\begin{aligned} 19 &= 1 \times 5 + 2 \times 7 \\ 22 &= 1 \times 6 + 2 \times 8 \\ 43 &= 3 \times 5 + 4 \times 7 \\ 50 &= 3 \times 6 + 4 \times 8 \end{aligned}$$

$$\begin{aligned} M_1 &= (1+4) \times (5+8) = 65 \\ M_2 &= (3+4) \times 5 = 35 \\ M_3 &= 1 \times (6-8) = -2 \\ M_4 &= 4 \times (7-5) = 8 \\ M_5 &= (1+2) \times 8 = 24 \\ M_6 &= (3-1) \times (5+6) = 22 \\ M_7 &= (2-4) \times (7+8) = -30 \\ 19 &= M_1 + M_4 - M_5 + M_7 \\ 22 &= M_3 + M_5 \\ 43 &= M_2 + M_4 \\ 50 &= M_1 - M_2 + M_3 + M_6 \end{aligned}$$

Algemeen: $M_2 = (a_{21} + a_{22}) \times b_{11}$; $M_4 = a_{22} \times (b_{21} - b_{11})$; dan $M_2 + M_4 = c_{21}$, etc.

Neem aan dat $n = 2^k$. We kunnen dan de matrices A, B en C ieder opsplitsen in vier $\frac{n}{2} \times \frac{n}{2}$ deelmatrices:

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}, B = \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix}, C = \begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix}$$

Terzijde: de complexiteit van matrixvermenigvulding is in $\Omega(n^2)$, want ieder algoritme moet in ieder geval de $n \times n$-matrices A en B lezen en het resultaat (alle c_{ij}) berekenen/schrijven.

We kunnen $C = A \cdot B$ nu recursief berekenen via*:

$$\begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix} = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \cdot \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix}$$

met
$$\begin{aligned} C_{11} &= A_{11} \cdot B_{11} + A_{12} \cdot B_{21} \\ C_{12} &= A_{11} \cdot B_{12} + A_{12} \cdot B_{22} \\ C_{21} &= A_{21} \cdot B_{11} + A_{22} \cdot B_{21} \\ C_{22} &= A_{21} \cdot B_{12} + A_{22} \cdot B_{22} \end{aligned}$$

Dit zijn 8 vermenigvuldigingen en 4 optellingen van $\frac{n}{2} \times \frac{n}{2}$-matrices. Voor $M(k)$, het aantal vermenigvuldigingen van array-elementen, geldt nu: $M(k) = 8M(k-1)$ en $M(0) = 1$, wat leidt tot $M(k) = 8^k = n^3$. Dat is hetzelfde als de 'gewone' methode.

*Ga voor $n = 4$ na dat dit correct is door een en ander uit te schrijven

Analoog aan het 2×2 geval kunnen we de berekening van de vier C_{ij}'s herschrijven zodanig dat we nog maar 7 vermenigvuldigingen van $\frac{n}{2} \times \frac{n}{2}$-matrices hoeven te doen.

$S_1, \ldots, S_{10}$ zijn $\frac{n}{2} \times \frac{n}{2}$-matrices die alle de som of het verschil van twee deelmatrices van A en B zijn:

$$
\begin{aligned}
S_1 &= B_{12} - B_{22} & \qquad S_6 &= B_{11} + B_{22} \\
S_2 &= A_{11} + A_{12} & \qquad S_7 &= A_{12} - A_{22} \\
S_3 &= A_{21} + A_{22} & \qquad S_8 &= B_{21} + B_{22} \\
S_4 &= B_{21} - B_{11} & \qquad S_9 &= A_{11} - A_{21} \\
S_5 &= A_{11} + A_{22} & \qquad S_{10} &= B_{11} + B_{12}
\end{aligned}
$$

*1969, Volker Strassen (1939 –)

$P_1, \ldots, P_7$ zijn $\frac{n}{2} \times \frac{n}{2}$ matrices die alle het product zijn van twee matrices S en/of deelmatrices van A en B:

$$
\begin{array}{lll}
P_1 & = A_{11} \cdot S_1 & = A_{11} \cdot B_{12} - A_{11} \cdot B_{22} \\
P_2 & = S_2 \cdot B_{22} & = A_{11} \cdot B_{22} + A_{12} \cdot B_{22} \\
P_3 & = S_3 \cdot B_{11} & = A_{21} \cdot B_{11} + A_{22} \cdot B_{11} \\
P_4 & = A_{22} \cdot S_4 & = A_{22} \cdot B_{21} - A_{22} \cdot B_{11} \\
P_5 & = S_5 \cdot S_6 & = A_{11} \cdot B_{11} + A_{11} \cdot B_{22} + A_{22} \cdot B_{11} + A_{22} \cdot B_{22} \\
P_6 & = S_7 \cdot S_8 & = A_{12} \cdot B_{21} + A_{12} \cdot B_{22} - A_{22} \cdot B_{21} - A_{22} \cdot B_{22} \\
P_7 & = S_9 \cdot S_{10} & = A_{11} \cdot B_{11} + A_{11} \cdot B_{12} - A_{21} \cdot B_{11} - A_{21} \cdot B_{12}
\end{array}
$$

