Bestand waar Thijs in typt (wegens GitHub)

* Hierin staat een tijdelijke bronnenlijst. Deze moet later in het grote verslag geïmplementeerd worden.
* Waar \*\*\*\* staat, moet later naar gekeken worden. Dit kan staan bij een twijfel in de formulering of bij de nummering van een bron, etc.

# Inleiding

**Moet nog geschreven worden (einde project)**  
Awef (hoe persoonlijk willen we dit maken? We moeten er rekening mee houden dat dit verslag als het goed is de verwezenlijking van geweldigheid gaat worden. Misschien wordt het bij de universiteit bekeken en zou het raar zijn als we beginnen met “hallo, wij zijn Steven en Thijs van het Candea College in Duiven!”)

## Deelvraag 1 inleiding

Elk jaar boekt de mens grootschalige vorderingen op het gebied van computers, zowel hardware als software. Iets waar wij echter nog niet in geslaagd zijn te maken is een ware *Artificial Intelligence*, al lukt de schijn van denken creëren steeds beter. Voorbeelden zijn de ‘persoonlijke assistenten’ die inmiddels in elke smartphone geïntegreerd zijn. Siri, Google Now en Coratana maken gebruik van spraakherkenning om de gebruiker de gevraagde informatie te tonen, maar denken zoals mensen doen ze hierbij niet. Hoe een computersysteem toch ‘intelligent’ kan zijn zullen wij onderzoeken in dit \*\*hoofdstuk / deelonderwerp / …\*\*

## Verschillende algoritmes

Computers hebben geen dus bewustzijn. Om deze reden kunnen ze niet zelf bepalen iets te doen. Waar computers wel in uitblinken, is het uitvoeren van taken die ze zijn opgelegd. Vaak komen deze taken in de vorm van code. Via code kan je computers opdrachten geven, bijvoorbeeld “laat een scherm zien”. De boodschap valt echter niet op deze manier over te brengen, afhankelijk van de taal waarin je programmeert zijn er vaste commando’s waar de computer op zal reageren.  
Naarmate de opdracht die je een computer wil laten uitvoeren complexer worden, zal ook het gebruik in deze commando’s een verandering zien. Hier komen algoritmes in het spel. Een algoritme is een soort stappenplan voor de computer, waarin een complexere handeling in duidelijke opdrachten weergegeven wordt. De volgende definitie geeft een betekenis in de meest algemene zin: een algoritme is een eindige reeks instructies om vanaf een beginpunt een bepaald doel te bereiken.[a]  
Een toegankelijke vergelijking is koken. Er is een input van voedsel waar uiteindelijk een gerecht uit moet komen, de output. Voor het tot stand komen van dit gerecht gebruik je misschien een recept. Dit recept is als het ware het algoritme.  
Uit de gegeven definitie kan is af te leiden dat het aantal mogelijke algoritmes ontzettend groot is. Niet alleen is het een ruim begrip, ook kan het desbetreffende doel waarschijnlijk op meerdere manieren bereikt worden. In deze verschillende methodes kan de een echter beter zijn dan de andere, bijvoorbeeld door efficiënter te zijn.

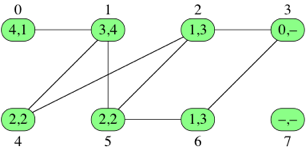
Uiteraard zijn er ook vele algoritmes die gebruik maken van toepassingen, zoals een *queue* en een *stack*, die betrekking hebben tot ons onderwerp. Enkele hiervan zullen hier beschreven worden:

### Breadth-first search (BFS)

Dit algoritme, bedacht in de jaren vijftig van de vorige eeuw door E.F. Moore[b], een Amerikaans professor in de wiskunde en computer sciences en een voortrekker in kunstmatig leven, is een zoekalgoritme voor data sets in de vorm van grafieken of ‘boom’-structuren, zoals een *Binary Tree* (Zie ook *Voorbeelden van algoritmes\*\*\*,* Figuur \*\*\*\*). In deze data set wordt een \*\*\*\*node\*\*\*\* als oorsprong benoemd, de *root.* Ook wordt een bepaalde uitkomst als doel gesteld. Vervolgens krijgt elke node twee waardes aangewezen:

* De afstand van de huidige node naar de root. Dit is het aantal stappen dat gezet moet worden om bij de root te komen.
* De node die vóór de huidige node kwam, de *predecessor*. Anders gezegd: bij welke node je uitkomt als je een enkele stap terug zet.

