De wiskunde van geheimschriften

R. Cramer, B. de Smit, P. Stevenhagen,
A. Stolk, M. Streng, L. Taelman

Februari – Maart 2008

Inhoudsopgave

College 1

Geheime communicatie

Wat is cryptografie?

Wat het *schrift* tot een van de fundamenten van de menselijke cultuur maakt, is de mogelijkheid die het geeft om gedachten op zo'n manier aan papier of een ander medium toe te vertrouwen, dat zonder verdere bemoeienis van de auteur tallozen kennis kunnen nemen van die gedachten. Deze prettige eigenschap wordt echter in bepaalde situaties als een nadeel ervaren: als de brief van Romeo in de handen van Julia's ouders valt, of de instructies van de legeraanvoerder in de handen van de generaals van het vijandige leger, is de auteur juist uiterst ongelukkig met de toegankelijkheid van het bericht. Geheimschriften zijn daarom al bijna zo oud als het schrift zelf, en er is een heel arsenaal aan methoden om berichten zodanig te *versleutelen* dat alleen de geadresseerde het bericht kan lezen. De *cryptografie* – het woord combineert de Griekse stammen $\kappa\rho\upsilon\pi\tau\epsilon\iota\nu$ = verbergen en $\gamma\rho\alpha\varphi\epsilon\iota\nu$ = schrijven – is de kunst om dat op betrouwbare wijze te doen.

De klassieke opzet van een cryptografisch systeem bestaat eruit dat de geadresseerde met de zender van het bericht een *sleutel* afspreekt die gebruikt kan worden om zijn bericht te decoderen. Idealiter zijn met de sleutel alle berichten eenvoudig te decoderen, en is zonder de sleutel aan geen enkel gecodeerd bericht informatie te ontlenen. De *cryptologie* is de wetenschap die voor dit soort vaag geformuleerde wensen een theoretisch kader schept, en probeert de veiligheid van systemen te *bewijzen*. De laatste twee colleges van deze masterclass geven daar een indruk van.

Vertrouwelijke informatie over publieke kanalen

Het bewijzen van veiligheid klinkt wellicht als een academische bezigheid, maar dat is bepaald niet het geval. In de eerste plaats heeft de geschiedenis ons geleerd dat vele 'onkraakbaar' geachte systemen toch steeds gebroken werden. De Duitse versleuteling in de tweede wereldoorlog van militaire communicatie met de Enigma-code is een beroemd geworden voorbeeld: ook toen de Bletchley Park groep van de beroemde Britse wiskundige Alan Turing er in slaagde om met behulp van wiskunde en de eerste primitieve computers steeds meer berichten te ontcijferen, bleef het Duitse geloof in de onmogelijkheid van het kraken van de code bestaan. In de tweede plaats is het zo dat versleuteling van berichten in onze tijd niet meer iets is dat voornamelijk een klein aantal topgeheimen van de staat betreft, en waar een gespecialiseerde club van experts full time aandacht aan kan besteden. Met de opkomst van internet is communicatie over min of meer publieke kanalen een zaak van de gewone man geworden, en behalve semi-vertrouwelijke communicatie zoals e-mail die verschaft, is er nu ook een intensief verkeer dat met allerhande webtransacties samenhangt. Vrijwel iedereen stuurt tegenwoordig zijn credit card gegevens over het web, of geeft electronisch zijn bankopdrachten. Zonder adequate versleuteling is zoiets niet mogelijk, en de eisen die hier gesteld worden zijn een beetje anders dan in de tijd van Enigma. Een belangrijk verschil is dat in veel gevallen de 'kennismaking' pas ten tijde van de transactie plaatsvindt, zodat de credit card eigenaar en de handelaar helemaal geen gelegenheid hebben om van te voren geheime sleutels voor hun communicatie af te spreken. Ze zijn daarom ook voor het afspreken van een sleutel aangewezen op het publieke kanaal waarlangs zij communiceren.

Dat het toch mogelijk is om langs publieke weg dagelijks miljoenen vertrouwelijke berichten te versturen, is een verdienste van de wiskunde, en deze masterclass zal uitleggen welke eenvoudige principes aanleiding geven tot een zeer populair versleutelingssysteem, RSA geheten, naar de drie ontdekkers Rivest, Shamir en Adleman. Er zijn diverse andere systemen, maar die zijn grotendeels op wat geavanceerdere wiskunde gebaseerd, die op de middelbare school niet aan de orde komt. Wie wiskunde gaat studeren en de geschikte colleges volgt komt een aantal ervan vanzelf tegen.

Figuur 1: Adi Shamir, Ronald Rivest en Leonard Adleman in hun studententijd.

Eenrichtingsfuncties in de wiskunde

Het RSA-systeem dat wij in deze masterclass bespreken, is een voorbeeld van een *public key cryptosystem*: het spreekt zowel de versleutelingsmethode als de sleutel tot het decoderen op publieke wijze af! Dat dit enigszins contraïntuitief tot een veilig systeem aanleiding geeft, is gebaseerd op het bestaan van zogenaamde *eenrichtingsfuncties* in de wiskunde. Dit zijn functies waar je een bericht in kunt stoppen om een versleuteld bericht als antwoord er uit te krijgen. Om het bericht te decoderen op grond van de beschikbare informatie is een berekening nodig waarvoor op dit moment meer tijd nodig is dan het heelal oud is, en dat maakt zo'n berekening totaal ondoenlijk. De maker van het systeem, die het bericht ontvangt, heeft echter beschikking over extra informatie, waarmee het heel eenvoudig is om de eenrichtingsfunctie 'om te keren', en uit het versleutelde bericht het originele bericht af te leiden.

