Algoritmiek

voor Informatica

5 februari 2025

Introductie

- docent: Rudy van Vliet
  rvvliet@liacs.nl

- assistenten werkcollege en practicum

- website: `https://liacs.leidenuniv.nl/~vlietrvan1/algoritmiek/`
  voor: inhoud

- Brightspace, voor: groepjes voor opdrachten,
  inleveren opdrachten, cijfers, email(, Kaltura)

- college: woensdag 15.15–17.00

- werkcollege: vrijdag 11.00–12.45 (acht weken)

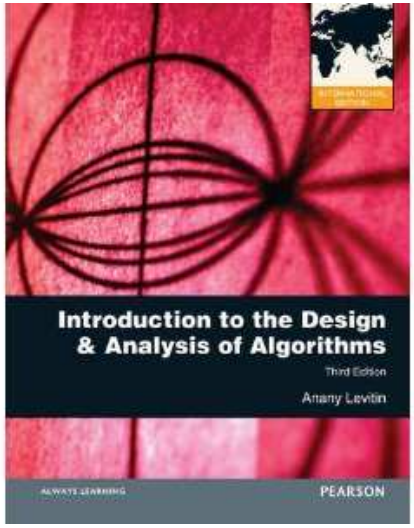
- practicumbijeenkomst: dinsdag 09.00–10.45 maar niet 11 februari

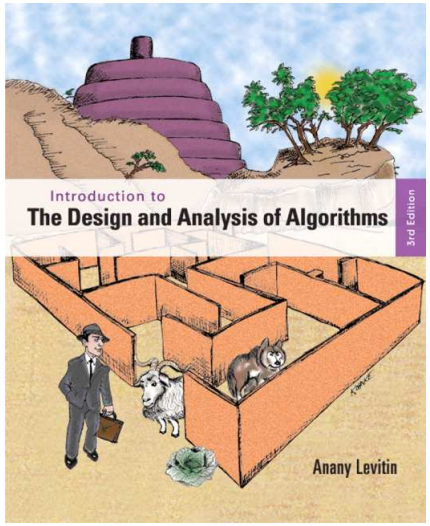
- tentamen: woensdag 21 mei 2025, 13.15–16.15

- hertentamen: maandag 30 juni 2025, 13.15–16.15

Bij dit college behorende boek:

Anany Levitin
Introduction to The Design and Analysis of Algorithms
third edition
Addison-Wesley, 2012