Merk op dat de berekening in de rechterkolom alleen voor onze informatie is, de enige *feitelijke* matrixvermenigvuldigingen staan in de middenkolom! Dit zijn er 7.

De matrices C_{11}, C_{12}, C_{21} en C_{22} kunnen nu als volgt bepaald worden:

$$\begin{array}{lll} C_{11} & = P_5 + P_4 - P_2 + P_6 & = A_{11} \cdot B_{11} + A_{12} \cdot B_{21} \\ C_{12} & = P_1 + P_2 & = A_{11} \cdot B_{12} + A_{12} \cdot B_{22} \\ C_{21} & = P_3 + P_4 & = A_{21} \cdot B_{11} + A_{22} \cdot B_{21} \\ C_{22} & = P_5 + P_1 - P_3 - P_7 & = A_{21} \cdot B_{12} + A_{22} \cdot B_{22} \end{array}$$

We krijgen dus zeven keer een (recursieve) matrixvermenigvuldiging van $\frac{n}{2} \times \frac{n}{2}$-matrices, en we doen achttien optellingen van $\frac{n}{2} \times \frac{n}{2}$-matrices per recursiestap.

Denk er aan dat matrixvermenigvuldiging niet commutatief is: doorgaans geldt $A \cdot B \neq B \cdot A$. Bij het herschrijven op deze en de vorige twee slides is echter nergens stiekem $X \cdot Y$ door $Y \cdot X$ is vervangen; alle gelijkheden zijn dan ook geldig. Controleer zelf dat het allemaal klopt.

Dit levert de volgende recurrente betrekkingen op voor het aantal vermenigvuldigingen van array-elementen $M(k)$ en het aantal optellingen van array-elementen $A(k)$, met $n = 2^k$:

$$M(k) = \begin{cases} 1 & k = 0 \\ 7M(k-1) & k > 0 \end{cases}$$

$$A(k) = \begin{cases} 0 & k = 0 \\ 7A(k-1) + 18 \cdot (2^{k-1})^2 & k > 0 \end{cases}$$

Oplossing: $M(k) = 7^k = 2^{\lg 7^k} = 2^{k \cdot \lg 7} = n^{\lg 7} \approx n^{2,81}$
$A(k) = 6 \cdot 7^k - 6 \cdot 4^k \in \Theta(n^{\lg 7})$.

We hebben dus: Strassen (1969) met $\Theta(n^{2,81})$

Het kan nog sneller!

Pan (1978):	$\Theta(n^{2,796})$
Coppersmith-Winograd (1990):	$\Theta(n^{2,376})$
Stothers (2010):	$\Theta(n^{2,3737})$
Williams (2012):	$\Theta(n^{2,3729})$
Le Gall (2014):	$\Theta(n^{2,3728639})$
Vassilevska Williams et al. (2024):	$\Theta(n^{2,371552})$

Dit soort algoritmen heeft echter zulke grote constanten verstopt in de Θ dat ze alleen asymptotisch sneller zijn dan Strassen voor matrices die te groot zijn om met de huidige hardware te berekenen. Vermoeden: $\Theta(n^{2+\epsilon})$ voor elke $\epsilon > 0$.

nature

Explore content ⌄ About the journal ⌄ Publish with us ⌄

nature › articles › article

Article | Open access | Published: 07 June 2023

Faster sorting algorithms discovered using deep reinforcement learning

Daniel J. Mankowitz ✉, Andrea Michi, Anton Zhernov, Marco Gelmi, Marco Selvi, Cosmin Paduraru, Edouard Leurent, Shariq Iqbal, Jean-Baptiste Lespiau, Alex Ahern, Thomas Köppe, Kevin Millikin, Stephen Gaffney, Sophie Elster, Jackson Broshear, Chris Gamble, Kieran Milan, Robert Tung, Minjae Hwang, Taylan Cemgil, Mohammadamin Barekatain, Yujia Li, Amol Mandhane, Thomas Hubert, … David Silver + Show authors