In het geval van RSA is zo'n functie gebaseerd op elementaire getaltheorie, en de bouwblokken waarmee het gemaakt wordt zijn de al duizenden jaren bekende *priemgetallen*: positieve gehele getallen $n > 1$ met de eigenschap dat hun enige positieve delers 1 en n zelf zijn. Daarvan bestaan er

hele kleintjes, zoals 2, 3 en 5, maar ook veel grotere zoals $2^{32\,582\,657} - 1$, een priemgetal van bijna tien miljoen decimale cijfers. Alle positieve gehele getallen zijn te krijgen als producten van priemgetallen, en de manier waarop dat gebeurt doet een beetje denken aan de manier waarop moleculen uit atomen opgebouwd zijn. Het volgende college gaat daar in detail op in. Voor nu is het voldoende om te weten wat priemgetallen zijn, en te beseffen dat de RSA-eenrichtingsfunctie gemaakt wordt met het behulp van twee grote priemgetallen p en q van een paar honderd cijfers elk, die door de ontvanger van de berichten gekozen worden. Hij maakt slechts het product $N = pq$ van deze beide priemgetallen voor algemeen gebruik bekend, en anders dan in het geval van $N = 91$, waarvan iedereen ziet dat het 7×13 is, kan voor een getal $N = pq$ van 400 cijfers niemand behalve degene die p en q gekozen heeft er achter komen wat p en q zijn.

Het wekt in eerste instantie misschien verbazing dat processen die in één richting erg eenvoudig zijn, zoals het vermenigvuldigen van twee grote priemgetallen p en q om N te krijgen, in omgekeerde richting zo moeilijk zouden zijn. Het is namelijk zo dat in de wiskunde allerlei ogenschijnlijk ondoenlijke problemen *wel* opgelost kunnen worden door slimme methoden te gebruiken, en RSA is gebaseerd op iets dat we kennelijk *niet* zo goed kunnen: het ontbinden van grote getallen in hun priemfactoren. Een goede kennis van wiskunde is vereist om te begrijpen wat echt moeilijke problemen zijn die tot een veilig systeem aanleiding geven, en in veel gevallen is er helemaal geen overtuigend bewijs dat iets echt moeilijk is: wie weet vindt morgen iemand een snelle methode om getallen te ontbinden! Het is dan goed om te weten dat er behalve RSA nog andere cryptosystemen bestaan die op geheel andere principes gebaseerd zijn, en dat de dag waarop alle onderliggende wiskundige problemen voor die systemen opgelost worden nog wel ver in de toekomst zal liggen.

Complexiteit

Sinds Turing in de jaren veertig van de vorige eeuw computers ontwierp om Enigma te ontcijferen, zijn computers tot gebruiksvoorwerpen geworden die in elk huishouden voorkomen, en de rekensnelheid van een moderne PC is naast die van Turings machines als de snelheid van een straaljager in vergelijking met die van een slak. Heel naief zou men dus kunnen denken dat een getal N van 200 cijfers gemakkelijk als een product van 2 priemgetallen van 100 cijfers herschreven kan worden door een sterke computer een paar

nachtjes te laten rekenen. Immers, een computer kan razendsnel twee gehele getallen van een paar honderd cijfers op elkaar delen, en een computer die gewoon delers van N probeert komt na eindig veel tijd zo'n factor van 100 cijfers wel tegen.

Eindig veel tijd is een vaag begrip, en een schatting van de looptijd van het proberen van delers op een computer laat zien dat we nooit op het antwoord zullen kunnen wachten. Immers, het proberen van alle 10^{100} delers van niet meer dan 100 cijfers kost een snelle computer die een biljoen delers per seconde kan testen nog steeds $10^{100}/10^{12} = 10^{88}$ seconden. Om een idee te krijgen van grote getallen: er gaan ongeveer $3 \cdot 10^7$ seconden in een jaar, een mensenleven duurt gemiddeld ruim $2 \cdot 10^9$ seconden en de geschatte leeftijd van het heelal is om en nabij de 10^{18} seconden. Dit laat zien dat zelfs als alle computers ter wereld ingezet worden, en deze nog eens duizend keer zo snel gaan rekenen als nu mogelijk is, het antwoord van zijn levensdagen niet in zicht komt.

Natuurlijk is het niet zo dat het domweg proberen van delers, ook wel *trial division* genoemd, de enige methode is om getallen te ontbinden. Er zijn veel betere methoden om te factoriseren, en die maken gebruik van geavanceerde wiskunde. Met een als de *getallenlichamenzeef* bekend staande methode is het sinds kort mogelijk om getallen van 200 cijfers *wel* te ontbinden – maar 400 cijfers is op dit moment nog veel te groot.

Om te beoordelen hoe goed een methode als de getallenlichamenzeef zou zijn om een getal van 400 cijfers te ontbinden is het niet voldoende om zo'n methode maar eens te testen op een computer. Een gedegen studie van het asymptotisch gedrag van oplossingsmethoden bij steeds groter wordende input is het terrein van de *complexiteitstheorie*. Die theorie doet geen uitspraken over vaste computers of input, maar zegt dingen als dat de looptijd van de trial division methode voor het ontbinden van een getal N in het ongunstigste geval – en dat is precies het RSA-geval van twee ongeveer even grote priemfactoren – orde van grootte $\sqrt{N}$ heeft. In vergelijking met de lengte van de input, die voor een getal N iets als $\log N$ cijfers nodig heeft, is dat van *exponentiële* grootte.

Exponentiële methoden worden voor groeiende inputlengte al snel ondoenlijk langzaam, en trial division is er een goed voorbeeld van. De methoden die zich wel goed gedragen bij groeiende inputlengte hebben een looptijd begrensd door een *polynomiale* uitdrukking in de lengte van de invoer. Voor het ontbinden van een getal N, waarbij de inputlengte orde van grootte $\log N$ heeft, zijn geen polynomiale methoden bekend. De vooruitgang in de laatste

decennia is echter spectaculair, en het 'onontbindbare' getal van 129 cijfers dat de ontdekkers van RSA in 1977 als voorbeeld ten tonele voerden werd 15 jaar later al ontbonden met een voorloper van de getallenlichamenzeef. Deze vooruitgang is het resultaat van slim gebruik van wiskundige technieken die op het eerste gezicht weinig met het ontbinden van gehele getallen te maken hebben. Ze maken deel uit van het arsenaal van moderne wiskunde dat je je als wiskundige van de 21^{e} eeuw eigen moet maken.