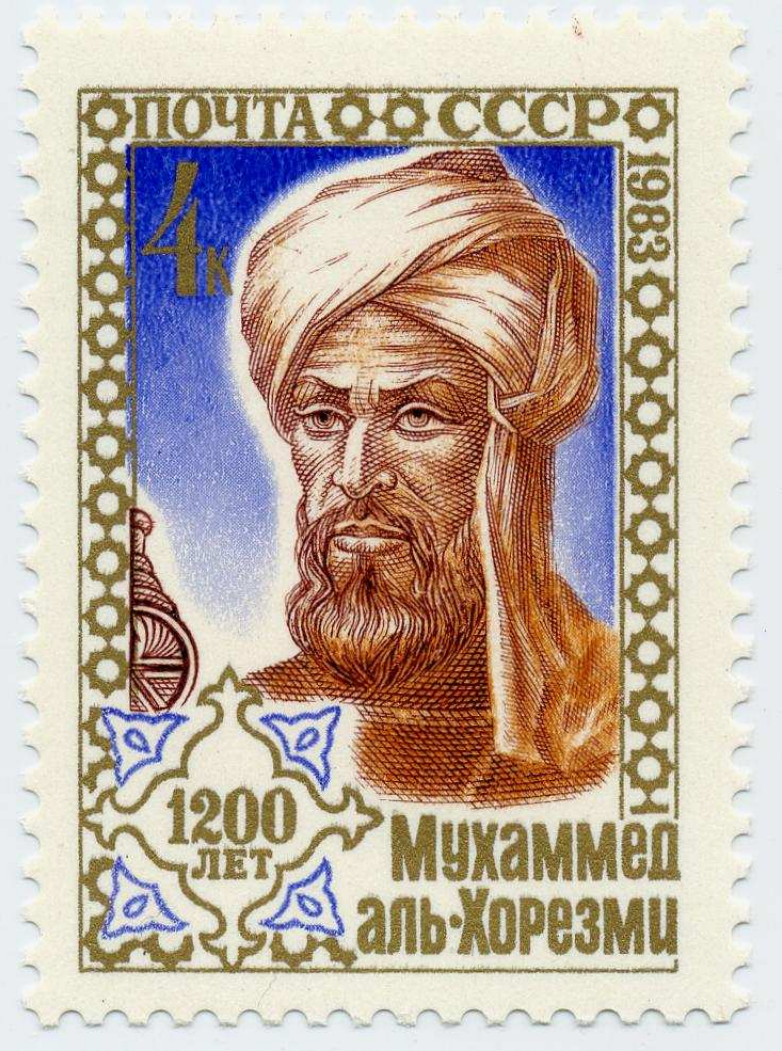
- drie programmeeropdrachten in C++
- nadruk op toepassen van besproken probleemoplossingsmethoden
- bij voorkeur in tweetallen (zoek je partner?)
- alle drie voldoende (≥ 5.5)
- `if (tentamen gehaald)`
  `if (programmeerwerk in orde)`
  `eindcijfer` $= \frac{2}{3}$ `tentamencijfer` $+\frac{1}{3}$ `practicumcijfer;`
  `else`
  `(nog) geen eindcijfer;`
  `else`
  `eindcijfer = tentamencijfer;`

- deadline
- twee weken uitloop, -1 pt per deel van week
- herkansingen aan eind semester, max $= 5.5$
- telaatenplagiaat
- ChatGPT

Een algoritme is een opeenvolging van **ondubbelzinnig** omschreven instructies die leiden tot een oplossing van een probleem. Het algoritme geeft voor elke willekeurige legale invoer van het probleem de vereiste uitvoer in een **eindig** aantal stappen (in eindige tijd).

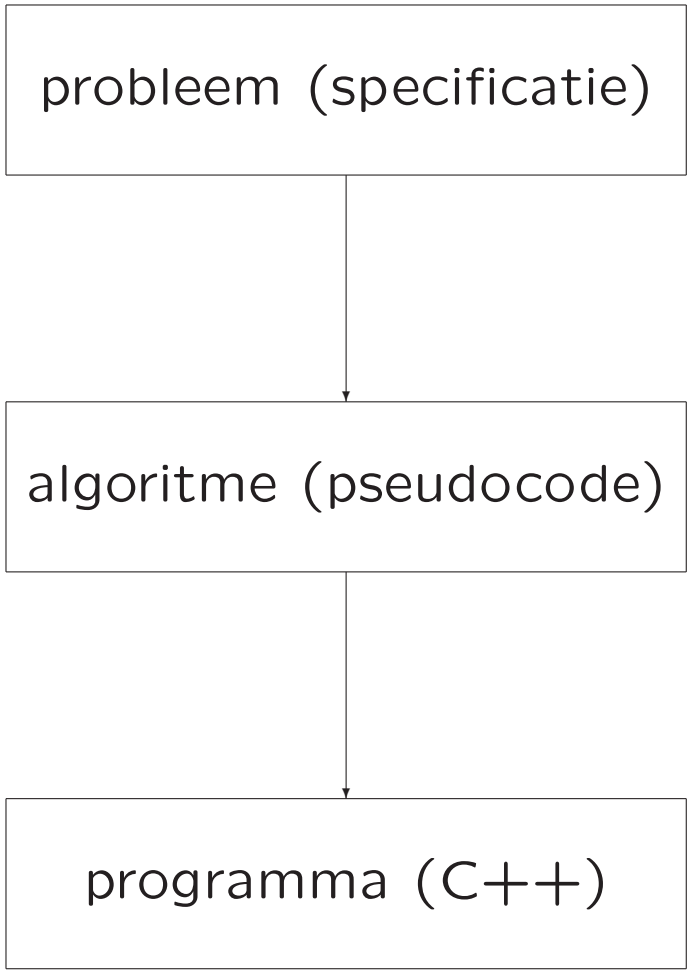
In de praktijk willen we dat een algoritme binnen redelijke tijd klaar is, en liefst zo snel mogelijk (**efficiëntie**): complexiteit van een algoritme.

