1. Inleiding

In dit hoofdstuk wordt verteld wat algoritmiek is, wat het belang van computers is en wat dit betekent voor biologen en 'Life Scientists'.

1.1 De naam[1]

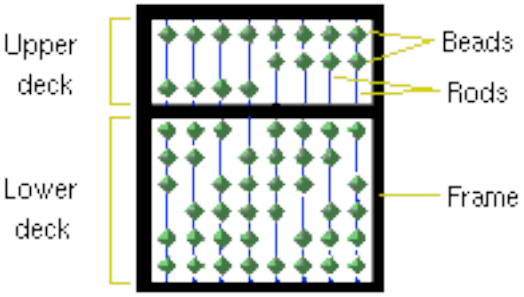
"We find only the older form '**algorism**' with its ancient meaning, i.e., the process of doing arithmetic using Arabic numerals. In the middle ages, abacists computed on the abacus, and algorists computed by algorism." (Donald Knuth, The Art of Computer Programming, Volume 1, Chapter 1)

Al eeuwen zijn er discussies over de denkwijze die ten grondslag liggen aan 'Algoritmiek'. Deze discussies hadden te maken met de manier van noteren van getallen en de manier van rekenen. Pas in 1202 introduceerde de Italiaan met de welluidende naam Leonardo di Pisa, beter bekend als Fibonacci, de Hindu-Arabische wijze van notatie van getallen zoals wij die vandaag de dag nog in Europa

[1] http://www.mathsyear2000.org/museum/floor4/gallery11/gal11_p2.html

gebruiken. Daarvóór noteerde men getallen op romeinse wijze, en men rekende op een abacus of telraam. Niet op papier en met een ganzeveer, want papier (papyrus of perkament) was veel te duur.

Vanaf het moment van introductie van deze nieuwe notatie ontstond de schoolstrijd tussen 'algoristen' en 'abacisten', een strijd die bijna 400 jaar zou duren[2].



The name Abacus derives from the Greek word *abax* meaning table or board covered with dust. The origins of the Abacus are buried deep in the history of mankind. It is known that in its 'modern' form it appeared in China in the 13th century AD. The Chinese Abacus is made of 13 columns with 2 beads on top (heaven) and 5 beads bellow (earth). The Japanese copied the Chinese Abacus around the 17th century AD and adapted it to their more delicate way of thinking. It has 21 columns with 1 bead on top (heaven) and 4 beads below (earth). The Abacus is still taught in the Far East as regular school training, and is used commonly in many places. In 1946 a contest between a Japanese Abacist (Kiyoshu Matzukai) and an Electronic computer was held for 2 days resulting in an unmistakable victory of the Abacist. The third modern form of the Abacus is Russian with 10 beads in 10 arched rows.[3]

In sommige europese landen was rekenen op 'algoristische' wijze zelfs bij wet verboden en kon je ervoor op de brandstapel belanden. Pas in de 16e eeuw, nadat papier tot een meer normaal gebruiksgoed was geworden door betere en goedkopere produktiemethoden, kwamen de algoristen geleidelijk aan de winnende hand. Maar tot op de dag van vandaag wordt er bijvoorbeeld in China, Japan, en Rusland routinematig (bijv. in de winkel bij de kassa) met behulp van telramen gerekend.

De komst van de digitale computer, zeg maar het 20e eeuwse digitale papier, heeft de positie van het algoritmisch benaderen en oplossen van numerieke en andere problemen nogmaals versterkt.[4]

