PROGRAMMEREN 2

TOPIC 6: ALGORITMES ONTWERPEN

UCLL HOGESCHOOL

OVERZICHT

- Wat zijn algoritmes?
- Algoritme van Euclides
- Algoritme voor doolhoven
- Informatica in context: doel en domeinen

TERUGKOPPELING NAAR VORIGE LESSEN

ALGORITMES

Zelfde vraag als vorige les: hoe zou je het concept 'algoritme' uitleggen?

ALGORITMES

Zelfde vraag als vorige les: hoe zou je het concept 'algoritme' uitleggen?

Een algoritme is een voorschrift om in een eindig aantal stappen een welbepaalde taak uit te voeren.

RESULTATEN

Wat zijn de resultaten van de volgende algoritmes?

- Cijferen (bv. een deling van twee grote getallen)
- Minimax-algoritme in een schaakprogramma
- Een GPS de route naar de campus laten berekenen
- Een neuraal netwerk initialiseren met willekeurige gewichten, en dan laten trainen op invoer

RESULTATEN: ALGEMEEN

Wat is het resultaat van een algoritme in het algemeen?

RESULTATEN: ALGEMEEN

Wat is het resultaat van een algoritme in het algemeen?

Door de uitvoering van een algoritme wordt een gegeven beginsituatie getransformeerd tot een beoogde eindsituatie.

RESULTATEN: ALGEMEEN

Wat is het resultaat van een algoritme in het algemeen?

Door de uitvoering van een algoritme wordt een gegeven beginsituatie getransformeerd tot een beoogde eindsituatie.

Opgelet: *transformeren* betekent niet noodzakelijk dat de beginsituatie altijd overschreven wordt. Denk bv. aan cijferen in de lagere school.

HET BEGRIP 'ALGORITME'

ETYMOLOGIE (OORSPRONG)

Arabisch: al-Ḳwārizmī → bijnaam van de wiskundige Abū Ja‘far Muhammad ibn Mūsa

Grieks: arithmos → nummer

Middeleeuws Europa schakelde over naar Arabisch talstelsel → 'algoritme' als rekenprocedure

17de eeuw: Leibniz veralgemeent de term 'algoritme

UITVOEREN VS OPSTELLEN

Algoritmes **uitvoeren** is 'eenvoudig'.

Algoritmes **opstellen** is moeilijk. Het vereist creativiteit en analytisch denken.

In beide gevallen is accuraatheid belangrijk: geen ruimte voor vrije interpretatie.

Computers hebben een voordeel qua uitvoering: ze volgen exact instructies en maken geen fouten.

EIGENSCHAPPEN

Een goed algoritme is:

- Voldoende algemeen
- Ondubbelzinnig:
  - De basistaken zijn simpel genoeg
  - Duidelijke volgorde (cfr. controlestructuren)
  - Duidelijke begin- en eindsituatie
- Altijd eindigend
- Correct: het doet wat gevraagd is
- Begrijpbaar (voor mensen)
- (Efficiënt)

EVALUEER

Vergelijk de kenmerken van een goed algoritme met enkele algoritmes uit vorige lessen:

1. Kaarten sorteren door ze telkens willekeurig omhoog te gooien en op te rapen
2. Minimax-algoritme
3. Bubble sort
4. Neuraal netwerk trainen

Let zeker op **eindigheid**, **correctheid**, **begrijpbaarheid** en **efficiëntie**.

ZELF OPSTELLEN: VOORBEELD 1

ALGORITME VAN EUCLIDES

ALGORITME VAN EUCLIDES

Dit algoritme bepaalt de grootste gemene deler (GGD) van twee getallen. Het werd in circa. 300 v.Chr. beschreven door Euclides.

Doel van de les: zelf het algoritme proberen af te leiden.

VOORBEELDEN

Probeer met de hand de GGD van volgende getallen te vinden:

1. 25 en 5
2. 50 en 20
3. 180 en 120

Denk telkens goed na wat je precies doet. Probeer het proces in je hoofd expliciet te maken in de vorm van een stappenplan in mensentaal. Schrijf het op op papier of digitaal.

Test je algoritme uit door het aan een medestudent te geven en hen de GGD te laten bepalen van volgende getallen:

1. 210 en 45
2. 252 en 105

OPLOSSING IN MENSENTAAL

Invoer: 2 positieve gehele getallen `x` en `y`.

1. Bepaal de rest `r` van `x / y`
2. Als `r = 0`, geef dan `y` als resultaat en stop
3. Als `r ≠ 0`, vervang `x` door `y` en `y` door `r` en ga terug naar stap 1

Uitvoer: een positief getal dat de GGD is van `x` en `y`.

Is dit algoritme eindig?