Het woord algoritme is afgeleid van Abu Ja'far Muhammed ibn Musa al-Khwarizmi (ca. 780–850), een belangrijk Arabisch wiskundige uit de middeleeuwen.


ПОЧТА СССР
1983
4к
1200
ЛЕТ
Мухаммед
аль-Хорезми

Een algoritme is een recept, methode, ... voor het oplossen van een probleem, met de volgende eigenschappen:

1. **eindigheid**: stopt na een eindig aantal stappen
2. **bepaaldheid**: bestaat uit ondubbelzinnig omschreven stappen
3. **invoer**: het is duidelijk wat geldige invoer is
4. **uitvoer**: er kan bewezen worden dat de correcte uitvoer wordt gegeven voor elke geldige invoer



ontwerp

implementatie

- ontwerpen en analyseren van algoritmen: Algoritmiek, Datastructuren
  - efficiëntie
    . theoretische analyse: tijdcomplexiteit, ruimtecomplexiteit
    . empirische analyse: testen
- snel en correct: Voorbereiding Programmeerwedstrijden
  najaar 2025 weer
- correctheid bewijzen: Program Correctness
- optimaliteit, NP-volledigheid: Complexity
  - optimaliteit: bestaat er een beter algoritme?
  - NP-volledigheid: problemen die (waarschijnlijk) niet efficiënt opgelost kunnen worden

In dit college bekijken we enkele belangrijke ontwerptechnieken voor het algoritmisch oplossen van problemen:

* brute force/exhaustive search
* backtracking
* divide and conquer
* dynamic programming
* greedy approach
* branch and bound

Probleem: *Gegeven* twee niet-negatieve gehele getallen m en n (niet beide nul). *Vraag*: wat is de grootste gemeenschappelijke deler, genoteerd als ggd(m,n), van m en n?

Voorbeelden:
ggd(60,24) = 12
ggd(200, 441) = 1
ggd(588,495) = 3
ggd(25,0) = . . .

We bekijken drie algoritmes voor het berekenen van ggd(m,n).

Het volgende algoritme (genoteerd in gewone taal) berekent ggd(m,n) door van alle integers $\leq$ min(m,n) te proberen of ze m en n beide delen.

Algoritme 1:

Stap 1 Bepaal min(m,n) en ken de waarde toe aan t;
Stap 2 Deel m door t. Als de rest 0 is, ga naar Stap 3; anders ga naar Stap 4;
Stap 3 Deel n door t. Als de rest 0 is, dan is t de ggd, dus return t en stop; anders ga naar Stap 4;
Stap 4 Laag t met 1 af en ga naar Stap 2;

Merk op dat dit algoritme fout gaat als m of n nul is. Er moet hier dus gelden: $m \neq 0$ *en* $n \neq 0$.

Aantal stappen:
Best case. . .
Worst case . . .

Onderstaande methode vindt de ggd door m en n te ontbinden in hun priemfactoren. De ggd is dan het product van de gezamenlijke priemfactoren.