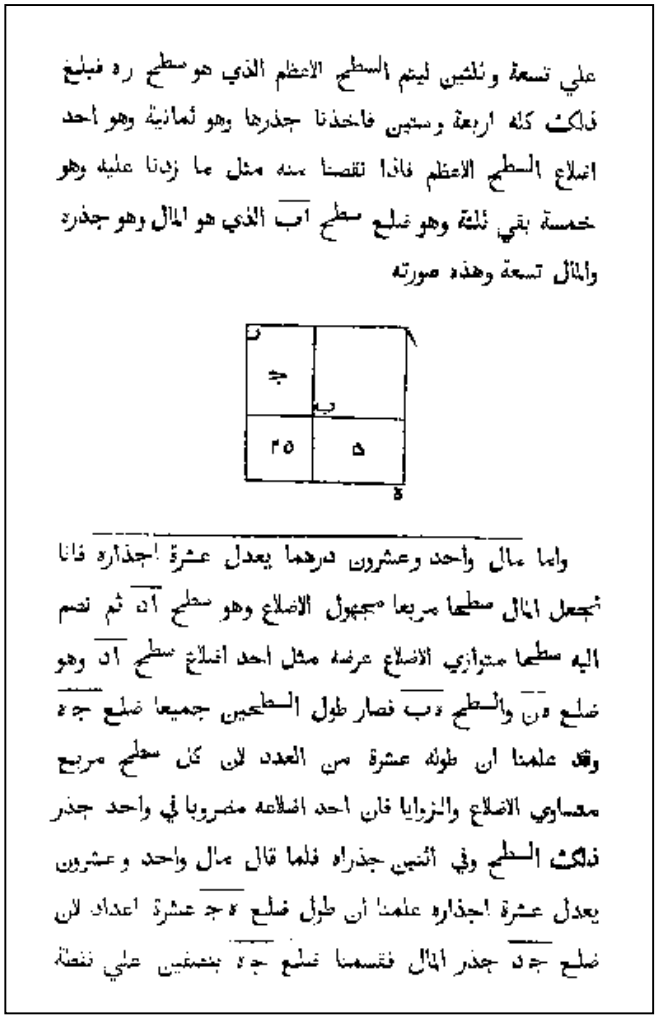
---

[2] http://www.scit.wlv.ac.uk/~cm1993/maths/mm2217/hanw.htm

http://faculty.philau.edu/LONDONM/history/Algorist.htm

[3] http://www.ee.ryerson.ca:8080/~elf/abacus/

[4] http://library.thinkquest.org/25018/english/text/history/h1.html



علي تسعة وثلثين ليتم السطح الاعظم الذي هو سطح ره فبلغ
ذلك كله اربعة وستين فاخذنا جذرها وهو ثمانية وهو احد
اضلاع السطح الاعظم فاذا نقصنا منه مثل ما زدنا عليه وهو
خمسة بقي ثلثة وهو ضلع سطح اب الذي هو المال وهو جذره
والمال تسعة وهذه صورته

واما مال واحد وعشرون درهما يعدل عشرة اجذاره فانا
نجعل المال سطحا مربعا مجهول الاضلاع وهو سطح اد ثم نضم
اليه سطحا متوازي الاضلاع عرضه مثل احد اضلاع سطح اد وهو
ضلع هن والسطح هب فصار طول السطحين جميعا ضلع جه
وقد علمنا ان طوله عشرة من العدد لان كل سطح مربع
متساوي الاضلاع والزوايا فان احد اضلاعه مضروبا في واحد جذر
ذلك السطح وفي اثنين جذراه فلما قال مال واحد وعشرون
يعدل عشرة اجذاره علمنا ان طول ضلع هج عشرة اعداد لان
ضلع جد جذر المال فقسمنا ضلع جه بنصفين علي نقطة

Over de herkomst van het woord algoritme bestaat ook al jaren verschil van mening. Er is bijvoorbeeld een opvatting die luidt dat het woord algoritme afgeleid zou zijn van logaritme, maar omdat het woord logaritme later is ontstaan dan algoritme, kan dit niet de oorsprong zijn. Volgens een andere zienswijze is algoritme een combinatie is van algiros (pijnlijk) en arithmos (getal). Hoewel dit in de praktijk vaak het geval is, is ook deze uitleg niet de juiste.

De meest waarschijnlijke veronderstelling is dat de naam is afgeleid van de naam van een beroemde 9e eeuwse Perzische wiskundige, *Abu Ja'far Mohammed ibn Mûsâ al-Khowârizmî*[5]

Hij wordt beschouwd als een van de grondleggers van de algoritmiek. Het woord 'algebra' is afgeleid van de titel van het boek (*Al-Jabr wa-al-Muqabilah*) dat hij over die materie schreef. Voor de liefhebbers staat links een stukje tekst uit dit beroemde werk afgebeeld:

1.2 Wat is algoritmiek?

In het dagelijkse leven heeft iedereen, zonder er bij stil te staan, te maken met algoritmiek (de leer van algoritmen). **Een algoritme is een beschrijving van de stappen die een processor (mens, computer, machine) moet doen om een proces uit te voeren**.