Nature **618**, 257–263 (2023) | Cite this article

422k Accesses | **1054** Altmetric | Metrics

Abstract

Fundamental algorithms such as sorting or hashing are used trillions of times on any given day[1]. As demand for computation grows, it has become critical for these algorithms to be as performant as possible. Whereas remarkable progress has been achieved in the past[2], making further improvements on the efficiency of these routines has proved challenging for both human scientists and computational approaches. Here we show how artificial intelligence can go beyond the current state of the art by discovering hitherto unknown routines. To realize this, we formulated the task of finding a better sorting routine as a single-player game. We then trained a new deep reinforcement learning agent, AlphaDev, to play this game. AlphaDev discovered small sorting algorithms from scratch that outperformed previously known human benchmarks. These algorithms have been integrated into the LLVM standard C++ sort

From: Discovering faster matrix multiplication algorithms with reinforcement learning

$$\begin{pmatrix} a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} & a_{1,5} \\ a_{2,1} & a_{2,2} & a_{2,3} & a_{2,4} & a_{2,5} \\ a_{3,1} & a_{3,2} & a_{3,3} & a_{3,4} & a_{3,5} \\ a_{4,1} & a_{4,2} & a_{4,3} & a_{4,4} & a_{4,5} \end{pmatrix} \begin{pmatrix} b_{1,1} & b_{1,2} & b_{1,3} & b_{1,4} & b_{1,5} \\ b_{2,1} & b_{2,2} & b_{2,3} & b_{2,4} & b_{2,5} \\ b_{3,1} & b_{3,2} & b_{3,3} & b_{3,4} & b_{3,5} \\ b_{4,1} & b_{4,2} & b_{4,3} & b_{4,4} & b_{4,5} \\ b_{5,1} & b_{5,2} & b_{5,3} & b_{5,4} & b_{5,5} \end{pmatrix} = \begin{pmatrix} c_{1,1} & c_{1,2} & c_{1,3} & c_{1,4} & c_{1,5} \\ c_{2,1} & c_{2,2} & c_{2,3} & c_{2,4} & c_{2,5} \\ c_{3,1} & c_{3,2} & c_{3,3} & c_{3,4} & c_{3,5} \\ c_{4,1} & c_{4,2} & c_{4,3} & c_{4,4} & c_{4,5} \end{pmatrix}$$

$$\begin{aligned}
h_1 &= a_{3,2}(-b_{2,1} - b_{2,5} - b_{3,1}) \\
h_2 &= (a_{2,2} + a_{2,5} - a_{3,5})(-b_{2,5} - b_{5,1}) \\
h_3 &= (-a_{3,1} - a_{4,1} + a_{4,2})(-b_{1,1} + b_{2,5}) \\
h_4 &= (a_{1,2} + a_{1,4} + a_{3,4})(-b_{2,5} - b_{4,1}) \\
h_5 &= (a_{1,5} + a_{2,2} + a_{2,5})(-b_{2,4} + b_{5,1}) \\
h_6 &= (-a_{2,2} - a_{2,5} - a_{4,5})(b_{2,5} + b_{5,1}) \\
h_7 &= (-a_{1,1} + a_{4,1} - a_{4,2})(b_{1,1} + b_{2,4}) \\
h_8 &= (a_{3,2} - a_{3,3} - a_{4,3})(-b_{2,3} + b_{3,1}) \\
h_9 &= (-a_{1,2} - a_{1,4} + a_{4,4})(b_{2,3} + b_{4,1}) \\
h_{10} &= (a_{2,3} + a_{2,5})b_{5,1} \\
h_{11} &= (-a_{2,1} - a_{4,1} + a_{4,2})(-b_{1,1} + b_{2,2}) \\
h_{12} &= (a_{4,1} - a_{4,2})b_{1,1} \\
h_{13} &= (a_{1,2} + a_{1,4} + a_{2,4})(b_{2,2} + b_{4,1}) \\
h_{14} &= (a_{1,3} - a_{3,3} + a_{3,4})(b_{2,4} + b_{3,1}) \\
h_{15} &= (-a_{1,2} - a_{1,4})b_{4,1} \\
h_{16} &= (-a_{3,2} + a_{3,3})b_{3,1} \\
h_{17} &= (a_{1,3} + a_{1,4} - a_{2,1} + a_{2,2} - a_{2,3} + a_{2,4} - a_{3,2} + a_{3,3} - a_{4,1} + a_{4,2})b_{2,2} \\
h_{18} &= a_{2,1}(b_{1,1} + b_{1,2} + b_{5,2}) \\
h_{19} &= -a_{2,5}(b_{5,1} + b_{5,2} + b_{5,5}) \\
h_{20} &= (-a_{1,5} + a_{2,1} + a_{2,3} - a_{2,5})(-b_{1,1} - b_{1,2} + b_{1,4} - b_{5,2}) \\
h_{21} &= (a_{2,1} + a_{2,3} - a_{2,5})b_{5,2} \\
h_{22} &= (a_{1,3} - a_{1,4} - a_{2,4})(b_{1,1} + b_{1,2} - b_{1,4} - b_{3,1} - b_{3,2} + b_{3,4} + b_{4,4}) \\
h_{23} &= a_{1,3}(-b_{3,1} + b_{3,4} + b_{4,4}) \\
h_{24} &= a_{1,5}(-b_{4,4} - b_{5,1} + b_{5,4}) \\
h_{25} &= -a_{1,3}(b_{1,1} - b_{1,4}) \\
h_{26} &= (-a_{1,3} + a_{1,4} + a_{1,5})b_{4,4} \\
h_{27} &= (a_{1,3} - a_{3,1} + a_{3,5})(b_{1,1} - b_{1,4} + b_{1,5} + b_{5,5}) \\
h_{28} &= -a_{3,4}(-b_{3,5} - b_{4,1} - b_{4,5}) \\
h_{29} &= a_{3,1}(b_{1,1} + b_{1,5} + b_{3,5}) \\
h_{30} &= (a_{3,1} - a_{3,3} + a_{3,4})b_{3,5} \\
h_{31} &= (-a_{1,4} - a_{1,5} - a_{3,4})(-b_{4,4} - b_{5,1} + b_{5,4} - b_{5,5}) \\
h_{32} &= (a_{2,1} + a_{4,1} + a_{4,4})(b_{1,3} - b_{4,1} - b_{4,2} - b_{4,3}) \\
h_{33} &= a_{4,3}(-b_{3,1} - b_{3,3})
\end{aligned}$$