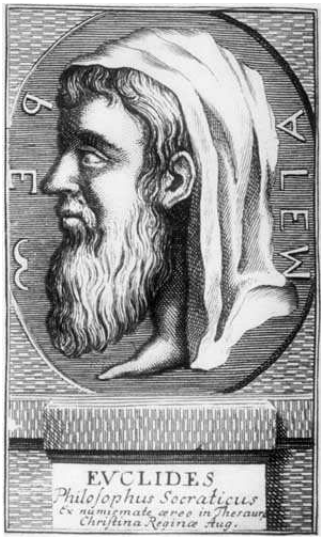
Voorbeeld: $60 = 2^2 * 3 * 5$ en $24 = 2^3 * 3$, dus $\text{ggd}(60, 24) = 2^2 * 3 = 12$.

Stap 1 Vind de priemontbinding van m;
Stap 2 Vind de priemontbinding van n;
Stap 3 Vind alle gemeenschappelijke priemfactoren;
Stap 4 Bereken het product van de gemeenschappelijke priemfactoren en return deze waarde;

Vraag: vinden we dit eigenlijk wel een algoritme?

Aantal stappen (voor priemontbinding / gemeenschappelijke factoren vinden):
Worst case: . . .
Best case: . . .

n	$\sqrt{n}$	$^{10}\log n$	$^{2}\log n$
10	3,16		
100	10,00		
1.000	31,62		
1.000.000	1.000,00		
1.000.000.000	31.622,78		



Euclides van Alexandrië (ca. 325 – 265 (voor Christus))

Prominent wiskundige uit de oudheid, vooral bekend vanwege zijn verhandeling over meetkunde: De Elementen.

Voorbeelden:
ggd(60,24) = 12
ggd(200, 441) = 1
ggd(588,495) = 3
ggd(25,0) = 25

Het algoritme van Euclides is gebaseerd op het herhaald gebruiken van de gelijkheid

$$\text{ggd}(m, n) = \text{ggd}(n, m \bmod n),$$

totdat de tweede parameter nul wordt.

Voorbeeld: ggd(60,24) = ggd(24, 12) = ggd(12, 0) = 12
ggd(588,495) = . . .

Het algoritme van Euclides in pseudocode (anders dan die in het boek wordt gebruikt):

```
while n ≠ 0 do
// elementaire operaties: deling en toekenning
    rest := m mod n;
    m := n;
    n := rest;
od
return m;
```

Zie ook dictaat en college Programmeermethoden.

Aantal stappen:
best case: . . .
worst case: . . .

Algoritme van Euclides: worst case

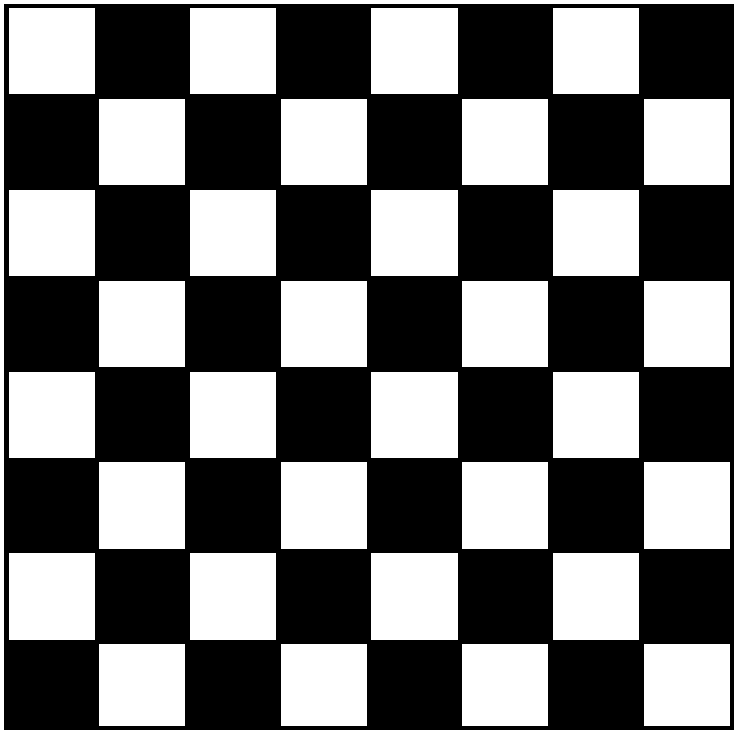
k	m	n
0	1	0
1	2	1
2	3	2
3	5	3
4	8	5
5	13	8
6	21	13
...	...	