Enkele voorbeelden uit het dagelijks leven:

[5] http://www-groups.dcs.st-andrews.ac.uk/~history/Mathematicians/Al-Khwarizmi.html.

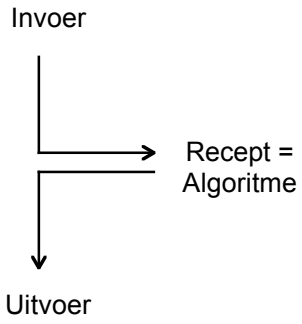
http://www.peak.org/~jeremy/calculators/alKwarizmi.html

http://users.erols.com/zenithco/khawariz.html

Proces	Algoritme	Stappen
een trui breien	breipatroon	één rechts, één averechts
een bouwpakket in elkaar zetten	gebruiks-aanwijzing	plak paneel A aan paneel B
een cake bakken	recept	klop drie eieren
een sonate van Beethoven spelen	bladmuziek	druk op toetsen van de piano
een kledingstuk maken	patroon	sla de zoom om, en stik die vast

voorbeeld: Een recept

Een algoritme is niets anders dan een recept dat gevolgd moet worden om de gewenste resultaten te krijgen. In bijlage 1 van dit hoofdstuk is een voorbeeld uit de practicum handleiding microbiologie afgebeeld. Dit is een recept voor de isolatie van thymine mutanten. We zien hierbij de invoer (wat hebben we nodig), het algoritme (de 7 stappen die uitgevoerd moeten worden) en de uitvoer (de platen met de kolonies). In schema wordt dit:



De microbioloog (of de student) is de uitvoerder (ook wel processor) van het algoritme. Bij algoritmen in de informatica is de computer de uitvoerder en moeten de recepten veel nauwkeuriger geschreven worden. Maar het principe van: geef de invoer en doorloop een recept in een aantal stappen, blijft hetzelfde. Als de invoer goed is en de stappen juist zijn beschreven, is de verkregen uitvoer ook de gewenste uitvoer.

Donald Knuth, beroemd Amerikaans expert op het gebied van de algoritmiek, geeft de volgende definitie van een algoritme.

Een algoritme is een procedure die aan 5 karakteristieken moet voldoen om resultaat te kunnen hebben:

1. Invoer

   In zijn meest algemene vorm is een algoritme, of recept, een transformatie die uitgaande van een set inputwaarden een bewerking toepast om een set output waarden te genereren. Een algoritme moet de eisen waaraan de input moet voldoen, duidelijk specificeren om de output toestand te kunnen bereiken.

2. Bepaaldheid

   Iedere stap, en de volgorde van de stappen, moet ondubbelzinnig bepaald zijn, d.w.z. mag niet voor verschilende interpretaties vatbaar zijn. Om die ondubbelzinnigheid te bereiken wordt gebruik gemaakt van een formeel gedefinieerde taal (computer-taal) en niet van een natuurlijke taal zoals Engels of Nederlands (Hoofdstuk 2 en 3).

3. Effectiviteit en correctheid.

   Een algoritme wordt geacht een probleem op te lossen. Van ieder voorgesteld algoritme zullen we moeten

aantonen dat het inderdaad het probleem op *kan* lossen. Vaak gebeurt dit met behulp van een mathematische of logische redeneervorm waarin wordt vastgesteld dat als aan alle input condities is voldaan en alle stappen van het algoritme in de correcte volgorde worden uitgevoerd dan inderdaad de gewenste ouput wordt geproduceerd.

4. Eindigheid

   Het gestelde doel moet in een *eindig* aantal stappen bereikt worden. We hoeven niet het exacte aantal stappen te weten. Een schatting van de bovengrens is voldoende, samen met een (mathematische of logische) redeneervorm waaruit blijkt dat het algoritme altijd binnen de gestelde bovengrens zal eindigen (Hoofdstuk 8).

5. Uitvoer

   Van ieder algoritme moet duidelijk zijn wat het moet bewerkstelligen. Net als dat bij input het geval was moet de output beschreven kunnen worden in termen van een resultaat met zekere eigenschappen.