Computerpracticum

Bij het computerpracticum gaan we gebruikmaken van het programma *Sage*. De eenvoudigste manieren om Sage te gebruiken zijn via de *commandoregel* en de *notebook interface*. De Sage commandoregel is op te starten met het commando `sage`. Je krijgt dan een scherm te zien met

```
sage:
```

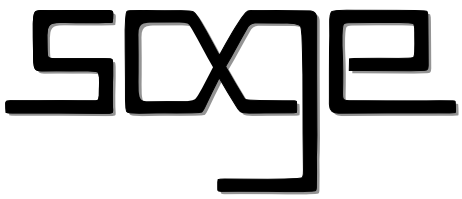
Daarachter kan je berekeningen en opdrachten intypen en uit laten voeren met de `[Enter]`-toets. Bijvoorbeeld

```
sage: 5*2^3-1
39
```

Probeer maar eens wat berekeningen uit.

De notebook interface is (ook thuis) gratis online te gebruiken vanaf `http://www.sagemath.org/`. Een notebook bestaat uit *cellen* waarin commando's ingetypt kunnen worden. Deze cellen kunnen uit meerdere regels bestaan en de commando's in een cel worden uitgevoerd als je op `[Shift]` + `[Enter]` drukt. Zie Figuur 3 voor een voorbeeld. Het voordeel van de notebook interface is dat je de cellen eenvoudig kan wijzigen en opnieuw kan uitvoeren. In de voorbeelden hieronder zullen we alles laten zien zoals het er op de commandoregel uitziet. De tekst '`sage:`' of '`....:`' hoef je nooit in te typen. Om snel handigheid te krijgen met Sage is het aan te raden de commando's uit deze tekst allemaal uit te proberen, naar wens met andere getallen.

Sage heeft veel ingebouwde wiskundige functies. Zo kan je Sage bijvoorbeeld vragen of een getal een priemgetal is.



Figuur 2: SAGE staat voor "Software for Algebra and Geometry Experimentation." Op `http://www.sagemath.org/` kan je documentatie vinden en Sage gratis downloaden of online gebruiken.

```
sage: is_prime(41)
True
sage: is_prime(91)
False
```

Berekeningen die te lang duren kunnen op de commandoregel afgebroken worden met `[Ctrl]+[c]`. In de notebook interface kan je via het menu "Action..." kiezen voor "Interrupt" om berekeningen af te breken. Probeer het maar eens met de volgende opdracht.

```
sage: 123^1000000000
```

Een *Fermatgetal* is een getal van de vorm $2^{2^n} + 1$, waarbij n een niet-negatief geheel getal is. De eerste vijf Fermatgetallen zijn $3, 5, 17, 257$ en 65537; allemaal priemgetallen! Probeer maar eens uit in Sage. Fermat vermoedde dat alle getallen van deze vorm priemgetallen zijn.

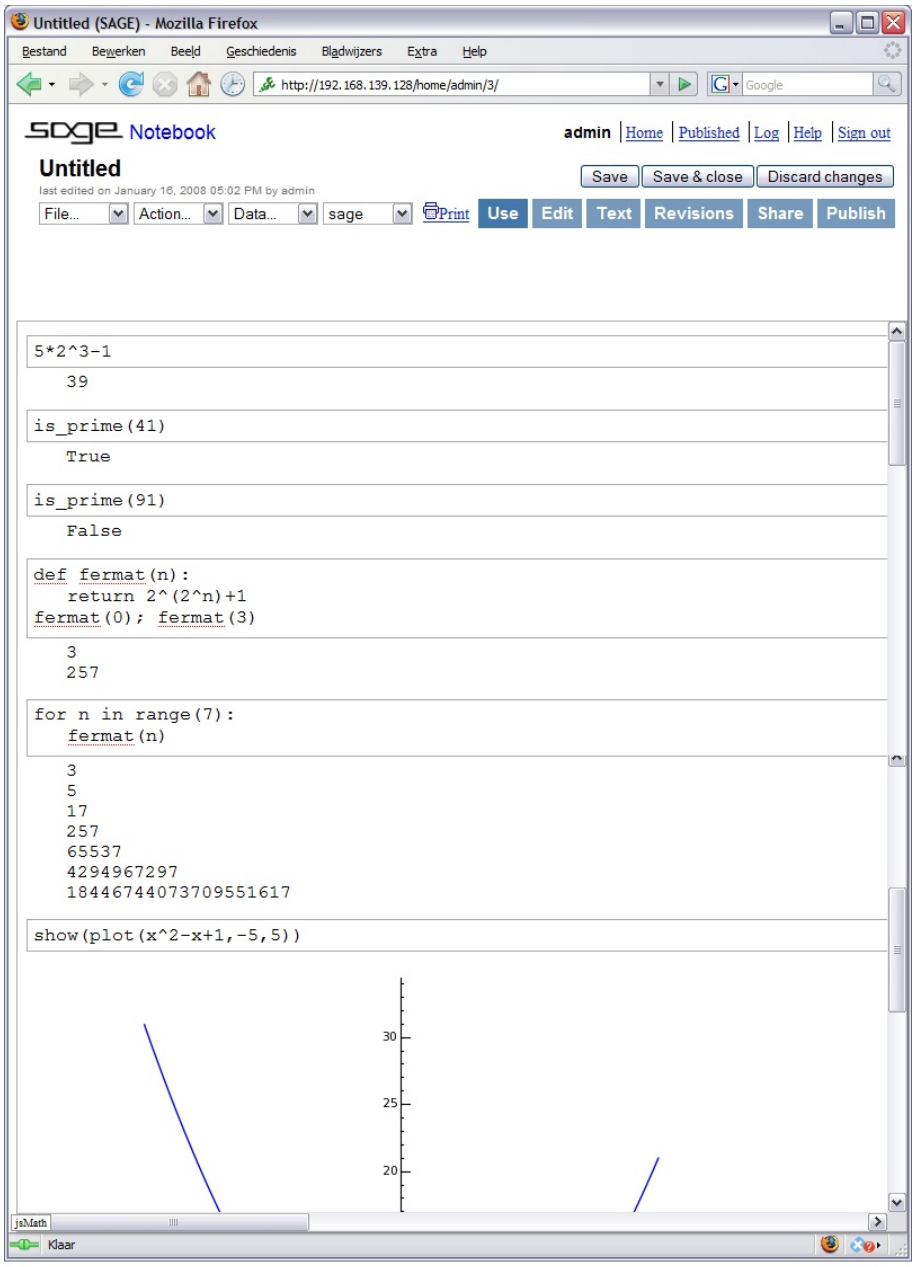
Opgave 1. Had Fermat gelijk?

