

Eerste college algoritmiek

7 februari 2013

Introductie

1

- eerste college: 7 februari 2013
- eerste werkcollege: 7 februari 2013
- **tentamen**: dinsdag 11 juni 2013, 10.00–13.00
- **hertentamen**: woensdag 7 augustus 2013, 14.00–17.00

3

- drie programmeeropdrachten in C++
- nadruk op toepassen van besproken probleemoplossingsmethoden
- bij voorkeur in tweetallen
- alle drie voldoende (≥ 6)
- **if** (tentamen gehaald)
 - if (programmeerwerk in orde)
 - eindcijfer = $\frac{2}{3}$ tentamencijfer + $\frac{1}{3}$ practicumcijfer;
 - else
 - (nog) geen eindcijfer;
 - else
 - eindcijfer = tentamencijfer;

5



7

- docent: R. (Rudy) van Vliet
rvvliet@liacs.nl
- werkgroep leider: Jan van Rijn
jvrjln@liacs.nl

• website: www.liacs.nl/home/rvvliet/algoritmiek/

- **college**: donderdag 11.15 – 13.00 in zaal Noordende
- **werkcollege**: donderdag 13:45 – 15:30 in zaal Noord-einde (of Palestuin)

2

Bij dit college behorende **boek**:

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



4

Een **algoritme** is een opeenvolging van **onduidelijk** 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).

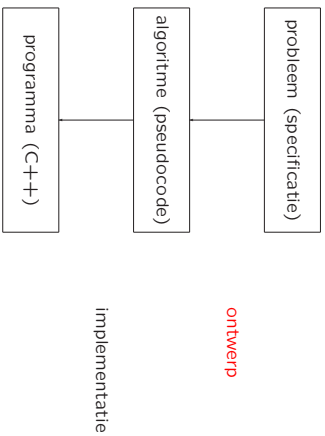
In de praktijk willen we dat een algoritme **effectief** is, dat wil zeggen binnen redelijke tijd klaar is, en liefst zo snel mogelijk (**efficiëntie**): **complexiteit** van een algoritme.

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

6

1. **eindigheid**: stopt na een eindig aantal stappen
2. **bepaaldheid**: bestaat uit onduidelijk 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
5. **effectiviteit**: de stappen zijn voldoende eenvoudig en het antwoord wordt binnen redelijke tijd gegeven

8



9

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

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

11



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

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

13

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

Algoritme 2:

- 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 –in tegenstelling tot het algoritme van Euclides– dit algoritme fout gaat als m of n nul is. Er moet hier dus gelden: $m \neq 0$ en $n \neq 0$.

15

- ontwerpen van algoritmen: [Algoritmiek](#), [Datastructuren](#)
- hoe beschrijven we het algoritme?
- correctheid bewijzen: [Programmeren & correctheid](#)
- analyse van algoritmen: [Complexiteit](#)
 - efficiëntie
 - . theoretische analyse: tijdcomplexiteit, ruimtecomplexiteit
 - . empirische analyse: testen
 - optimaliteit: bestaat er een beter algoritme?

10

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

Voorbeeld: $\text{ggd}(60,24) = 12$; $\text{ggd}(25,0) = 25$;
 $\text{ggd}(200, 441) = 1$; $\text{ggd}(588,495) = 3$.

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: $\text{ggd}(60,24) = \text{ggd}(24, 12) = \text{ggd}(12, 0) = 12$

12

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](#).

Er zijn nog andere algoritmen voor het berekenen van de ggd van twee getallen. We bekijken er twee. Welke van de drie is het efficiëntst?

14

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;

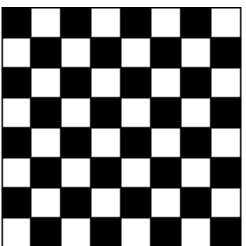
Vraag: vinden we dit eigenlijk wel een algoritme?

16

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

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

17



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.