Grofweg: $n \geq \text{fib}(k+1)$

Ofwel: $k \leq \ldots$

Andere bekende problemen die goed algoritmisch (al dan niet efficiënt) aan te pakken zijn:

- sorteren
- zoeken
- kortste pad in een graaf
- handelsreizigersprobleem
- vierkleurenprobleem
- convex hull probleem
- stabiele huwelijken
- vier-op-een-rij, schaken, go, . . . (?)
- dames op schaakbord
- torens van Hanoi
- en vele andere



Een dominosteen bedekt precies twee vakjes van het schaakbord. De ogen op de steen zijn voor dit probleem niet relevant. Dit geldt ook (meestal) voor de kleur van de velden van het bord.

Puzzel 1:

a. Kun je een 8 bij 8 (schaak)bord geheel volleggen met dominostenen? (En zo ja, op hoeveel manieren?)

b. Idem voor een 7 bij 7 bord

c. Idem voor een 8 bij 8 bord waarbij de velden linksboven en rechtsonder zijn weggelaten

d. Idem voor een 8 bij 8 schaakbord waarbij één wit veld en één zwart veld zijn weggelaten

Puzzel 2:

$$\begin{array}{r} \mathrm{D\ O\ N\ A\ L\ D} \\ \mathrm{G\ E\ R\ A\ L\ D} \\ \hline \mathrm{R\ O\ B\ E\ R\ T} \end{array} +$$

Elke letter stelt een cijfer voor $(0, 1, \ldots, 9)$ en verschillende letters corresponderen met verschillende cijfers. Het cijfer 0 komt niet voor als meest linkse cijfer in een getal. (Dus i.h.b. is $\mathrm{D} \neq 0$, $\mathrm{G} \neq 0$ en $\mathrm{R} \neq 0$.)

Om het makkelijker te maken is gegeven dat $\mathrm{D} = 5$.

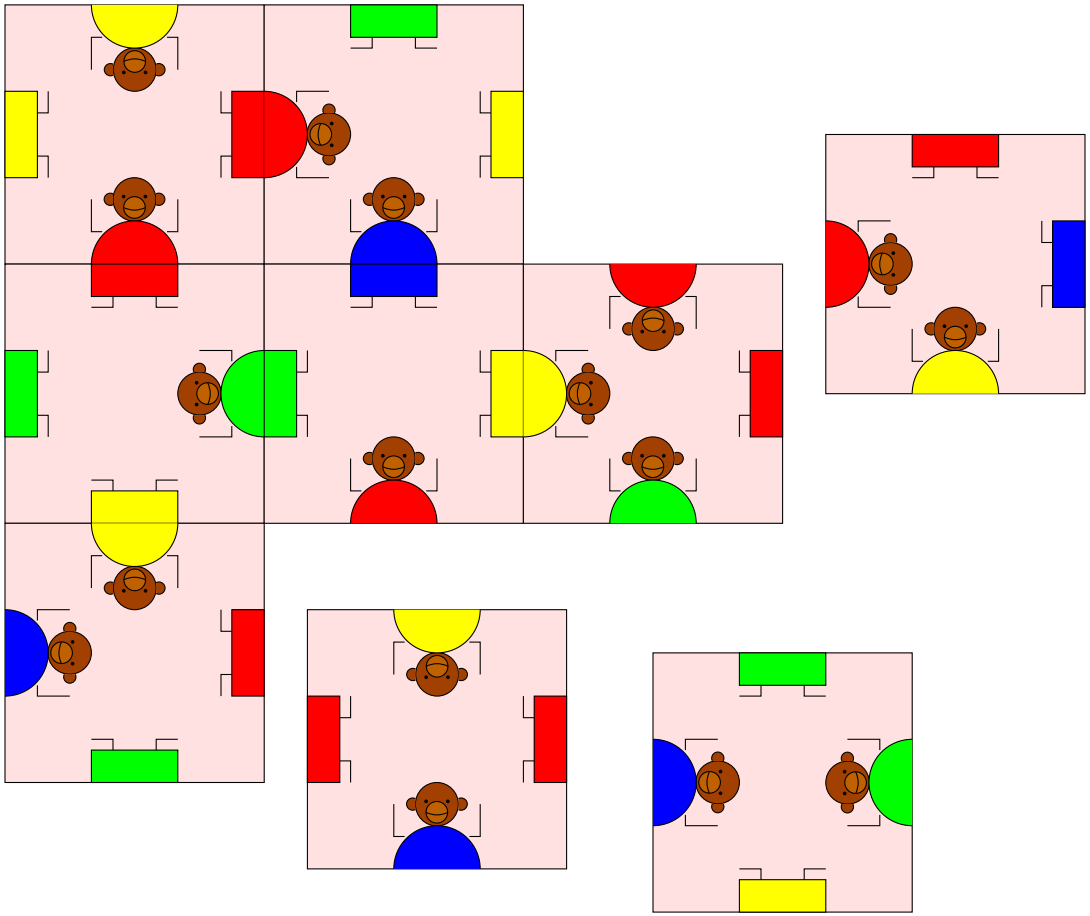
Vergelijk ook Levitin, hoofdstuk 3.4, opgave 11.