Hoe zit het nu met deze 5 punten in ons voorbeeld van de isolatie van thymine mutanten.

1. Invoer

   De invoer specificatie laat in het voorbeeld weinig aan duidelijkheid te wensen over. Een uitputtende en duidelijke opsomming van ingredienten gaat aan het recept vooraf. Aangezien het recept bedoeld is om door mensen uitgevoerd te worden en niet door een computer volstaat in dit geval het gebruik van alllerlei technische termen (zoals SM, TRIM, Thymine, fysiologisch zout, NaCl, steriel, plaatbuis, pipet, bacterie, suspensie), er van uitgaande dat een voldoende wegwijs gemaakt persoon aan het recept begint.

2. Bepaaldheid

   De vraag is of alle stappen in het recept en hun volgorde voldoende uitputtend en ondubbelzinnig zijn beschreven. Om dit vast te stellen is niet alleen begrip van het jargon vereist (Drychalski spatel), maar ook wordt soms impliciete kennis van deelprocessen verondersteld (zoals spatelen, pipetteren, incuberen). Andere deelprocessen, zoals de UV belichting, worden wel in hun stappen ontleed.

3. Effectiviteit en correctheid.

   Dit is hier lastig van te voren vast te stellen. De proef kan natuurlijk mislukken door onoordeelkunig handelen. Aangezien aangenomen mag worden dat het recept is opgesteld door een ervaren thymine-mutanten-isoleerder, is de veronderstelling van effectiviteit en correctheid impliciet aanwezig. Het is onduidelijk in hoeverre een mathematische of logische redeneervorm ons hier kan helpen om de effectiviteit en correctheid van het recept vast te stellen.

4. Eindigheid

   Het proces wordt duidelijk in een eindig aantal stappen beschreven, met als laatste stap 3 dagen wachten voor dat iets van een resultaat bekeken kan worden.

5. Uitvoer

   Ook hier geen onduidelijkheid. Er moeten bacteriekolonies zijn opgekomen en het aantal resistenten moet telbaar zijn.

1.3 Algoritmiek met de computer

In deze syllabus gaan we niet behandelen hoe je een breipatroon of een cake recept moet volgen. We zullen ons gaan richten op het maken van algoritmen voor de computer.

In het dagelijks leven neemt de computer een steeds grotere plaats in. Allerlei processen die vroeger alleen door middel van veel handwerk tot stand konden worden gebracht, worden nu met een druk op de knop in een geautomatiseerde procesgang uitgevoerd. Er zijn talloze redenen voor het gebruiken van computers. Om redenen van nauwkeurigheid worden ze ingezet voor boekhoudkundige taken, zoals de studietoelagen administratie, of het opstellen van mathematische tabellen. Gebruiksgemak speelt een voorname rol, zoals bijvoorbeeld bij de tekstverwerker als opvolger van de typmachine. Of het in korte tijd kunnen verwerken van grote hoeveelheden gegevens, zoals bij het opstellen van weersvoorspellingen of het maken van een spoorboekje. Of om processen te laten verlopen onder omgevingsomstandigheden waarin mensen niet of slechts tegen zeer hoge kosten kunnen werken, zoals bijvoorbeeld in de ruimte. In al deze gevallen komen de door de computer te verwerken problemen voort uit menselijke ondervindingen. De problemen ontstaan en worden ontdekt, geanalyseerd en opgelost op de manier zoals mensen met problemen omgaan. Dat gebeurt vaak in een complexe combinatie van intuitie, gokken, ervaring, inzicht, trial and error, en op goed geluk. Een computer, aan de andere kant, is een machine die alleen maar heel eenvoudige handelingen kan verrichten, maar wel op een hele snelle manier. Als probleemoplossingen door zo'n machine worden bewerkt, dan moeten ze worden uitgedrukt in termen die de computer kan begrijpen, d.w.z. een combinatie van eenvoudige, logische, rekenkundige, en symbolische instructies. Het grootste obstakel wordt dan gevormd door het gapende gat tussen het **probleem**, meestal in typisch menselijke termen verwoord, en een **oplossing** in de vorm van een serie opdrachten die door een computer kunnen worden uitgevoerd. Dat gapende gat kan worden gedicht door de inzet van een (computer)taal die door de machine kan

worden verstaan, en een wijze van instructies geven die door de machine kan worden gevolgd.