19

$$\begin{array}{r}
 1\ 2\ 3\ 4\ 5\ 6 \\
 c_1\ c_2\ c_3\ c_4\ c_5 \\
 5\ 0\ N\ A\ L\ 5 \\
 G\ E\ R\ A\ L\ 5 \\
 \hline
 R\ O\ B\ E\ R\ T
 \end{array}
 +$$

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

1. Uit kolom 6: $T = 0$ en $c_6 = 1$.
2. Uit kolom 5: $1 + 2L = R + 10c_4 \Rightarrow R$ onteven.
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: $c_2 + O + E = O + 10c_1 \Rightarrow c_2 + E = 10c_1$. 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.

21

Monkey puzzel: leg de 9 stukken zo in een vierkant dat op alle aangrenzende stukken boven- en onderkant van apples 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

23

Puzzel 1:

- Kun je een 8 bij 8 (schaak)bord geheel volleggen met dominostenen? (En zo ja, op hoeveel manieren?)
- Idem voor een 7 bij 7 bord
- Idem voor een 8 bij 8 bord waarbij de velden linksboven en rechtsonder zijn weggelaten
- Idem voor een 8 bij 8 schaakbord waarbij één wit veld en één zwart veld zijn weggelaten

18

Puzzel 2:

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

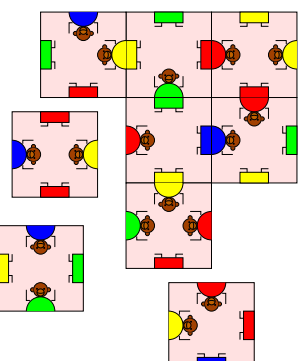
Elke letter staat een cijfer voor (0, 1, ..., 9) en verschillende letters corresponderen met verschillende cijfers. Het cijfer 0 komt niet voor als meest linkse cijfer in een getal. (Dus i, h. b. is $D \neq 0$, $G \neq 0$ en $R \neq 0$.)

Om het makkelijker te maken is gegeven dat $D = 5$.

Vergelijk ook Levitin, hoofdstuk 3.4, opgave 11.

20

Puzzel 3: The Monkey puzzle



22

We bekijken de volgende **fundamentele datastructuren**:

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

Zie ook: [Programmeermethoden](#).

24

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

Implementatie:

- array
- pointers
 - enkelverbonden lijst
 - dubbelverbonden lijst

Speciale lijsten:

- stapel
- rij
- priority queue

25

Een **graaf** G wordt gedefinieerd als een paar (V, E) , waarbij V een eindige verzameling is van **knoppen** (vertices) en E een verzameling van paren van knoppen: 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 cycli
- samenhangend
- acyclisch

27

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

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

Adjacency list representatie in C++:

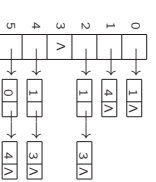
```
struct buur {
    int knooppuntnr;
    buur* volgende;
}
buur* inhoud[n];
```

29

Adjacency matrix en adjacency list voor voorbeeldgraaf 2:

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

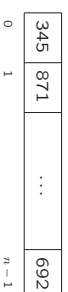
adjacency matrix



adjacency list

31

- array



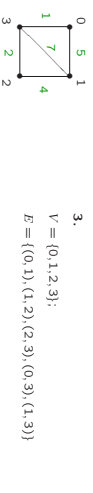
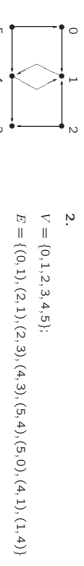
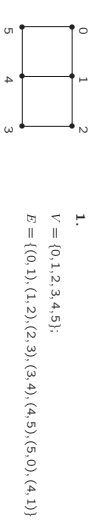
- enkelverbonden lijst



- dubbelverbonden lijst



26

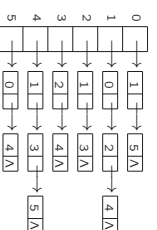


28

Adjacency matrix en adjacency list voor voorbeeldgraaf 1:

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

adjacency matrix



adjacency list

30

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

- **Werkcollege:** donderdag 7 februari 2013, 13:45–15:30, zaal Noord-einde

- **Opgaven:** zie <http://www.liacs.nl/home/vvliet/algoritmiek/>
- **Volgend college:** donderdag 14 februari 2013, 11:15–13:00, zaal Noord-einde

32