$$\begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \\ c_1 & c_2 & c_3 & c_4 & c_5 & \\ 5 & O & N & A & L & 5 \\ G & E & R & A & L & 5 \\ \hline R & O & B & E & R & T \end{array} +$$

Hierin is c_i de overdracht, dus $c_i = 0$ of 1 $(i = 1, \ldots 5)$

1. Uit kolom 6: $T = 0$ en $c_5 = 1$.
2. Uit kolom 5: $1 + 2L = R + 10 \cdot c_4 \Rightarrow$ R oneven.
3. Uit kolom 1: $c_1 + 5 + G = R \Rightarrow R \geq 5 + G > 5$,
   dus $R = 7$ of $R = 9$.
4. Uit kolom 2: $c_2 + O + E = O + 10 \cdot c_1 \Rightarrow c_2 + E = 10 \cdot c_1$.
   Als $c_1 = 0$, dan (c_2 en) $E = 0$, maar we hadden al $T = 0$.
   Gevolg: $c_1 = 1$, $E = 9$, $c_2 = 1$ en $R = 7$.
5. Doe de rest zelf.

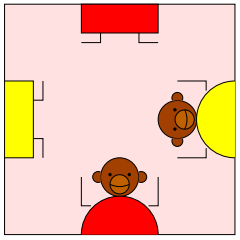
Puzzel 3: The Monkey puzzle



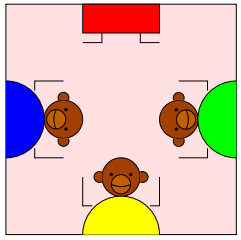
Monkey puzzle: leg de 9 stukken zo in een vierkant dat op alle aangrenzende stukken boven- en onderkant van aapjes van dezelfde kleur precies aansluiten.

Het is eenvoudig in te zien dat de puzzel van de vorige sheet opgelost kan worden. Dit is echter niet altijd het geval: het al dan niet oplosbaar zijn hangt van de stukken af.

Stel bijvoorbeeld dat we 9 dezelfde stukken hebben:



puzzel kan
opgelost worden



puzzel kan niet
opgelost worden

We bekijken de volgende fundamentele datastructuren:

- lineaire lijsten
- stapels en rijen
- priority queues
- grafen
- bomen

Zie ook: Programmeermethoden.

Een lineaire lijst is een lineair geordende, eindige collectie gegevens van hetzelfde type.