Het vak algoritmiek leert je hoe je een proces door de computer kan laten uitvoeren. Het begin wordt altijd gevormd door de *analyse van het proces* om *aard en aantal basale stappen* (= *elementaire acties*) te achterhalen en die vervolgens *te vertalen* in een algoritme. Daarna gaan we het algoritme omzetten in instructies geschreven in een computertaal, de zg. *implementatie* van het algoritme

1.4 Algoritmiek en levenswetenschappen

Recepten zijn leuk voor keukenprinsen en algoritmiek is leuk voor programmeurs, maar waarom moet een bioloog of een LS&T'er weten hoe je een algoritme schrijft? Wij kunnen toch gewoon de standaard programma's gebruiken die programmeurs gemaakt hebben!?

Toch is het, om (in ieder geval) twee redenen, voor biologen en 'Life Scientists' belangrijk dat ze weten hoe een algoritme in elkaar zit.

Ten eerste gebeurt het vaak dat een standaard programma een vreemd antwoord geeft (of helemaal geen antwoord). Als je weet hoe het programma ongeveer werkt, is het antwoord vaak makkelijker te verklaren en kan je leren om makkelijker en sneller met de computer te werken.

Ten tweede zijn de problemen die je binnen de Biologie en de 'Life Sciences' tegen komt vaak zeer specifiek en bestaan er geen standaard programma's voor. Je kan dan iemand inhuren, of wachten totdat iemand een programma voor je wil schrijven. Je kan echter ook het programma zelf schrijven (nadat je het probleem hebt ontleed in hapklare brokken waarvan de oplossingen stuk voor stuk in een recept (= algoritme) zijn te formuleren). Voor kleinere problemen is deze aanpak vaak de beste, snelste en goedkoopste oplossing. Maar ook voor grotere en meer complexe problemen kan het de enige manier zijn om je eigen ideën omtrent een oplossing in een werkbare en werkende vorm te gieten en ook aan anderen door te geven. Om voorbeelden te vinden hoeven we niet eens ver van huis. Zo zijn binnen de sectie Theoretische Biologie van het EEW specifieke ideeën ontwikkeld omtrent het vinden van de juiste secundaire structuur van RNA en die ideën zijn vervolgens vastgelegd in het programma ***STAR*** (van Batenburg et al., 1990); de ideeën voor phylogenie-reconstructie en historische biogeografie in het programma

CAFCA (Zandee, 1987), en die voor de statistische analyse van gedragsgegevens in het programma ***The Analyst*** (Haccou & Meelis 1992, van Batenburg et al. 1994)

1.5 Wanneer?

Er zijn een aantal gevallen waarbij het makkelijk zou zijn om een computer te gebruiken.

1. Het probleem (de berekening van een probleem) is zo groot dat het niet meer met gebruik van papier en potlood alleen valt op te lossen.

Voorbeeld:

Bij het berekenen van statistische toetsen speelt de hoeveelheid waarnemingen een belangrijke rol. Met de toename van het aantal gegevens neemt het aantal berekeningen echter ook sterk toe. Voor de computer is het aantal echter geen probleem, een oplossing per computer ligt dus voor de hand. Sommige toetsen, benodigd voor biologische problemen, zijn echter zo specifiek, dat ze niet in een standaard programma terug te vinden zijn. Het loont dan vaak de moeite om zelf een programma (algoritme) te schrijven.

2. Om te kijken wat het gedrag van een model is, wordt vaak dit model een aantal keer met verschillende parameters uitgevoerd. Aan de hand daarvan kunnen uitspraken gedaan worden over het gedrag van het model.

Voorbeeld:

Om te weten te komen wat de invloed is van bepaalde parameters op de interactie tussen prooi en predator wordt vaak een model opgesteld en herhaaldelijk getest. Aan de hand van de antwoorden wordt dan gekeken welke parameters een belangrijke rol spelen en welke niet.