Het commando `factor` geeft de ontbinding in priemgetallen van een getal.

```
sage: factor(12345678)
2 * 3^2 * 47 * 14593
```

Opgave 2. Voer de onderstaande opdrachten uit (deze keer zonder de getallen te variëren). Het kan best even duren voordat het resultaat van de tweede berekening verschijnt.

```
sage: is_prime(10^70 + 31)
sage: factor(10^70 + 31)
```



Figuur 3: De *notebook interface* van Sage.

Het bepalen van de priemfactoren van een getal is voor de computer een stuk lastiger dan beslissen of een getal priem is of niet.

Sage gebruikt de programmeertaal *Python*. Deze taal kun je gebruiken om zelf functies te definiëren in Sage en om opdrachten te herhalen. Bijvoorbeeld:

```
sage: def fermat(n):
....:     return 2^(2^n)+1
....:
sage: fermat(0)
3
sage: fermat(3)
257
```

Zie ook Figuur 3 voor hoe dit er in de notebook interface uitziet. De eerste regel geeft aan dat we een functie '`fermat`' definiëren waarvan de invoervariabele n heet. Door het inspringen weet Sage wat er binnen de functie hoort en wat niet; dit is dus absoluut noodzakelijk. Met het `return`-commando kunnen we aangeven wat de uitvoerwaarde van de functie is. Sage weet dat de definitie van de functie afgelopen is wanneer het inspringen is afgelopen. Op de commandoregel is dat zodra er op een niet-ingesprongen regel op `[[Enter]` gedrukt wordt.

Opgave 3. Een Mersennegetal is een getal van de vorm $2^n - 1$. Definiëer een functie `mersenne` die voor invoer n de uitvoer $2^n - 1$ geeft.

Het herhalen van een commando kan met de `for`-opdracht. Op de commandoregel is het hierbij nodig dat je zelf op `[backspace]` drukt om de inspringing te beëindigen en zo Sage te vertellen dat de `for`-opdracht over is.

```
sage: for n in range(7):
....:     fermat(n)
....:
3
5
17
257
65537
4294967297
18446744073709551617
```

Figuur 4: Het programma Sage maakt gebruik van de programmeertaal *Python*, één van 's werelds meest gebruikte programmeertalen. Als je programmeert in Sage, leer je dus meteen een taal die ook buiten de wiskunde gebruikt wordt. Python is begin jaren '90 bedacht door Guido van Rossum bij het Centrum voor Wiskunde en Informatica (CWI) in Amsterdam. Voor meer informatie over Python, zie `http://www.python.org/`.

Met deze opdracht wordt '`fermat(n)`' uitgevoerd voor elk geheel getal n van 0 tot en met 7.

In plaats van een `for`-opdracht kan je ook een `while`-opdracht gebruiken om opdrachten te herhalen. Het volgende commando vindt bijvoorbeeld het kleinste priemgetal groter dan 200. Dat gebeurt door te beginnen met $k = n + 1$ en telkens als k niet priem is, k met 1 op te hogen.

```
sage: def volgende_priem(n):
....:     k=n+1;
....:     while not is_prime(k):
....:         k=k+1;
....:     return k
....:
sage: volgende_priem(200)
211
```

Deze functie is echter al in Sage voorgeprogrammeerd, hij heet `next_prime`.

```
sage: next_prime(200)
211
```

Opgave 4. Vind het kleinste priemgetal groter dan je telefoonnummer.

Een *priemgetallentweeling* is een paar $\{p, p+2\}$ van twee priemgetallen p en $p+2$ met verschil 2. Bijvoorbeeld 3 en 5, maar ook 17 en 19. Men vermoedt dat er oneindig veel priemgetallentweelingen bestaan, maar niemand heeft dit ooit kunnen bewijzen. Met de volgende opdracht kunnen we de eerste priemgetallentweeling vinden waarbij beide priemgetallen tenminste n zijn.

```
sage: def tweeling(n):
....:     k=n
....:     while not(is_prime(k) and is_prime(k+2)):
....:         k=k+1
....:     return k, k+2
....:
sage: tweeling(200)
(227, 229)
```

Opgave 5. Vind de kleinste priemgetallentweeling waarbij elk getal 30 cijfers heeft.

Opgaven

6. Vind een priemgetal waarvan de eerste acht cijfers je geboortedatum zijn in de notatie jjjjmmdd.

7. (programmeeropdracht) Een palindroomgetal is een getal dat van achter naar voor hetzelfde is als van voor naar achter, zoals bijvoorbeeld 121 of 5335.

1. Vind het kleinste palindroompriemgetal van 7 cijfers.
2. Vind alle palindroompriemgetallen van 2, 4 of 6 cijfers, wat valt je op?

College 2

Priemgetallen

Getaltheorie

Voordat we in het vijfde college uit zullen leggen hoe RSA werkt, gaan we ons twee colleges lang bezighouden met gehele getallen en hun elementaire eigenschappen.

De studie van getallen of *getaltheorie* heeft een lange geschiedenis. Het getalbegrip zelf komt in alle ontwikkelde culturen voor vanwege zijn evidente praktische waarde. De eerste diepergaande theoretische beschouwingen vinden we in de klassieke Griekse cultuur. De school rond Pythagoras ontwikkelde in de zesde eeuw voor Christus een natuurfilosofie gebaseerd op het gehele getal als 'basis der dingen', en wiskunde was bij de Grieken een essentieel onderdeel van de opvoeding. Hoewel de hedendaagse getaltheorie, de tak van wiskunde die zich met eigenschappen van gehele getallen en generalisaties daarvan bezighoudt, zich losgemaakt heeft van de filosofie, gaat een belangrijk deel van de getaltheorie in deze masterclass terug tot de Grieken. Dit alles is te vinden in het bekende tekstboek *De Elementen* dat Euclides rond 300 voor Christus schreef. Iets modernere wiskunde die ook aan de orde komt werd ontwikkeld door de Franse jurist Pierre de Fermat (1601–1665) en de Zwitser Leonhard Euler (1707–1783), een groot wiskundige en tekstboekenschrijver aan wie wij vrijwel onze volledige moderne notatie te danken hebben.