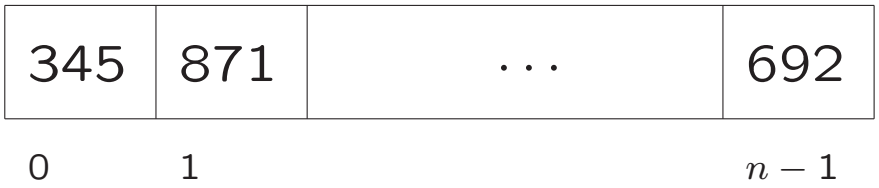
Implementatie:

- array / vector
- pointers
  - enkelverbonden lijst
  - dubbelverbonden lijst

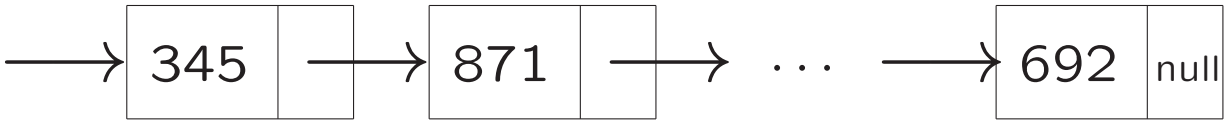
Speciale lijsten:

- stapel
- rij
- priority queue (niet per se lijst)

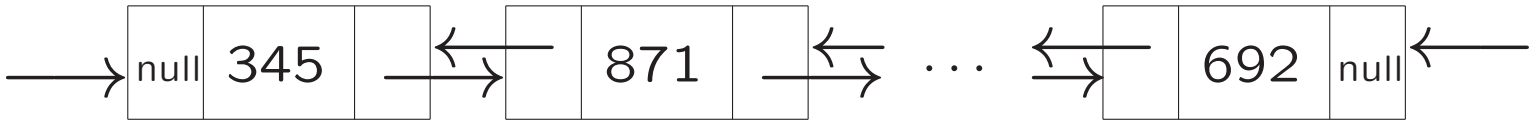
- array / vector



- enkelverbonden lijst



- dubbelverbonden lijst

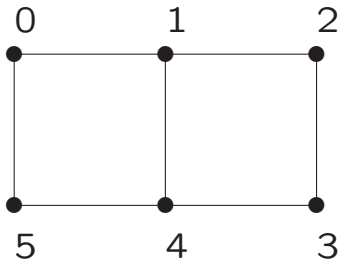


Een graaf G wordt gedefinieerd als een paar (V, E), waarbij V een eindige verzameling is van knopen (vertices) en E een verzameling van paren van knopen: de takken/pijlen (edges). Een tak/pijl (u, v) verbindt de knopen u en v met elkaar.

Terminologie:

- ongerichte/gerichte graaf
- gewogen graaf
- paden en kringen (cykels)
- samenhangend
- acyclisch

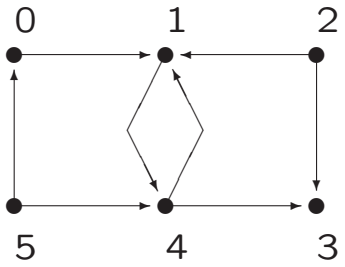
Grafen: voorbeelden



1.

$V = \{0, 1, 2, 3, 4, 5\}$;

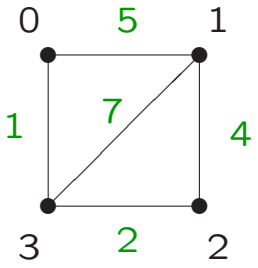
$E = \{(0, 1), (1, 2), (2, 3), (3, 4), (4, 5), (5, 0), (4, 1)\}$



2.

$V = \{0, 1, 2, 3, 4, 5\}$;

$E = \{(0, 1), (2, 1), (2, 3), (4, 3), (5, 4), (5, 0), (4, 1), (1, 4)\}$



3.