$$\begin{aligned}
h_{51} &= a_{2,2}(b_{2,1} + b_{2,2} - b_{5,1}) \\
h_{52} &= a_{4,2}(b_{1,1} + b_{2,1} + b_{2,3}) \\
h_{53} &= -a_{1,2}(-b_{2,1} + b_{2,4} + b_{4,1}) \\
h_{54} &= (a_{1,2} + a_{1,4} - a_{2,2} - a_{2,5} - a_{3,2} + a_{3,3} - a_{4,2} + a_{4,3} - a_{4,4} - a_{4,5})b_{2,3} \\
h_{55} &= (a_{1,4} - a_{4,4})(-b_{2,3} + b_{3,1} + b_{3,3} - b_{3,4} + b_{4,3} - b_{4,4}) \\
h_{56} &= (a_{1,1} - a_{1,5} - a_{4,1} + a_{4,5})(b_{3,1} + b_{3,3} - b_{3,4} + b_{5,1} + b_{5,3} - b_{5,4}) \\
h_{57} &= (-a_{3,1} - a_{4,1})(-b_{1,3} - b_{1,5} - b_{2,5} - b_{5,1} - b_{5,3} - b_{5,5}) \\
h_{58} &= (-a_{1,4} - a_{1,5} - a_{3,4} - a_{3,5})(-b_{5,1} + b_{5,4} - b_{5,5}) \\
h_{59} &= (-a_{3,3} + a_{3,4} - a_{4,3} + a_{4,4})(b_{4,1} + b_{4,3} + b_{4,5} + b_{5,1} + b_{5,3} + b_{5,5}) \\
h_{60} &= (a_{2,5} + a_{4,5})(b_{2,3} - b_{5,1} - b_{5,2} - b_{5,3} - b_{5,2} - b_{5,3}) \\
h_{61} &= (a_{1,4} + a_{3,4})(b_{1,1} - b_{1,4} + b_{1,5} - b_{2,5} - b_{4,4} + b_{4,5} - b_{5,1} + b_{5,4} - b_{5,5}) \\
h_{62} &= (a_{2,1} + a_{4,1})(b_{1,2} + b_{1,3} + b_{2,2} - b_{4,1} - b_{4,2} - b_{4,3}) \\
h_{63} &= (-a_{3,3} - a_{4,3})(-b_{2,3} - b_{3,3} - b_{3,5} - b_{4,1} - b_{4,3} - b_{4,5}) \\
h_{64} &= (a_{1,1} - a_{1,3} - a_{1,4} + a_{4,1} - a_{4,3} - a_{4,4})(b_{1,1} - b_{1,4} + b_{1,5}) \\
h_{65} &= (-a_{1,1} + a_{4,1})(-b_{1,3} + b_{1,4} + b_{2,4} - b_{5,1} - b_{5,3} + b_{5,4}) \\
h_{66} &= (a_{1,1} - a_{1,2} + a_{1,3} - a_{1,5} - a_{2,2} - a_{2,5} - a_{3,2} + a_{3,3} - a_{4,1} + a_{4,2})b_{2,4} \\
h_{67} &= (a_{2,5} - a_{3,5})(b_{1,1} + b_{1,2} + b_{1,5} - b_{2,5} - b_{4,1} - b_{4,2} - b_{4,5} + b_{5,2} + b_{5,5}) \\
h_{68} &= (a_{1,1} + a_{1,3} - a_{1,4} - a_{1,5} - a_{4,1} - a_{4,3} + a_{4,4} + a_{4,5})(-b_{5,1} - b_{5,3} + b_{5,4}) \\
h_{69} &= (-a_{1,3} + a_{1,4} - a_{2,3} + a_{2,4})(-b_{2,4} - b_{5,1} - b_{5,2} + b_{5,4} - b_{5,2} + b_{5,4}) \\
