Complexiteit 2023 — college 7

21 maart 2023

---

lets anders dan rijtjes

Vorige keer

- Shellsort
  - sorteert deelrijtjes in rondes a.d.h.v. een serie (afnemende) stapgroottes: k-sorteren
  - complexiteit sterk afhankelijk van serie stapgroottes; Shell-rijtje $O(n^2)$ en Hibbard $O(n\sqrt{n})$
- Binary Insertion sort
- Merge Insertion sort: bereikt de theoretische ondergrens voor een aantal kleine n, maar nog steeds niet optimaal
- Counting sort: tellen en in de juiste volgorde overschrijven; $\Theta(n + k)$, of $O(n)$ als $k \in O(n)$.
- Stabiel sorteren: behoudt volgorde gelijke elementen
- Radix sort: sorteert cijfer voor cijfer, van minst significant (rechts) naar meest significant (links); stabiel intern sorteeralgoritme nodig; $\Theta(dk + dn)$, of $O(n)$ als d constant en $k \in O(n)$.

Vandaag

- Polynoomevaluatie
- Matrixvermenigvuldiging
- Eulerkringen
- Hamiltonkringen

Polynoomevaluatie

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: $a_0, a_1, \ldots, a_n$ en x
Gevraagd: $p(x)$

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

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

Gewone methode

Algoritme 1: "gewoon"

```
1 pol := a_0 + a_1 × x; // n ≥ 1
2 macht := x;
3 for i := 2 to n do
4 |   macht := macht × x;
  |   // berekent x^2, x^3, ...
5 |   pol := pol + a_i × macht;
6 od
```

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

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

Horner

Algoritme 2: methode van Horner*

```
1 pol := a_n;
2 for i = n − 1 downto 0 do
3 |   pol := pol × x + a_i;
4 od
```

Gebaseerd op:

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

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

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

*gepopulariseerd door William Horner (1819); Chinese bronnen bekend uit de derde eeuw

Schema's

Algoritmen gebaseerd op het doen van vergelijkingen konden we beschrijven met **beslissingsbomen**.
Een model om rekenkundige algoritmen (algoritmen die zijn gebaseerd op $+$, $-$, $\times$ en $/$) 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 en x) of tussenresultaten van eerdere stappen
- de laatste stap uit het schema berekent het eindresultaat (hier dus $p(x)$)

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

Optimaliteit

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

Lemma. Een schema (dat alleen +, − en × 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 op 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 zo nodig de derde. We hebben zo een $(+,-)$-stap geëlimineerd uit het schema.
Gebruik nu inductie.

Optimaliteit — en verder

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 kan oplossen met minder vermenigvuldigingen en optellingen. De methode van Horner is derhalve optimaal.*

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

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

Met preprocessing

Polynoomevaluatie met preprocessing: bewerk het polynoom tot een polynoom in een speciale vorm waarop een nieuw evaluatiealgoritme 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 te evalueren polynoom monisch is. Hetzelfde geldt voor de aanname dat $n = 2^k - 1$.

Speciale vorm

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 Horner 7 vermenigvuldigingen en 7 optellingen, maar het kan met 5 en 10.

Constructie

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:

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

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 worden gebracht. Algoritme met complexiteit: zie opgave 48.

Evaluatie

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

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

Aantal vermenigvuldigingen

Zij $M(k)$ het aantal vermenigvuldigingen dat wordt gedaan 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: $M(k) = 2^{k-1} - 1 \rightarrow \frac{n-1}{2}$

Aantal optellingen

Zij $A(k)$ het aantal optellingen dat wordt gedaan 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: $A(k) = 3 \times 2^{k-1} - 2 \rightarrow \frac{3n-1}{2}$

Matrixvermenigvuldiging

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 $C = A \times B$ zijn dan gedefinieerd als volgt: $c_{ij} = \sum_{k=1}^{n} a_{ik} \times b_{kj}$.