Is het correct en efficiënt? → moeilijker aan te tonen

PROGRAMMEREN

Probeer dit algoritme nu te programmeren in Python:

- Schrijf het algoritme als een functie die je kan oproepen
- Je mag recursie gebruiken (cfr. les rond AI), maar dit hoeft niet
- Welke andere programmeerconcepten ga je nodig hebben? En welke wiskundeconcepten?
- Schrijf de voorbeelden uit vorige slides op als testen door middel van `assert`. Controleer dat je code slaagt op alle testen

> **i** Geen idee (meer) hoe `assert` werkt? Zoek zelf online op! Probeer een antwoord te vinden binnen 120s.

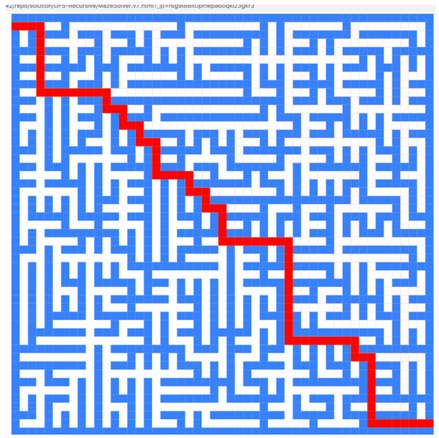
ZELF OPSTELLEN: VOORBEELD 2

DOOLHOVEN DOORKRUISEN

PADEN EN DOOLHOVEN

Er zijn verschillende algoritmes om paden te vinden doorheen open ruimtes, doolhoven, wegennetwerken, etc. Sommige zoeken enkel naar *een* pad, andere zoeken specifiek naar het kortste pad.

We focussen hier op het maken van een algoritme om *een* pad te vinden in een doolhof. Geïnspireerd op de Griekse mythe van Theseus, Ariadne en de minotaurus, nemen we een lange draad en een pot verf mee.



Link

ALGORITMES OPSTELLEN UIT VOORBEELDEN

Strategie: door verschillende voorbeelden van doolhoven ten analyseren en het pad systematisch te zoeken, kunnen we patronen herkennen in wat we doen.

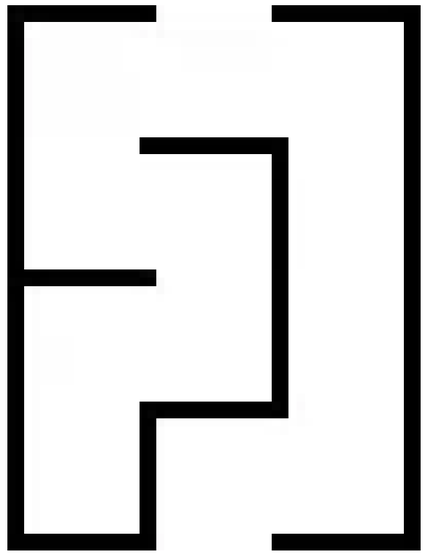
Nadien kunnen we de patronen omvormen tot een degelijk algoritme.

Gebruik de draad en de verf slim:

- De draad stelt het afgelegde pad voor
- De verf stelt een markering voor → wat zou je op de grond/muur kunnen tekenen?

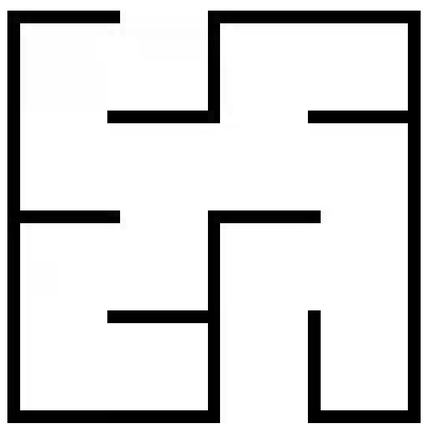
VOORBEELD 1

Zoek het pad op een systematische manier. Kan je een stap/instructie/regel afleiden voor het algoritme?



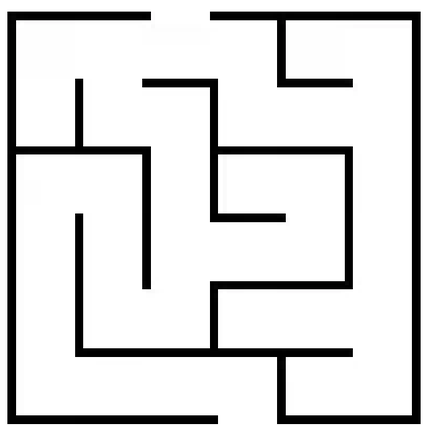
VOORBEELD 2

Zoek het pad op een systematische manier. Kan je een stap/instructie/regel afleiden voor het algoritme?