Voordat je echter achter de computer aan de slag gaat is het verstandig om eerst je gedachten te ordenen en over de structuur van je toekomstige programma na te denken. Dit is de kern van algoritmiek.

1.5 Doel van het college

Het college algoritmiek en de bijbehorende werkgroep heeft tot doel om gevoel en inzicht te ontwikkelen voor het stapsgewijs, receptmatig, oplossen van daartoe geeigende probleemstellingen uit de levenswetenschappen, en dat inzicht te gebruiken in het vorm geven van een programmeerbare oplossing.

Het vorm geven van die oplossing gebeurt in de programmertaal R.

Op basis van college en werkgroep wordt je geacht tijdens de volgende fases van je studie te kunnen onderscheiden welke problemen in de diverse subdisciplines van de Biologie en 'Life Sciences' geschikt te maken zijn voor de algoristische aanpak en deze aanpak ook vorm te geven.

Bijlage 1

Voor de proef hebben we nodig:

- 2 platen met synthetisch (SM) medium, deze zijn niet gemarkeerd
- 1 plaat SM + TRIM (zwarte streepcode)
- 2 platen SM + TRIM + Thymine (rode streepcode)
- fysiologisch zoutoplossing (0,9% NaC1) om de verdunningen in te maken
- 5 Steriele plaatbuizen
- steriele pipetten
- 1 plaatbuis met bacteriesuspensie

UITVOERING ISOLATIE THYMINE MUTANTEN

1. Vul vier steriele plaatbuizen elk met 9,9 ml fysiologische zoutoplossing, vul de vijfde buis met 4,5 ml fysiologisch zout. Deze handelingen moeten allemaal steriel verricht worden.

2. Neem de plaatbuis met bacterie suspensie en spatel hiervan 0,1 ml uit op een TRIM+Thy plaat m.b.v. een Drychalski spatel.

3. Pipetteer uit dezelfde plaatbuis met bacteriesuspensie 0,1 ml in een buis met 9,9 ml fysiologisch zoutoplossing, meng deze op de vortex (buis aan de bovenkant vasthouden !), dit is de 10^{-2} verdunning. Pipetteer uit deze 10^{-2} verdunningsbuis 0,1 ml in een nieuwe buis met 9,9 ml fysiologische zoutoplossing, vortex weer, dit is de 10^{-4} verdunning. Pipetteer 0,5 ml uit deze 10^{-4} verdunningsbuis in de buis met 4,5 ml fysiologische zoutoplossing, vortex weer. Dit is de 10^{-5} verdunning. Spatel 0,1 ml van de 105 verdunning uit op een SM plaat.

4. Nu gaan we de bacteriën met UV licht bestralen. Pipetteer 0,5 ml uit de oorspronkelijke plaatbuis met bacteriesuspensie midden in een lege petrischaal (deksel er weer op doen) . Loop heel voorzichtig (de druppel moet in het midden blijven liggen) naar de UV-kast. Zet de veiligheidsbril op en plaats de petrischaal met gesloten deksel in de UV-kast. Bestraal de bacterien 20 seconden door het deksel te verwijderen en na 20 seconden weer terug te plaatsen. Loop terug naar je plaats.

5. Spatel 0,1 ml van het bestraalde mengsel uit op een TRIM+Thy plaat, en ook 0,1 ml op een TRIM plaat.

6. Verdun de bestraalde bacteriën tot 10^{-4}. Spatel van deze 10^{-4} verdunning 0,1 ml uit op een SM plaat. Incubeer alle vijf platen bij 37°C.

De platen kunnen na drie dagen bekeken worden.

Tel het aantal opgekomen kolonies op alle platen, en bereken het aantal bacteriën per ml in de oorspronkelijke (onbestraalde) suspensie. Bijv. op de SM plaat zijn 50 kolonies aanwezig. Het aantal bacteriën in de oorspronkelijke suspensie was dan $50 \times 10 \times 10^5 = 5 \times 10^7$ bacteriën/ ml.

De factor 10 is de omrekeningsfactor omdat er maar 0,1 ml is uitgeplaat, en de omrekeningsfactor 10^5 is de verdunningsfactor.

Bereken nu ook het aantal levende bacteriën/ml in de bestraalde suspensie. Hoeveel van de cellen zijn gedood door de bestraling?