Getaltheorie heeft lang een aureool van uiterst 'pure wiskunde' gehad, een gebied van grote esthetische verfijning doch zonder praktische toepassingen. Eerst in de laatste dertig jaar is daar verandering in gekomen, niet in de laatste plaats door de ontdekking van het RSA-cryptosysteem in 1976.

Figuur 5: Pierre de Fermat.

Priemgetallen

De verzameling van gehele getallen waarmee we in deze aantekeningen zullen werken is de oneindige verzameling

$$\mathbb{Z} = \{\ldots, -3, -2, -1, 0, 1, 2, 3, \ldots\}$$

die ontstaat door het toevoegen van negatieve getallen aan de verzameling $\mathbb{N} = \{0, 1, 2, \ldots\}$ van niet-negatieve of *natuurlijke* getallen. Sommigen rekenen 0 niet tot de natuurlijke getallen, en 0 is niet-natuurlijk in de zin dat het net als de negatieve getallen een vinding van later datum is die bij de Grieken nog niet voorkomt. Binnen $\mathbb{Z}$ kan men zoals bekend onbeperkt optellen, aftrekken en vermenigvuldigen. De uitkomst van deze bewerkingen worden som, verschil en product genoemd. Een product van $n \geq 1$ gelijke getallen x wordt met x^n aangegeven, de n-de macht van x. We definiëren $x^0 = 1$ voor iedere $x \neq 0$. Algemener is het handig een *leeg product*, d.w.z. een product van nul elementen, gelijk aan 1 te nemen.

We zeggen dat een geheel getal x een deler is van een geheel getal y of y deelt als er een geheel getal z bestaat met $xz = y$. Notatie: $x \mid y$. Ieder geheel getal deelt 0 (neem $z = 0$), maar alle andere getallen hebben slechts eindig veel delers. Een natuurlijk getal n heeft altijd de natuurlijke getallen 1 en n zelf als delers. Een *priemgetal* p is een natuurlijk getal groter dan 1 dat geen andere natuurlijke delers heeft dan 1 en p zelf. Het kleinste priemgetal is 2, en een korte berekening doet vermoeden dat de verzameling

$$\mathcal{P} = \{2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, \ldots\}$$

Figuur 6: Leonhard Euler staat op het Zwitserse bankbiljet van 10 Frank.

van priemgetallen oneindig is. Kenmerkend voor wiskunde is dat men schijnbaar 'voor de hand liggende' waarheden niet accepteert voor een *bewijs* gegeven is. In de getaltheorie is dit nog eens extra gerechtvaardigd, omdat vele 'voor de hand liggende' zaken later onjuist bleken te zijn. In het onderhavige geval is ons vermoeden echter correct.

2.1. Stelling. *Er zijn oneindig veel priemgetallen.*

Bewijs. We geven het klassieke bewijs dat bij Euclides te vinden is. Het leidt een tegenspraak af uit de aanname dat er slechts eindig veel priemgetallen zouden zijn en is daarmee een *bewijs uit het ongerijmde*.

Stel dat de eindige verzameling $\mathcal{P} = \{p_1, p_2, p_3, \ldots, p_N\}$ de verzameling van alle priemgetallen is. Dan definiëren we een getal $n > 1$ door $n = p_1p_2p_3 \ldots p_N + 1$. Laat d de kleinste deler van n groter dan 1 zijn. Dan is d een priemgetal, want iedere deler van d is ook een deler van n. Vanwege onze aanname geldt dus $d = p_i \in \mathcal{P}$ voor zekere i. Nu is $d = p_i$ een deler van n en van $p_1p_2p_3 \ldots p_N$, dus ook van $n - p_1p_2p_3 \ldots p_N = 1$. Dit is een absurde conclusie, dus de aanname dat er slechts eindig veel priemgetallen zijn kan niet correct zijn. □

Eenduidige priemfactorontbinding

Van de bewerkingen $+$ en $\times$ die op de verzameling $\mathbb{Z}$ gedefinieerd zijn is de optelling verreweg de eenvoudigste. De *definitie* van de natuurlijke getallen laat zien dat ieder natuurlijk getal verkregen wordt door het getal 1 herhaald

bij zichzelf op te tellen. (De *lege som* heeft de waarde 0.) Iets formeler kan men zeggen dat 1 een *additieve voortbrenger* is. Voor de vermenigvuldiging liggen de zaken beduidend minder eenvoudig. Het hoofdresultaat van dit college, de *stelling van de eenduidige priemfactorontbinding*, laat zien dat de priemgetallen de *multiplicatieve voortbrengers* van de natuurlijke getallen zijn. Een deel van de uitspraak is het volgende resultaat.

2.2. Lemma. Ieder natuurlijk getal $n \geq 1$ is te schrijven als een product van priemgetallen.

Bewijs. We gebruiken de *methode van het kleinste tegenvoorbeeld*. Deze methode berust er op dat iedere niet-lege verzameling van natuurlijke getallen een kleinste element bezit. We bewijzen de stelling weer uit het ongerijmde: stel dat er natuurlijke getallen zijn die niet te schrijven zijn als product van priemgetallen. Dan bestaat er ook een kleinste element, zeg n, met deze eigenschap. Deze n is niet gelijk aan 1 (het lege product) en ook geen priemgetal, dus hij heeft een deler n_1 met $1 < n_1 < n$. Schrijven we nu $n = n_1 n_2$, dan zijn n_1 en n_2 natuurlijke getallen die kleiner zijn dan n. Wegens onze keuze van n betekent dit dat n_1 en n_2 te schrijven zijn als product van priemgetallen. Vermenigvuldigen we deze beide producten, dan zien we dat n ook een product van priemgetallen is. Tegenspraak met de aanname. □

Het schrijven van een geheel getal als een product van priemgetallen noemt men het *ontbinden* of *factoriseren* van het getal. De uitkomst heet de *priemfactorontbinding* of *factorisatie* van het getal. We hebben nu het bestaan van *een* priemfactorontbinding bewezen voor elk natuurlijk getal. Minder eenvoudig is het te laten zien dat zo'n ontbinding in essentie *eenduidig* is. Hiermee bedoelen we dat als we twee ontbindingen in priemfactoren

$$n = p_1 p_2 \dots p_s = q_1 q_2 \dots q_t$$

van hetzelfde getal n hebben, het altijd zo is dat s en t gelijk zijn en dat de rij priemgetallen $q_1, q_2, \dots, q_t$ op volgorde na *gelijk* is aan de rij $p_1, p_2 \dots, p_s$. De stelling van de eenduidige priemfactorontbinding, die zegt dat dit het geval is, laat zich als volgt formuleren.

