Inleiding programmeren

1e jaar wis-, natuur- en sterrenkunde

Universiteit van Amsterdam

oktober 2013

Opgaves bij college 2

basis wiskunde en getaltheorie

1. priemgetallen

opgave 1: schrijf een programma `'primes.py'` dat het duizendste priemgetal berekent en print. Print ook de lijst van alle 1000 priemgetallen.

computing hints:

- een manier om te testen of een getal a een veelvoud is van een getal b (b deelt a met rest 0) is het gebruik van de %-operator. In python code geeft **a%b** de rest (8%3 is 2). Check de werking op de command-line.
- Gebruik `list` om reeksen getallen te bewaren. Check de documentatie.

Hoewel een computer je in staat stelt om snel te rekenen is het toch belangrijk om voor elk probleem de optimale strategie te bepalen. Hier bijvoorbeeld:

strategie hints:

- Behalve 2 zijn even getallen nooit een priemgetal
- Verzin hoe je per priem-kandidaat bijhoudt of het wel/niet priem is als je over kandidaat delers heenloopt. Bedenk van tevoren hoe je de lijst met gevonden priemgetallen gaat opslaan.
- Wanneer kan je stoppen ? Als je wilt bepalen of 37 een priemgetal, welke kandidaat delers bekijk je voordat je zeker weet dat het een priemgetal is ?
- Print voor elke kandidaat informatie zodat je weet waar je bent in de berekening en je ziet of de computer ook echt jouw strategie volgt.
- Zorg dat het programma stopt bij het 1000ste priemgetal. Bedenk dat je programma waarschijnlijk niet het eerste priemgetal heeft gegenereerd (2).

Als je wilt controleren of je programma goed werkt kan je je gevonden lijst priemgetallen hier matchen met een lijst bekende priemgetallen:
`http://primes.utm.edu/lists/small/1000.txt`

opgave 2: Welke getallen (onder n=10000) vormen de langste reeks aaneengesloten niet-priemgetallen ? Start met de code uit vraag 1.

hacker: maak bij het testen van getal n gebruik van de kennis over de priem-priemgetallen onder de n. Tot welk priemgetal kom je in 1 minuut ?

2. Getaltheorie

a) Vermoeden van Goldbach: Het vermoeden van Goldbach is een van de oudste onopgeloste problemen is in de wiskunde. Goldbach postuleert dat:

Elk even getal groter dan 2 kan geschreven worden als de som van 2 priemgetallen

Een priemgetal mag hierbij twee keer gebruikt worden. Hoewel dit inderdaad klopt voor alle getallen tot $4 \cdot 10^{18}$ is er nog geen bewijs. We gaan ons steentje bijdragen.

opgave 3: Laat zien dat alle even getallen tot 1000 inderdaad te schrijven zijn als de som van 2 priemgetallen

b) Bevriende getallen: Twee getallen A en B zijn *bevriende* getallen als de som van de delers van A (niet A zelf, maar inclusief 1) B oplevert. Tegelijkertijd moet de som van de delers van B A opleveren. Het eerste paar bevriende getallen is al lang bekend (220,284). Er geldt immers:

som delers $220 = 1+ 2 + 4 + 5 + 10 + 11 + 20 + 22 + 44 + 55 + 110 = 284$
som delers $284 = 1 + 2 + 4 + 71 + 142 = 220$

opgave 4: vind het volgende paar bevriende getallen en beschrijf je strategie

3. Hacker (priemgetal eigenschappen)

Vanuit de elementaire getaltheorie is bekend dat voor grote n het product van alle priemgetallen onder de n minder is dan e^n en dat bij groeiende n het product steeds dichter bij e^n komt te liggen.

opgave 5: schrijf een programma `'products.py'` dat laat zien dat dit zo is. Specifiek: bepaal voor elke n de som van de logaritmes van alle priemgetallen (van 2 tot n) en op het scherm print: a) het getal n, b) de som van de logarithmes van alle priemgetallen onder n en c) hun ratio

computing hints:

- als je wiskundige operaties wilt gebruiken in python moet je die eerst 'beschikbaar' maken. Dat doe je door ze bovenin je programma te 'importeren'. Als je de sinus functie wilt gebruiken om bijvoorbeeld sin(0.5) uit te rekenen moet je die sinus-functie eerst laden. Bovenin je programma zet je dan (twee opties):

  optie 1) `import math` importeert de math library
  In je programma zeg je dan `math.sin(0.5)`

  optie 2) `from math import *` importeert expliciet alle functies
  In je programma kan je dan direct `sin(0.5)` gebruiken

strategie hints en wiskundige achtergrond:

- een directe check is om het product van de priemgetallen te vergelijken met e^n. Helaas levert het product al snel een zeer groot getal op, wat problemen met numerieke precisie betekent (bespreken we later). Door aan beide kanten van het =-teken de logaritme te nemen wordt het product van priemgetallen een som van de logaritmes. De oorspronkelijke vergelijking hierboven wordt dan:

$$\sum_{i\ (i=\text{prime},i<n)} \log(i) < n,$$

  waarbij het verschil tussen beide kanten kleiner wordt als n groeit.