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

```
1 for i := 1 to n do
2 |   for j := 1 to n do
3 |   |   c_ij := 0;
4 |   |   for k := 1 to n do
5 |   |   |   c_ij := c_ij + a_ik × b_kj;
6 |   |   od
7 |   od
8 od
```

Voorbeeld 2×2

Voorbeeld: product van twee 2×2-matrices algemeen.

$$\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix} = \begin{pmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{pmatrix}$$

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

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

Voorbeeld 2×2

We berekenen c_{21}

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

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

We kunnen c_{21} trouwens ook berekenen als:
$(a_{21} + a_{22}) \times b_{11} + a_{22} \times (b_{21} - b_{11})$. Dit lijkt weinig zinvol, maar toch...

Voorbeeld 2×2: aantal vermenigvuldigingen

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.

Recursief

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

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

Terzijde: de complexiteit van matrixvermenigvuldiging 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.

Recursief

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

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

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

Dit zijn 8 vermenigvuldigingen en 4 optellingen van $\frac{n}{2} \times \frac{n}{2}$-matrices. Voor $M(k)$, het aantal vermenigvuldigingen van elementen, geldt: $M(k) = 8M(k-1)$ en $M(0) = 1$, wat $M(k) = 8^k = n^3$ geeft. Hetzelfde als de "gewone" methode.

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

Strassen

Analoog aan het 2×2-geval kunnen we de berekening van de vier C_{ij}'s herschrijven zodat 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 zijn van twee deelmatrices van A en B:

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

Strassen

$P_1, \ldots, P_7$ 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:

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

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.

Strassen

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

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

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

Recurrente betrekking

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

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

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

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

Nog sneller

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

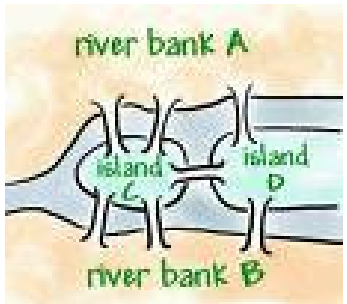
Het kan nog sneller!

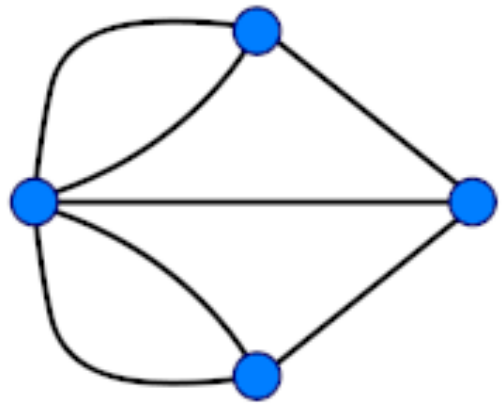
Pan (1978)	$\Theta(n^{2,796})$
Coppersmith-Winograd (1990)	$\Theta(n^{2,376})$
Stothers (2010)	$\Theta(n^{2,3737})$
Williams (2012)	$\Theta(n^{2,3729})$
Le Gall (2014)	$\Theta(n^{2,3728639})$

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 $\epsilon > 0$.

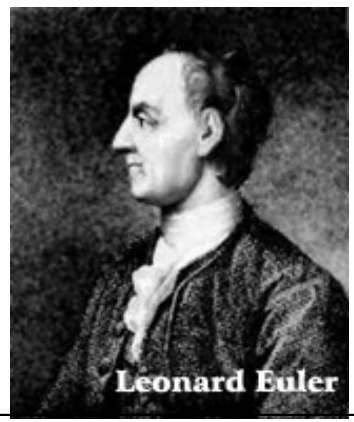
Euler

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 $\mathcal{G} = (V, E)$.

- 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 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† is een speciaal geval van een kring. Zo'n pad bestaat dus uit allemaal verschillende knopen.