2.3. Stelling. *Ieder natuurlijk getal n is op eenduidige wijze te schrijven als een product*

$$n = \prod_{p \in \mathcal{P}} p^{n_p}$$

waarin de exponenten n_p natuurlijke getallen zijn die slechts voor eindig veel priemgetallen p verschillend van 0 zijn.

Het bewijs van deze stelling berust op een aantal eigenschappen van gehele getallen die we eerst zullen bespreken. We introduceren eerst het nuttige begrip *deling met rest*. Met de uitspraak dat we voor twee natuurlijke getallen een dergelijke deling uit kunnen voeren wordt het volgende bedoeld.

2.4. Lemma. Laat a en b natuurlijke getallen zijn, en stel $b \neq 0$. Dan bestaan er natuurlijke getallen q en r zodat

$$a = qb + r \qquad \text{en} \qquad 0 \leq r < b.$$

Bewijs. De verzameling $S = \{a, a-b, a-2b, a-3b, \ldots\}$ bevat natuurlijke getallen (bijvoorbeeld $a \in S$) en *dus* een kleinste natuurlijk getal $r = a - qb$. Omdat $r - b$ kleiner is dan r en ook in S ligt geldt $r - b < 0$, dus we hebben $0 \leq r < b$ als verlangd. □

We noemen het getal r in voorgaand lemma de *rest* van a bij deling door b.

De *grootste gemene deler* van twee gehele getallen a en b die niet beide 0 zijn is per definitie het grootste gehele getal d dat zowel a als b deelt. Notatie: $\gcd(a, b)$. Merk op dat de ggd van a en b gelijk is aan de ggd van de absolute waarden $|a|$ en $|b|$. We zeggen dat a en b *onderling ondeelbaar* zijn als $\gcd(a, b) = 1$. Anders dan velen denken is de berekening van de grootste gemene deler iets waarvoor men geen priemfactorontbinding nodig heeft. De meest efficiënte berekening verloopt als volgt.

2.5. Euclidische Algorithme. *Definieer voor natuurlijke getallen a en b de rij niet-negatieve gehele getallen $r_0, r_1, r_2, \ldots$ door*

$$\begin{aligned} r_0 &= |a| \\ r_1 &= |b| \\ r_{k+1} &= \textit{(rest van } r_{k-1} \textit{ bij deling door } r_k\textit{) als } r_k \neq 0. \end{aligned}$$

Dan bestaat er een k zodat $r_k = 0$, en voor deze k geldt $\gcd(a, b) = r_{k-1}$.

Bewijs. Omdat de getallen in de rij $r_0, r_1, r_2, \ldots$ steeds kleiner worden, maar nooit negatief, moet de gelijkheid $r_k = 0$ optreden voor zekere k. Dan geldt

$r_{k-1} = \text{ggd}(r_{k-1}, 0) = \text{ggd}(r_{k-1}, r_k)$, en omdat we al weten dat $\text{ggd}(a, b) = \text{ggd}(|a|, |b|) = \text{ggd}(r_0, r_1)$ geldt is het verder voldoende te bewijzen dat

$$\text{ggd}(r_0, r_1) = \text{ggd}(r_1, r_2) = \ldots = \text{ggd}(r_{k-1}, r_k).$$

Dit is een herhaalde toepassing van de volgende uitspraak:

> als a en $b \neq 0$ natuurlijke getallen zijn en r de rest van a bij deling door b, dan geldt $\text{ggd}(a, b) = \text{ggd}(b, r)$.

De geldigheid hiervan zien we in door op te merken dat uit de gelijkheid $a = qb + r$ volgt dat ieder getal dat zowel a als b deelt ook $r = a - qb$ deelt, en dat omgekeerd iedere gemeenschappelijke deler van b en r ook $a = qb + r$ deelt. De gemeenschappelijke delers van a en b zijn dus precies de gemeenschappelijke delers van b en r, en in het bijzonder zijn dan de ggd's gelijk. □

Uit de berekening van de ggd leiden we de volgende belangrijke eigenschap af.

2.6. Stelling. *Laat a en b gehele getallen zijn met ggd gelijk aan d. Dan bestaan er $x, y \in \mathbb{Z}$ zodat $xa + yb = d$.*

Bewijs. Definieer de rij r_k van resten als in 2.5. Kies $x_0 = \pm 1$ en $y_0 = 0$ alsmede $x_1 = 0$ en $y_0 = \pm 1$ zodat de vergelijkingen

$$x_0 a + y_0 b = r_0 \ (= |a|)$$
$$x_1 a + y_1 b = r_1 \ (= |b|)$$

gelden. De deling met rest geeft ons getallen q_k zodat $r_{k-1} = q_k r_k + r_{k+1}$ geldt. Dit betekent dat we uit bovenstaande twee vergelijkingen een rij vergelijkingen

$$x_k a + y_k b = r_k \qquad \text{voor } k = 0, 1, 2, \ldots$$

kunnen maken waarin de $(k+1)$-de vergelijking ontstaat door de k-de q_k maal van de $(k-1)$-de af te trekken. Anders gezegd: de getallen x_k en y_k voldoen net als r_k aan de betrekkingen $x_{k-1} = q_k x_k + x_{k+1}$ en $y_{k-1} = q_k y_k + y_{k+1}$. We hebben al gezien dat er een k is waarvoor $r_k = 0$ en $r_{k-1} = d$, en in dat geval is de $(k-1)$-de vergelijking van de verlangde soort. □

2.7. Gevolg. Laat a en b onderling ondeelbare getallen zijn. Dan bestaan er $x, y \in \mathbb{Z}$ zodat $xa + yb = 1$. □

We bewijzen met behulp van het voorafgaande gevolg een eigenschap van priemgetallen waarop we het bewijs van 2.3 zullen baseren.