h_{70} &= (a_{2,3} - a_{2,5} + a_{4,3} - a_{4,5})(-b_{3,1} - b_{3,2} - b_{3,3}) \\
h_{71} &= (-a_{3,1} + a_{3,3} - a_{3,4} + a_{3,5} - a_{4,1} + a_{4,3} - a_{4,4} + a_{4,5})(-b_{5,1} - b_{5,3} - b_{5,5}) \\
h_{72} &= (-a_{2,1} - a_{2,4} - a_{4,1} - a_{4,4})(b_{4,1} + b_{4,2} + b_{4,3}) \\
h_{73} &= (a_{1,3} - a_{1,4} - a_{1,5} + a_{2,3} - a_{2,4} - a_{2,5})(b_{1,1} + b_{1,2} - b_{1,4} + b_{2,4} + b_{5,2} - b_{5,4}) \\
h_{74} &= (a_{2,1} - a_{2,3} + a_{2,4} - a_{3,1} + a_{3,3} - a_{3,4})(b_{4,1} + b_{4,2} + b_{4,3}) \\
h_{75} &= -(a_{1,2} + a_{1,4} - a_{2,2} - a_{2,5} - a_{3,1} + a_{3,2} + a_{3,4} + a_{3,5} - a_{4,1} + a_{4,2})b_{2,5} \\
h_{76} &= (a_{1,3} + a_{3,3})(-b_{1,1} + b_{1,4} - b_{1,5} + b_{2,4} + b_{3,4} - b_{3,5}) \\
c_{1,1} &= -h_{10} + h_{12} + h_{14} - h_{15} - h_{16} + h_{53} + h_5 - h_{66} - h_7 \\
c_{2,1} &= h_{10} + h_{11} - h_{12} + h_{13} + h_{15} + h_{16} - h_{17} - h_{44} + h_{51} \\
c_{3,1} &= h_{10} - h_{12} + h_{15} + h_{16} - h_1 + h_2 + h_3 - h_4 + h_{75} \\
c_{4,1} &= -h_{10} + h_{12} - h_{15} - h_{16} + h_{52} + h_{54} - h_6 - h_8 + h_9 \\
c_{1,2} &= h_{13} + h_{15} + h_{20} + h_{21} - h_{22} + h_{23} + h_{25} - h_{43} + h_{49} + h_{50} \\
c_{2,2} &= -h_{11} + h_{12} - h_{13} - h_{15} - h_{16} + h_{17} + h_{18} - h_{19} - h_{21} + h_{43} + h_{44} \\
c_{3,2} &= -h_{16} - h_{19} - h_{21} - h_{28} - h_{29} - h_{36} + h_{42} + h_{44} - h_{47} + h_{48} \\
&\dots
\end{aligned}$$

4 bij 5 keer 5 bij 5

76 vermenigvuldigingen

- **Volgende college:** vrijdag 25 april, 11.00 – 12.45,
  Gorlaeusgebouw zaal BM.0.40

- **Straks werkcollege:** vrijdag 11 april, 13.15 - 15.00,
  Gorlaeusgebouw zalen BW.0.29 en BW.0.30
  Opgaven: 43, 44, 47, 48 ac, 45, 46b,
  opgave 2 tentamen 4/6/2019

- **Derde huiswerkopgave (Shellsort):**
  deadline maandag 28 april

- **Website:**
  liacs.leidenuniv.nl/~graafjmde/COMP/