*Foundations of Computer Science (FoCS): *circuit*

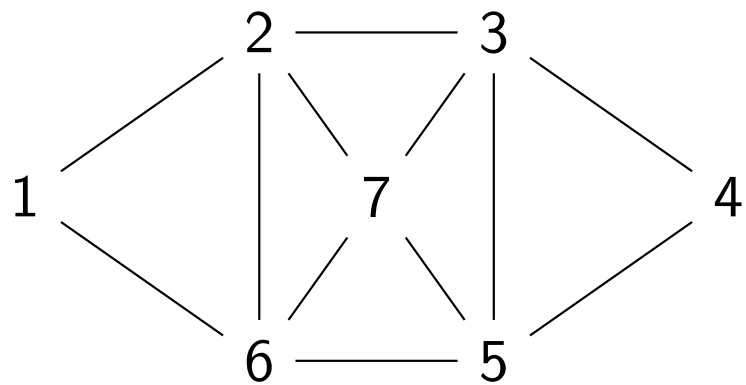
†FoCS: *cykel*

Eulerkringen

Gegeven een samenhangende, ongerichte graaf $\mathcal{G} = (V, E)$.

Definitie: een kring in $\mathcal{G}$ die alle takken van $\mathcal{G}$ 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 $\mathcal{G} = (V, E)$. Heeft $\mathcal{G}$ een Eulerkring?

Dit is een voorbeeld van een beslissingsprobleem: het antwoord is ja of nee.

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: $O(|E|)$.

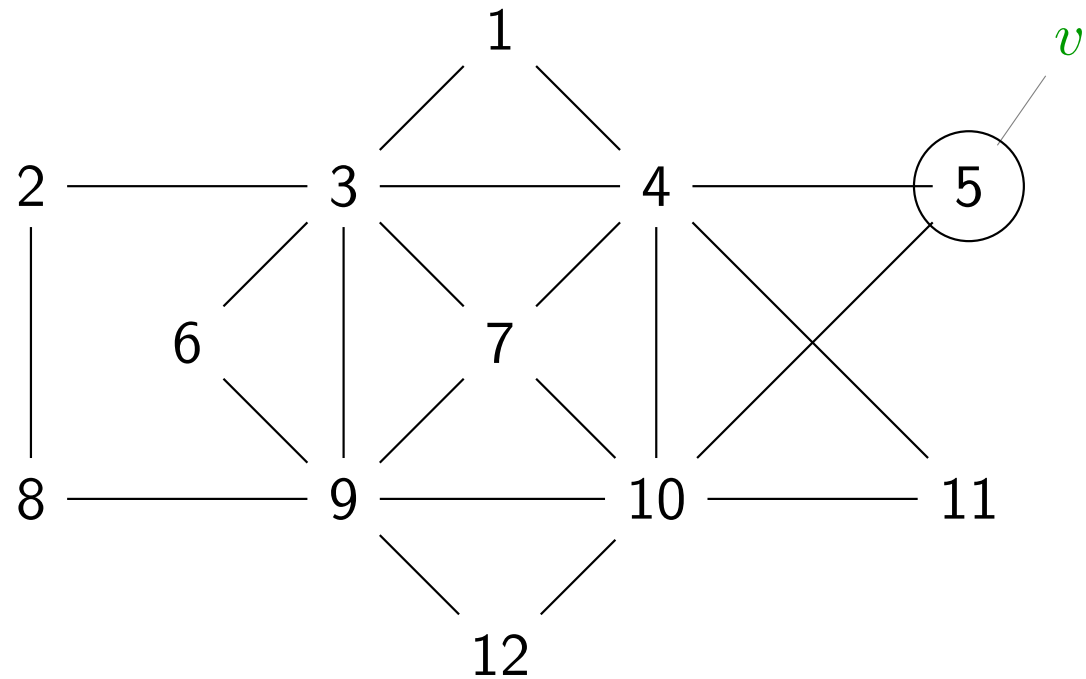
Opmerking
Het bewijs van "$\Leftarrow$" is constructief: het geeft je meteen een (overigens) $O(|E| + |V|)$ algoritme om zo'n Eulerkring te vinden.