VOORBEELD 3

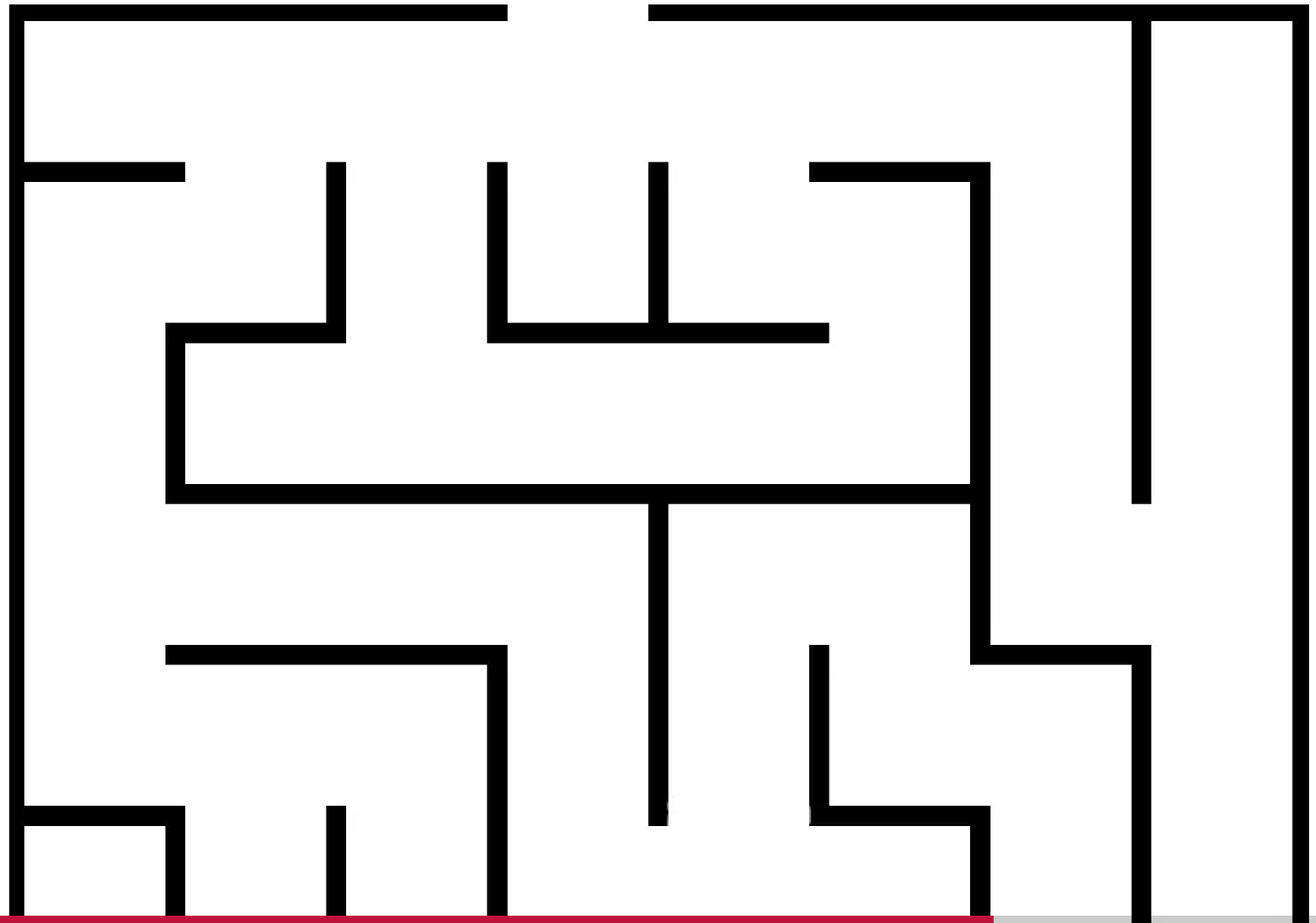
Zoek het pad op een systematische manier. Kan je een stap/instructie/regel afleiden voor het algoritme?



VOORBEELD 4

Opgelet: wat is hier anders dan in de vorige voorbeelden?

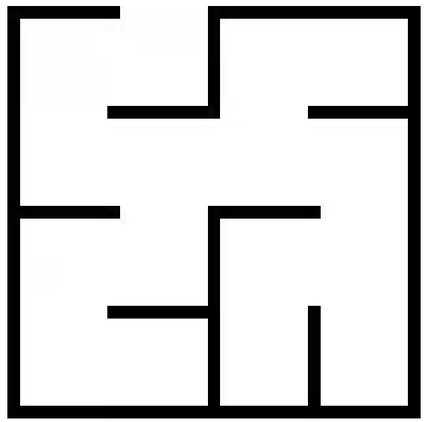
Zoek het pad op een systematische manier. Kan je een stap/instructie/regel afleiden voor het algoritme?