$V = \{0, 1, 2, 3\}$;

$E = \{(0, 1), (1, 2), (2, 3), (0, 3), (1, 3)\}$

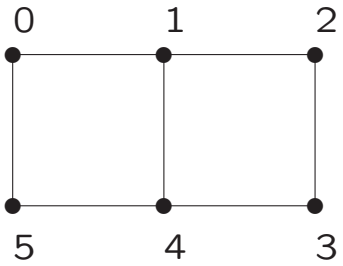
Adjacency matrix: de **ongerichte** graaf wordt gerepresenteerd door een tweedimensionaal array `int graaf[n][n]` (n het aantal knopen), waarbij `graaf`$[i][j]$ aangeeft of er een tak tussen i en j zit.

Adjacency list: de **ongerichte** graaf wordt gerepresenteerd door een eendimensionaal array `buur* graaf[n]` (n het aantal knopen), waarbij `graaf`$[i]$ de ingang is tot een lijst van buren van knoop i.

Adjacency list representatie in C++:

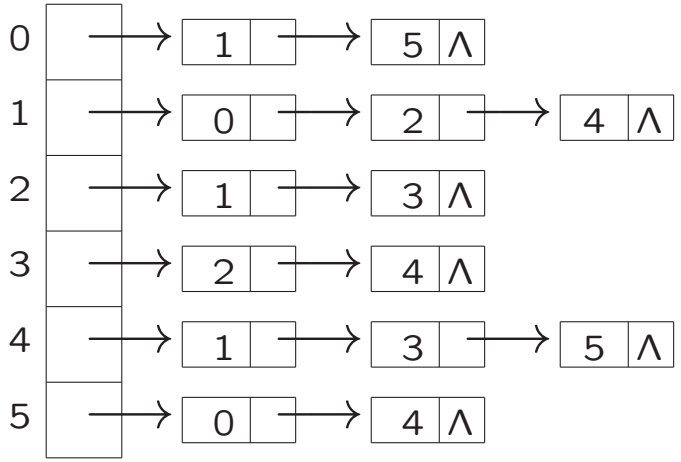
```
class buur
{ public:
    int knoopnummer;
    buur* volgende;
};  // buur
buur* graaf[n];
```

Adjacency matrix en adjacency list voor voorbeeldgraaf **1**:



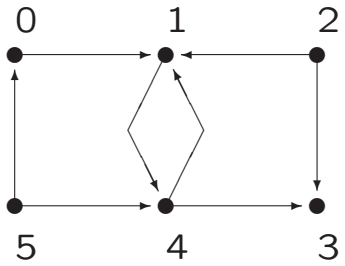
$$\begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

adjacency matrix



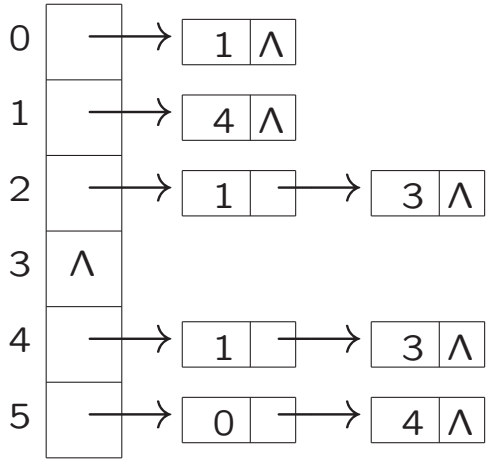
adjacency list

Adjacency matrix en adjacency list voor voorbeeldgraaf **2**:



$$\begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

adjacency matrix



adjacency list

- **Lezen/leren bij dit college:**
  Paragrafen 1.1, 1.2, 1.4 (t/m graafrepresentaties)

- **Werkcollege:**
  vrijdag 7 februari 2025, 11.00–12.45,
  zalen BW.0.05, BW.0.06, BW.0.07, BW.0.08

- **Opgaven:**
  zie `https://liacs.leidenuniv.nl/~vlietrvan1/algoritmiek/`

- **Volgend college:**
  woensdag 12 februari 2025, 15.15–17.00