Constructie

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
10 fi
```

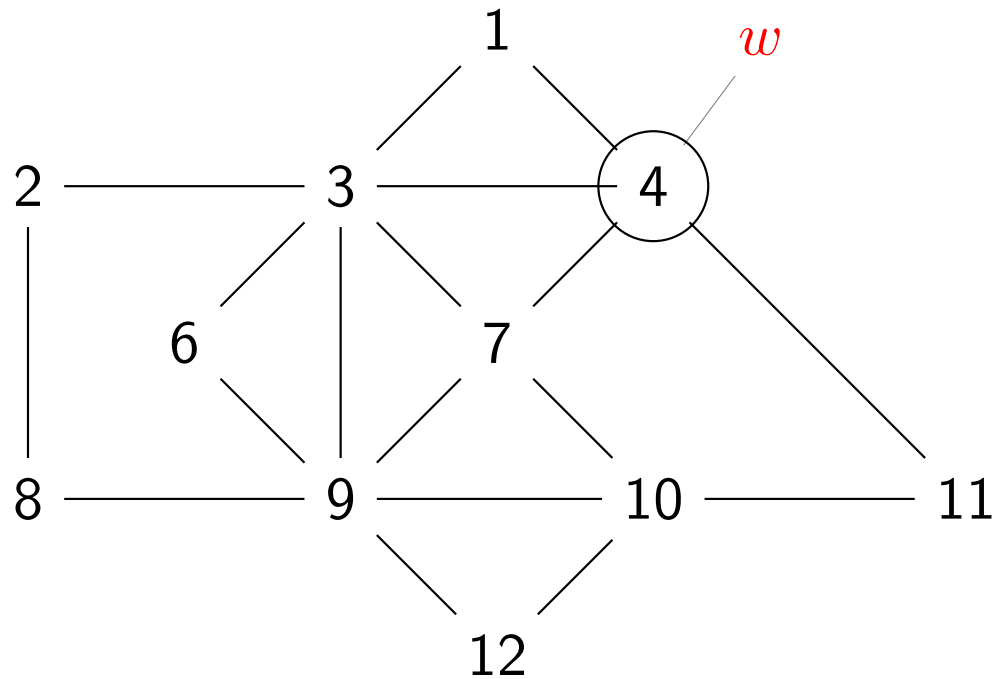
Opbouw Eulerkring



Kring vanuit v: 5, 4, 10, 5

$\mathcal{C}$ = 5, 4, 10, 5

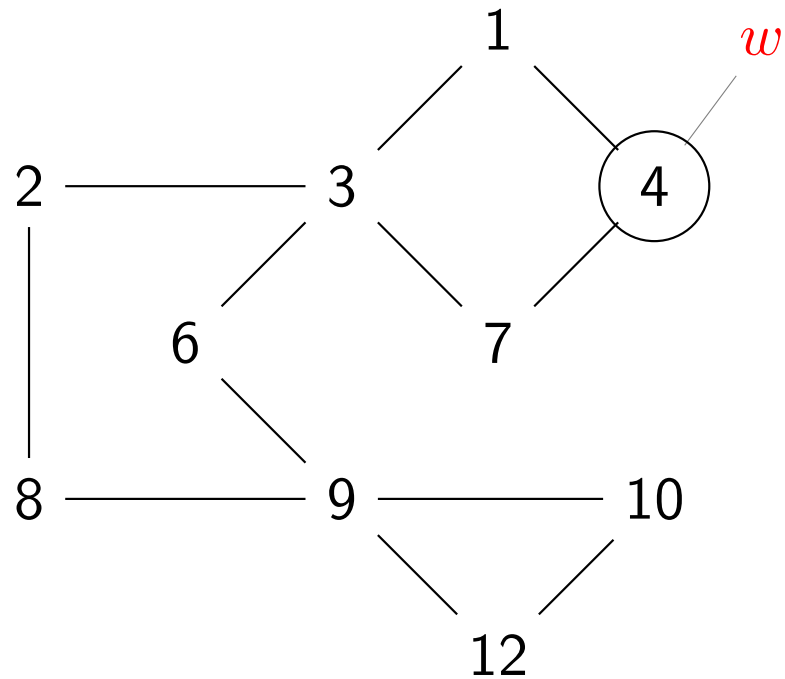
Opbouw Eulerkring



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

$\mathcal{C}$ = 5, 4, 11, 10, 7, 9, 3, 4, 10, 5

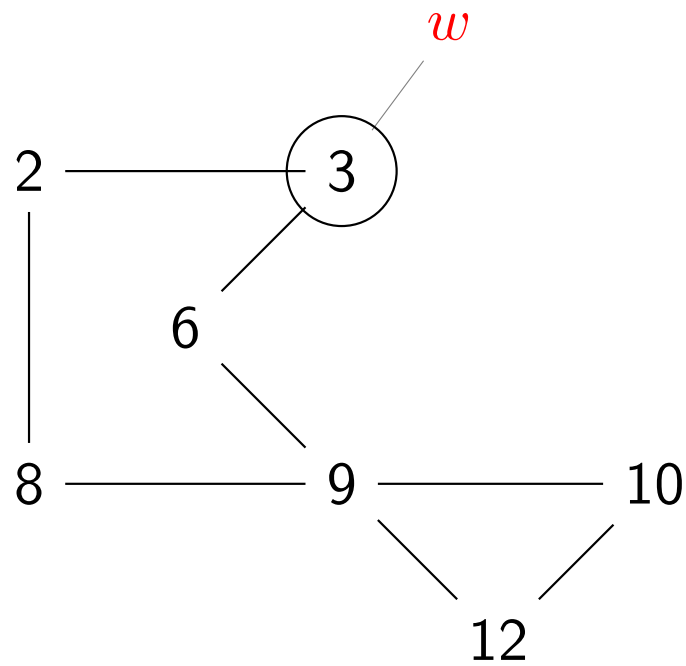
Opbouw Eulerkring



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

$\mathcal{C}$ = 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)

$\mathcal{C}$ = 5, 4, 1, 3, 2, 8, 9, 10, 12, 9, 6, 3, 7, 4, 11, 10, 7, 9, 3, 4, 10, 5

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 $O(|E|)$ stappen

- gebruik een kopie van $\mathcal{G}$ ($O(|V| + |E|)$ om die te maken).
- haal tijdens de constructie van $\mathcal{C}$ een tak weg (uit de kopie) zodra die is gebruikt.