VOORBEELD 5

Opgelet: wat is hier anders dan in de vorige voorbeelden?

Zoek het pad op een systematische manier. Kan je een stap/instructie/regel afleiden voor het algoritme?



MEER VOORBEELDEN

Nog meer voorbeelden nodig om ideëe op te testen?

- https://www.mazegenerator.net/
- https://www.ohmydots.com/creator-maze.html

ALGORITME IN MENSENTAAL

Invoer: een doolhof, een draad, een pot verf en een borstel.

Uitvoer: een pad doorheen het doolhof (de draad) OF er is geen pad.

ALGORITME IN MENSENTAAL

1. Maak de draad bij de ingang vast en neem alles mee
2. Sta je aan de uitgang?
    1. Indien JA: de draad toont het pad doorheen het doolhof. STOP
    2. Anders: ga naar 3
3. Loopt er een draad over dit kruispunt?
    1. Indien JA: keer terug naar het vorige kruispunt, rol de draad op, en markeer de doorgang. Ga naar 2
    2. Anders: ga naar 4
4. Vertrekt er vanaf dit kruispunt een verbinding zonder draad of verf?
    1. Indien JA: kies er één uit, stap naar het volgende kruispunt en rol de draad af. Ga naar 2
    2. Anders: ga naar 5
5. Sta je terug bij de ingang?
    1. Indien JA: er is geen weg door de doolhof, STOP
    2. Anders: keer terug naar het vorige kruispunt, rol de draad op en markeer de doorgang. Ga naar 2

BEMERKINGEN

- Sommige lijnen zijn overbodig. Welke?
- Eindigheid is vrij makkelijk te bewijzen
- Correctheid is veel moeilijker te bewijzen
- **Enkel testen zijn niet genoeg om eindigheid en correctheid te bewijzen**
- Efficiëntie van algoritmes bespreken we later meer in detail

PROGRAMMEREN

We gaan dit algoritme niet vertalen naar Python tijdens de les. Je kan dit wel doen als thuisopdracht voor je portfolio. Iets om bij stil te staan:

- Hoe stel je een doolhof voor? Welke datastructuur gebruik je?
- Hoe stel je de draad en de verf voor?

CONTEXT

INFORMATICA, DISCIPLINES EN PROGRAMMEERTALEN

PROGRAMMA'S EN PROGRAMMEERTALEN

Om algoritmen aan een computer duidelijk te maken, is een geformaliseerde notatie nodig. Mensentaal is te ambigu; er zijn teveel interpretaties mogelijk

Een **programma** bevat de beschrijvingen van algoritmen zoals ze door een machine moeten worden uitgevoerd en is geschreven in een programmeertaal.

Een **programmeertaal** is een geformaliseerde notatie voor programma's.

Informatica draait rond het **bestuderen** en **opstellen (maken)** van algoritmes, om ze uiteindelijk door een computer te laten uitvoeren. Programmeertalen vormen de schakel tussen menselijke concepten en het binaire denken van computers.

DOMEINEN BINNEN INFORMATICA

De meest populaire domeinen:

- **Algoritmiek**: eigenschappen van algoritmes en problemen onderzoeken
- **Software engineering**: grotere systemen opbouwen door kleinere systemen te combineren
- **Gegevensmodellering, gegevensbeheer, datastructuren**
- **Artificiële Intelligentie**: algoritmes die intelligent gedrag simuleren
- **Mens-machine interactie**: software ontwikkelen vanuit het standpunt van eindgebruikers, Augmented Reality, Virtual Reality
- **Netwerken**: computers laten communiceren en samenwerken in groep
- **Veilige software**: systemen beveiligen tegen slechte actoren
- **Computer Graphics**: beelden genereren (filmindustrie, videogames, simulaties, ...)
- **Programmeertalen**: talen ontwerpen en implementeren

DOMEINEN OVER INFORMATICA

Het domein informatica wordt hier bekeken vanop afstand:

- Vakdidactiek
- Social Informatics
- Computer ethics
- technologiefilosofie

DOMEINEN VOOR INFORMATICA

Kennis binnen deze domeinen helpt bij het begrijpen van concepten, regels, eigenaardigheden, ... binnen informatica:

- Numerieke wiskunde
- Statistiek
- Elektronica
- ...

Informatica en *Computerwetenschappen* betekenen hetzelfde. Informatica wordt vooral gebruikt in het Nederlands en is geïnspeerd op de Franse benaming. De rest van de wereld spreekt voornamelijk over *Computer science* of *Computing science*.