2.8. Lemma. Laat p een priemgetal zijn dat een product van twee gehele getallen a en b deelt. Dan is p een deler van a of een deler van b (of van allebei).

Bewijs. Als $p \mid a$ geldt is er niets te bewijzen, dus stel dat p geen deler is van a. Dan is 1 de grootste gemeenschappelijke deler van p en a, want p is een priemgetal. Vanwege het zojuist bewezen gevolg bestaan er dus $x, y \in \mathbb{Z}$ zodat $xp + ya = 1$. Vermenigvuldigen we deze vergelijking met b, dan krijgen we

$$xpb + yab = b.$$

Omdat p beide termen in het linkerlid deelt moet ook $p \mid b$ gelden, en dat is precies wat we moesten bewijzen. □

Door het voorafgaande lemma herhaald toe te passen zien we dat een priemgetal p slechts een product $x_1 x_2 \dots x_s$ van gehele getallen kan delen als tenminste één van de getallen x_i deelbaar is door p. Deze eigenschap van priemgetallen zullen we nu gebruiken om de eenduidigheid van de priemfactorontbinding van natuurlijke getallen als geformuleerd in 2.3 te bewijzen. Omdat we het bestaan van zo'n ontbinding al in 2.2 bewezen hebben is daarmee het bewijs van stelling 2.3 voltooid.

We willen bewijzen dat uit een gelijkheid van ontbindingen

$$p_1 p_2 \dots p_s = q_1 q_2 \dots q_t$$

volgt dat de priemgetallen in linker- en rechterlid op volgorde na dezelfde zijn. We mogen wel aannemen dat $s \geq t$ geldt. Omdat p_1 een deler is van $q_1 q_2 \dots q_t$ is er een q_i die deelbaar is door p_1. Omdat q_i geen andere positieve delers dan 1 en q_i zelf heeft volgt hieruit dat $p_1 = q_i$ voor zekere $i \in \{1, 2, \dots, t\}$. Delen we links door p_1 en rechts door q_i, dan krijgen we een gelijkheid met links $s - 1$ priemfactoren en rechts $t - 1$ priemfactoren, en we kunnen ons argument herhalen voor p_2, en dan voor p_3, enzovoort. Na s stappen staat links het lege product 1 en rechts een product van $t - s$ priemen. Omdat geen priemgetal 1 kan delen betekent dit dat rechts ook de priemen 'op' zijn, zodat $t = s$ geldt en de priemen q_i links op volgorde na gelijk zijn aan de priemen p_j links. □

Opgaven

1. Bepaal de ggd van $a =$ je telefoonnummer (zonder netnummer) en $b =$ je geboortedatum (geschreven als 890312 voor bijvoorbeeld 12 maart 1989) en vind $x, y \in \mathbb{Z}$ zodat $xa + yb$ gelijk is aan deze ggd.

2. Vind een oplossing van de vergelijking $123x + 456y = 789$.

3. Laat zien dat de getallen q en r in lemma 2.4 eenduidig bepaald zijn door a en b. Is lemma 2.4 ook waar als we $a \in \mathbb{Z}$ een willekeurig geheel getal nemen?

4. Bewijs dat iedere gemeenschappelijke deler van twee gehele getallen a en b die niet beide nul zijn een deler is van $\text{ggd}(a, b)$.

5. Laat d de ggd van twee gehele getallen a en b zijn die niet beide nul zijn. Bewijs dat a/d en b/d onderling ondeelbaar zijn.

6. Bewijs dat een getal $n > 1$ een priemgetal is als het geen delers d bezit met $1 < d \leq \sqrt{n}$.

7. Bewijs dat $a = \prod_{p \in \mathcal{P}} p^{m_p}$ dan en slechts dan een deler is van $b = \prod_{p \in \mathcal{P}} p^{n_p}$ als $n_p \geq m_p$ voor alle p. Leid hieruit af dat de grootste gemene deler van a en b van de bovenstaande vorm gelijk is aan $d = \prod_{p \in \mathcal{P}} p^{\min(m_p, n_p)}$. Hier staat $\min(m_p, n_p)$ voor het minimum van de getallen m_p en n_p.

8. Laat a en b twee onderling ondeelbare getallen zijn, en stel dat a een deler is van bc. Bewijs dat a een deler is van c.

Computerpracticum

We hebben vorige keer al gezien dat je met `is_prime` kan bepalen of een getal priem is en dat `factor` de ontbinding in priemgetallen van een getal geeft.

Met het commando `divisors` kunnen we een lijst met alle delers van een getal bepalen.

```
sage: divisors(30)
[1, 2, 3, 5, 6, 10, 15, 30]
```

We kunnen een lijst zoals die door het commando `divisors` wordt uitgevoerd ook gebruiken bij het maken van lusjes. Bekijk het volgende voorbeeld.

```
sage: for d in divisors(30):
....:     print d, is_prime(d)
....:
1 False
2 True
3 True
5 True
6 False
10 False
15 False
30 False
```

Opgave 9. Maak een functie die de som van alle delers van een getal bepaalt.

Opgave 10. Een getal heet perfect als de som van alle delers van het getal precies twee keer het getal is. Bijvoorbeeld, 6 is perfect, want $1+2+3+6 = 12$ is gelijk aan $2 \cdot 6$. Bepaal alle perfecte getallen van ten hoogste drie cijfers.

In het college van vandaag is behandeld hoe je de grootste gemene deler van twee getallen bepaalt. Hiervoor gebruiken we de Euclidische algoritme. Een belangrijke stap hierbij is het uitvoeren van deling met rest. Hiervoor is het Sage-commando `divmod`.

```
sage: divmod(7654321,1234567)
(6, 246919)
sage: 1234567 * 6 + 246919
7654321
```

Opgave 11. Voer zelf de Euclidische algoritme uit om de ggd van 1234567 en 7654321 te bepalen. Gebruik Sage om je berekeningen te doen.