- houd de kring $\mathcal{C}$ bij als enkelverbonden lijst, met een pointer naar de eerste knoop die nog onbewandelde takken heeft. We kiezen in stap 6 dus de eerste de beste knoop op $\mathcal{C}$ die nog onbewandelde takken heeft.

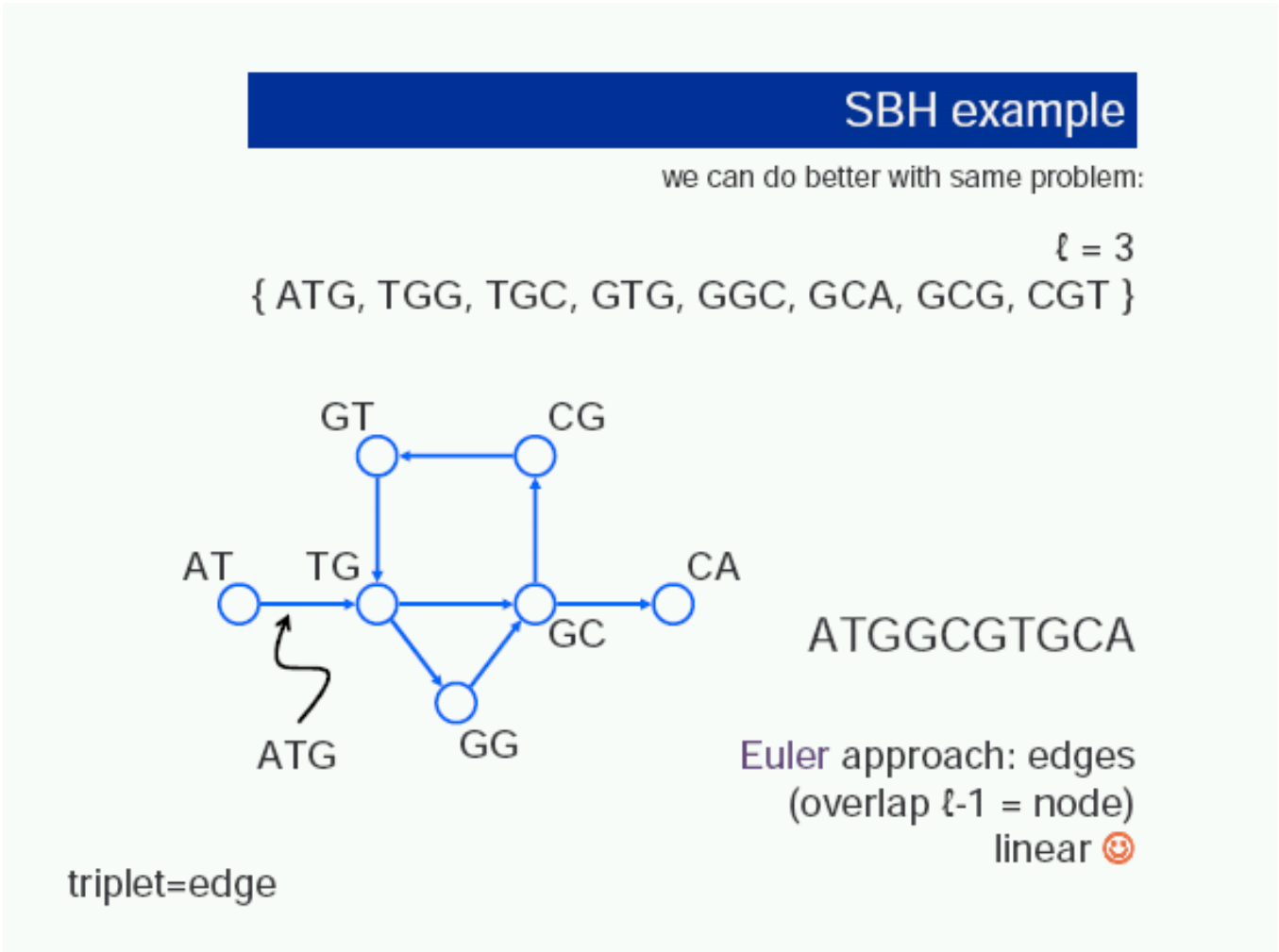
Opmerkingen

Opmerkingen bij implementatie en complexiteit:

- gebruik een adjacencylist als datastructuur en laat de buurlijsten oplopend zijn gesorteerd
- als een tak (v, w) is gebruikt, halen we alleen w uit de buurlijst van v maar niet andersom (dat komt later)
- de twee implementatie-opmerkingen hierboven zijn bedoeld om complexiteit $O(|E|)$ te verkrijgen i.p.v. $O(|E|^2)$
- zie opgave 50
- zie opgave 52 voor een alternatief (langzamer) algoritme (Fleury, 1883)
- om de complexiteit te beschrijven hebben we verschillende soorten operaties geteld en op één hoop gegooid

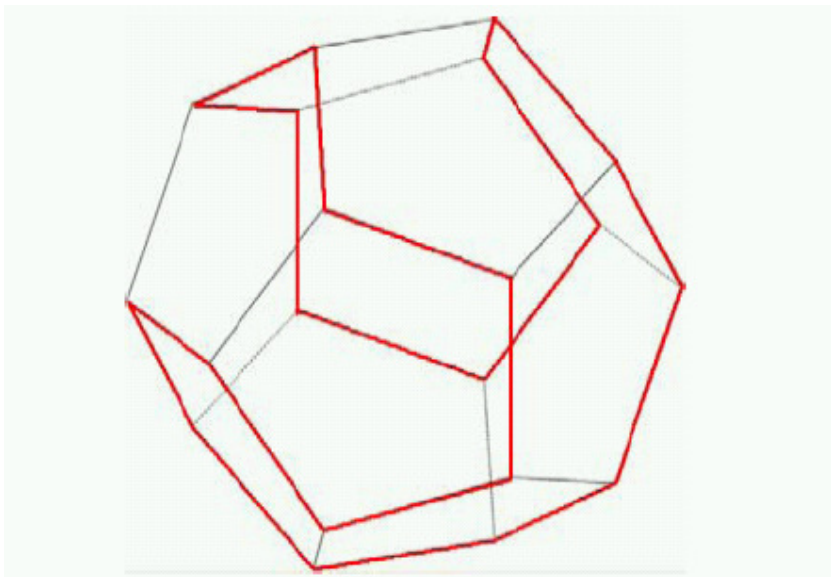
Toepassing (DNA)

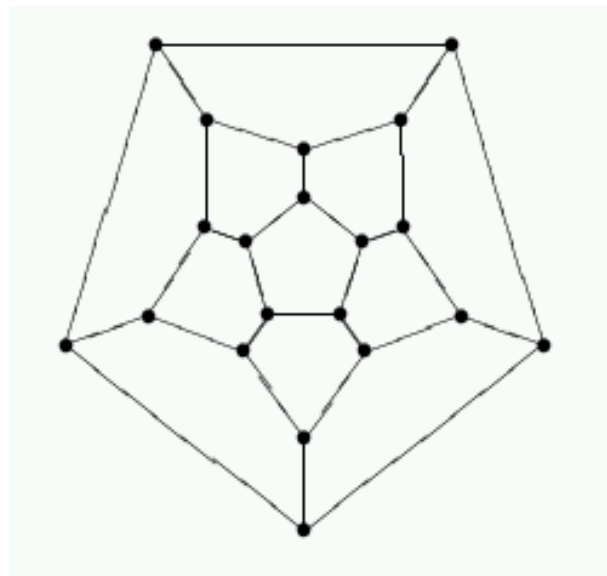
Uit: Molecular Computational Biology (oud mastercollege)



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 $\mathcal{G} = (V, E)$.