Tip: begin met je werkende programma van opgave 1.

opgave 6: maak een grafiek om je resultaat zichtbaar te maken

4. Hacker (Diophantische McNuggets)

In de wiskunde bestaat er een klasse vergelijkingen die bekend zijn onder **diophantische vergelijkingen**. Het zijn vergelijkingen waar de variabelen alleen geheeltallig kunnen zijn. De bekendste diophantische vergelijking is:

$$x^n + y^n = z^n$$

Voor n=2 zijn er oneindig veel oplossingen (waarden van x,y,z die elke geheeltallig zijn): Pythagoras. De beroemde stelling van Fermat zegt dat voor waarden groter dan 2 er geen geheeltallige oplossing is. Ook McDonalds gebruikt diophantische vergelijkingen. Jazeker. McDonalds verkoop namelijk McNuggests in verpakkingen van 6, 9 of 20 McNuggets. Het is bijvoorbeeld mogelijk om exact 15 McNuggets te kopen (1x6 + 1x9), maar onmogelijk om 16 McNuggets te kopen. We gaan in deze opgave berekenen wat het grootste aantal McNuggets is dat je kan bestellen dat niet precies 'past'. Om te kijken of je precies n McNuggets kan kopen moet je een diophantische vergelijking oplossen, nl.: vind positieve gehele getallen a, b en c zodanig dat:

$$6a + 9b + 20c = n$$

Voor we in opgave 5 gaan bepalen wat het grootste aantal McNuggets is dat *niet* precies besteld kan worden proberen we 2 strategie-hints uit te werken.

opgave 7: Laat zien dat het mogelijk is om precies 50, 51, 52, 53, 54 en 55 McNuggets te bestellen (met pen en papier of eigen programma). Laat steeds zien hoeveel doosjes van 6,9 en 20 McNuggets je krijgt.

Theorema: Als het mogelijk is exact x, x+1, ... en x+5 McNuggets te bestellen dan betekent dat dat je *elk* aantal McNuggets $\geq$ x kan bestellen als de McNuggets komen in dossjes van 6, 9 en 20.

opgave 8: Beschrijf waarom bovenstaand theorema waar is (in tekst) en overtuig jezelf dat het antwoord uit vraag 3 (50, 51, ..., 55) betekent dat ook (56, 57, ... 61) een oplossing zijn. Sterker: alle aantallen boven de 50.

Dan komen we bij de eigenlijke hoofdvraag:

opgave 9: schrijf een programma dat het grootste aantal McNuggets (N_{max}) bepaalt dat niet precies past in doosjes van 6, 9 en 20.

Note: de output van het programma moet als volgt op het scherm komen:
`Het grootste aantal McNuggets dat niet exact besteld kan worden is: ...`

strategie hints:

- gebruik de antwoorden uit vraag 1 en 2 in je strategie
- Bedenk voor je begint goed welke grootheden je bij moet houden als je door de verschillende mogelijkheden heen-'loopt'.

Youtube filmpje met het goede antwoord: McDonaldsmedewerkertje pesten

Optioneel: van concreet probleem naar algemene functie

Nu we een concrete vraag hebben opgelost kunnen we ons programma uitbreiden naar een meer algemene oplossing. We gaan nu oplossingen bekijken voor verschillende keuzes van doosjes grootte (wel nog steeds 3 doosjes). Stel dat er een variabele in je programma is (tuple of array van lengte 3) is die `packages` heet en de grootte van de verschillende McNugget doosjes bevat.

opgave 10: Schrijf een programma dat gegeven 3 verschillende doosjes het grootste aantal McNuggets vindt (onder de 200) dat niet precies in een geheel aantal doosjes past.

Note: Zorg dat je programma de volgende output geeft:
`Gegeven 3 McNugget doosjes met grootte x, y en z, dan is het grootste aantal McNuggets dat niet exact besteld kan worden:` $\mathtt{N}_{\max}$.

strategie hints:

- We limiteren onze test tot 200 omdat er in sommige gevallen geen grootste aantal is, denk bijvoorbeeld aan 3 doosjes van 10.
- Test je programma op een groot aantal keuzes door de variabele in `packages` te veranderen en vergeet niet ook (6,9,20) mee te nemen.