Bovendien heeft elke node een *state.* Deze state houdt bij of een node al gecontroleerd is.



Figuur \*\*\*\*

Hierboven is een voorbeeld van een dataset te zien bestaande uit acht nodes. In elke node staat eerst de afstand tot node 3 aangegeven en daarnaast de predecessor. Een liggend streepje betekent dat er geen waarde is. (De afstand van 3 tot 3 is niets, tussen 3 en 7 is geen verbinding, enz.)  
Node 3 wordt als root genomen. Daarna worden de nodes bekeken die aan 3 grenzen: 2 en 6 hebben een afstand van 1. De predecessor van deze nodes is 3. Vervolgens worden de nodes die aan 2 grenzen bekeken en herhaald het proces zich voor al deze nodes. Daarna worden de nodes die aan 6 grenzen bekeken. De afstand tot de root is bij deze nodes dus 2, er moeten immers twee stappen gezet worden weer bij 3 te komen. Dit proces wordt herhaald totdat het doel gevonden is.  
De aangrenzende nodes worden echter niet tegelijkertijd gecontroleerd. In dit geval wordt eerst 2 en daarna 6 gecheckt, daardoor is de predecessor van 5 node 2 en niet node 6. Op het moment dat node 5 gecontroleerd wordt, wordt de state op ‘bezocht’ gezet. Als vervolgens de aanliggende nodes van 6 bekeken worden, wordt opgemerkt dat 5 al gecontroleerd is. Zo weet het systeem dat 5 niet nogmaals gecheckt hoeft te worden. Omdat ook de root al als bezocht staat, wordt 3 ook niet gecontroleerd.

Om te zorgen dat het programma alle nodes afgaat, wordt een queue gebruikt. Dit is een wachtrij voor de nodes, er geldt: eerste erin, eerste eruit.  
In eerste instantie is node 3 de queue. Deze node wordt eruit gehaald en de aangrenzende nodes worden gecheckt en in de queue geplaatst, die is bestaat nu uit 2 op de eerste en 6 op de tweede plek. Dan wordt 2 eruit gehaald en sluiten 4 en 5 achteraan in de queue. 3 is al bezocht en hoeft dus niet toegevoegd te worden. Telkens wordt nu de eerste node uit de queue gehaald en worden de aangrenzende, niet eerder gecontroleerde nodes achteraan geplaatst. Hieruit valt op te maken dat prioriteit in de queue afhankelijk is van de afstand tot de root.

Uiteindelijk wordt ook de node die als doel werd gesteld gecontroleerd. Op dat moment zou het zoekproces stilgezet kunnen worden, als je slechts geïnteresseerd bent in het doel. Het programma kan nu laten zien want de afstand is tot de root.

## Voorbeelden algoritmes

Awefawef (willen we voorbeelden maken?)

* Binary tree (sorting system) -> BFS kan gebruikt worden om i.p.v. het doel te vinden, de stappen daarnaartoe naast elkaar te zetten.
* Maze generator
* The Oracle of Bacon

## Afbeeldingsresultaat voor binary treeGeen AI

Dit is geen AI want …

## Conclusie

Figuur \*\*\*\*: Een Binary Tree

Awef

* Van de algoritmes: wat is voor ons nuttig? (queue, stack)
* Wat is een zelflerend systeem

Gebruikte bronnen (temporary)

[a] Link: <http://www.woorden.org/woord/algoritme>   
Geraadpleegd op: 22- 05 – 2017   
Laatst gewijzigd op: niet bekend

Link: <https://beebom.com/examples-of-artificial-intelligence/>   
Geraadpleegd op: 22- 05 – 2017   
Laatst gewijzigd op: 26 – 09 – 2017

Link: <http://study.com/academy/lesson/what-is-an-algorithm-in-programming-definition-examples-analysis.html>  
Geraadpleegd op: 22- 05 – 2017   
Laatst gewijzigd op: 2017

[b] Link: <http://www.secfac.wisc.edu/senate/2003/0929/1727(mem_res).pdf>   
Geraadpleegd op: 23- 05 – 2017   
Laatst gewijzigd op: 23 – 09 – 2003