Definitie: een Hamiltonkring in $\mathcal{G}$ is een kring die elke knoop precies één keer bevat.

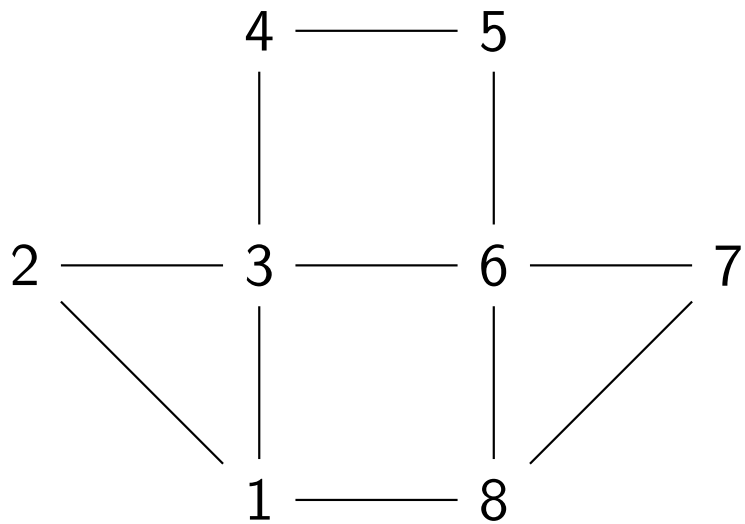
Bijbehorend beslissingsprobleem: Hamiltonkringprobleem (HC) voor ongerichte grafen*. Gegeven een ongerichte graaf $\mathcal{G} = (V, E)$. Heeft $\mathcal{G}$ een Hamiltonkring?

Naamgeving: als een graaf $\mathcal{G}$ een Hamiltonkring heeft, spreken we van een ja-instantie van het probleem. Als zo'n kring niet bestaat, heet $\mathcal{G}$ een nee-instantie.

*analoog voor gerichte grafen

Voorbeeld 1

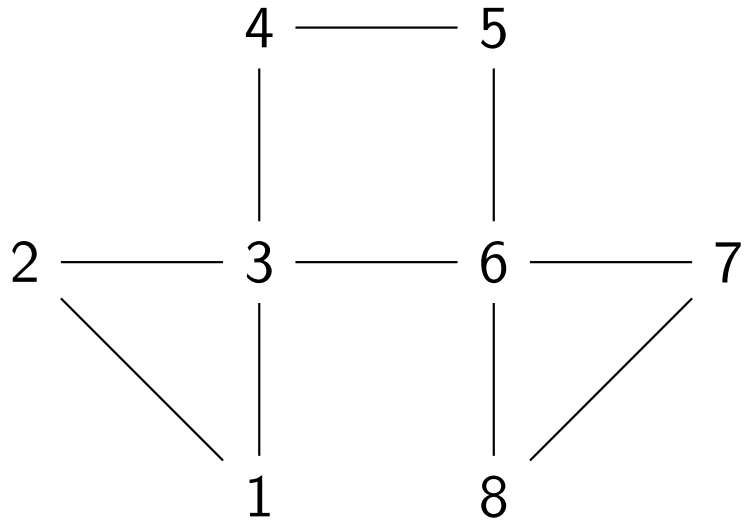
Voorbeeld 1:



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.

Toepassing

Vermomde Hamiltonproblemen:

1. 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?
2. 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?
3. (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

1. een (super)exponentieel algoritme is snel gevonden: genereer alle $n!$ mogelijke Hamiltonkringen (n is het aantal knopen van $\mathcal{G}$) en controleer voor elk of het een Hamiltonkring is.
2. het *controleren* of een kandidaat-Hamiltonkring echt een Hamiltonkring is, is *polynomiaal*.
3. het *vinden* van een Hamiltonkring daarentegen is/lijkt *héééééél moeilijk* (d.w.z. (super)exponentieel).
4. er is geen polynomiaal algoritme bekend voor dit probleem; zelfs niet voor het beslissingsprobleem HC, dat alleen vraagt *of* $\mathcal{G}$ een Hamiltonkring heeft.
5. er is ook niet bewezen dat een exponentieel algoritme *nodig* is om HC op te lossen.

Instanties

6. als $\mathcal{G}$ een ja-instantie is van HC, dus een Hamiltonkring heeft, is er een eenvoudige (d.w.z. polynomiale) manier om dat aan te tonen met behulp van de juiste hint (certificaat). Immers, in dit geval is het certificaat zo'n Hamiltonkring: controle of deze echt een Hamiltonkring is, is polynomiaal.
7. de enige manier om te laten zien dat $\mathcal{G}$ een nee-instantie is, dus geen Hamiltonkring bevat, lijkt: som alle $n!$ kandidaat-Hamiltonkringen op en laat zien dat ze alle geen Hamiltonkring zijn. Dat is (super)exponentieel.
8. er zijn eenvoudige en lastige instanties... → NP-volledigheid

Om te onthouden van vandaag

- Polynoomevaluatie
  - Methode van Horner
  - Met preprocessing en monische polynomen
- Matrixvermenigvuldiging
  - Strassen: $\Theta(n^{\lg 7})$
- Eulerkringen
  - d.e.s.d.a. graad van elke knoop even
  - Constructiealgoritme
- Hamiltonkringen
  - kandidaat-oplossing polynomiaal te controleren
  - gewoon oplossen vermoedelijk hééééééél moeilijk

(Werk)college

28 maart: géén hoor- of werkcollege

Volgende college: dinsdag 4 april, 9u00–10u45, zaal 412 (Snellius)

Werkcollege: zodadelijk van 11u00 tot 12u45, computerzalen 302–304 en 303 (Snellius)
Opgaven uit het dictaat: 49, 51, 52a, 47, 48

Huiswerk

Huiswerk 2

Inleveren op Brightspace, vóór maandagmiddag 27 maart. D.w.z.: uiterlijk maandag 27 maart 11u59. Daarna nog wel feedback maar geen cijfer.