Als je alleen de rest bij deling wilt weten en niet ook het quotient, dan kan je ook gebruik maken van de `%`-operator.

```
sage: 7654321 % 1234567
246919
```

De volgende functie bepaalt de grootste gemene deler van de positieve getallen a en b.

```
sage: def ggd(a,b):
    while b > 0:
        r = a % b
        a = b
        b = r
    return a
```

Deze functie is ook al voorgeprogrammeerd in Sage, hij heet `gcd`.

Het berekenen van ggd's gaat heel snel, dat merk je wanneer je hele grote getallen invoert.

Opgave 12. Voer de volgende berekeningen uit.

```
sage: gcd(2^10 - 1. 3^10 - 1)
sage: gcd(2^1000 - 1, 3^1000 - 1)
sage: gcd(2^100000 - 1, 3^100000 - 1)
```

Wanneer getallen a en b grootste gemene deler d hebben, zijn er gehele getallen x en y zodat $ax + by = d$. We kunnen Sage ook vragen deze getallen uit te rekenen, daarvoor gebruiken we het commando `xgcd`.

```
sage: a = 2^100 - 1
sage: a
1267650600228229401496703205375
sage: b = 3^100 - 1
sage: b
515377520732011331036461129765621272702107522000
sage: d, x, y = xgcd(a, b)
sage: d
138875
sage: x
475960879022767897664371255304089361426688566501
sage: y
-117069928296717766293329116968 2
sage: a * x + b * y
138875
```

College 3

Modulair rekenen

In deze paragraaf zullen we zien hoe men met gehele getallen rekent *modulo* een gegeven getal $n \neq 0$. In het dagelijks leven beschouwt beschouwt iedereen wel eens in een of andere vorm getallen modulo een ander getal. Men kan hierbij denken aan het onderscheid even-oneven (getallen modulo 2) of de door een horloge aangewezen tijd van de dag (modulo 12 of 24 uur).

We zeggen dat twee gehele getallen $x, y \in \mathbb{Z}$ *congruent* zijn *modulo* een getal $n \neq 0$ als het verschil $x - y$ deelbaar is door n. We noteren dit als $x \equiv y \mod n$: een *congruentie* modulo n. Merk op dat congruent modulo n en congruent modulo $-n$ hetzelfde is. We zullen daarom meestal $n > 0$ nemen. In termen van deling met rest (voor willekeurige gehele getallen als in opgave 3) kunnen we de definitie ook anders formuleren.

3.1. Lemma. Twee getallen x en y in $\mathbb{Z}$ zijn dan en slechts dan congruent modulo een getal $n > 0$ als hun rest bij deling door n gelijk is.

Bewijs. Schrijf $x = q_1 n + r_1$ and $y = q_2 n + r_2$. Dan geldt $x - y = (q_1 - q_2)n + (r_1 - r_2)$, dus $x \equiv y \mod n$ precies wanneer $n \mid r_1 - r_2$. Omdat $-n < r_1 - r_2 < n$ geldt $n \mid r_1 - r_2$ alleen als $r_1 = r_2$. □

We nemen nu voorlopig een vast getal $n > 0$. De verzameling van alle getallen in $\mathbb{Z}$ die modulo n met een gegeven getal x congruent zijn is de oneindige verzameling

$$\overline{x} = x + n\mathbb{Z} = \{x + nk : k \in \mathbb{Z}\}.$$

Een dergelijke verzameling heet een *restklasse modulo* n. Merk op dat x zelf in de restklasse $\overline{x}$ zit. Twee elementen zitten dan en slechts dan in één

restklasse modulo n als ze congruent zijn modulo n, d.w.z. als ze dezelfde rest bij deling door n hebben. Omdat er n verschillende resten modulo n zijn (namelijk $0, 1, 2, \ldots, n-1$) zijn er precies n restklassen modulo n. Ieder geheel getal is precies in één restklasse modulo n bevat, namelijk de klasse die we met $\overline{x}$ aangaven.

We gaan nu op de voor de hand liggende manier optelling en vermenigvuldiging van restklassen modulo n definiëren:

$$\overline{x} + \overline{y} = \overline{x+y}$$
$$\overline{x} \cdot \overline{y} = \overline{x \cdot y}.$$

Om te zien dat dit een goede definitie is moeten we nagaan dat de uitkomst niet van de keuze van een specifieke x in de restklasse $\overline{x}$ afhangt. Anders gezegd: als x' en y' getallen zijn met $\overline{x} = \overline{x}'$ en $\overline{y} = \overline{y}'$, dan moet gelden dat $\overline{x'+y'} = \overline{x+y}$ en $\overline{x' \cdot y'} = \overline{x \cdot y}$. Om dit in te zien moeten we verifiëren dat de verschillen

$$(x'+y')-(x+y) = (x'-x)-(y'-y) \qquad \text{en} \qquad x'y'-xy = x'(y'-y)+(x'-x)y$$

deelbaar zijn door n als $x'-x$ en $y'-y$ deelbaar zijn door n. Uit de manier waarop we de verschillen opgeschreven hebben is dit echter direct duidelijk.

De verzameling van restklassen modulo n wordt aangegeven met $\mathbb{Z}/n\mathbb{Z}$. Optellen en vermenigvuldigen in $\mathbb{Z}/n\mathbb{Z}$ is niet ingewikkelder dan optellen en vermenigvuldigen in $\mathbb{Z}$. Het belangrijkste verschil is dat de 'getallen' in $\mathbb{Z}/n\mathbb{Z}$ geen eenduidige notatie hebben: $\overline{x} = \overline{y}$ betekent niet dat $x = y$ in $\mathbb{Z}$ geldt. Het geldt wel als we afspreken de restklasse $\overline{x}$ altijd te noteren als $\overline{r}$, met $r \in \{0, 1, 2, \ldots, n-1\}$ de rest van x bij deling door n. In dit geval zeggen we dat we *x reduceren modulo n*.

De optelling in $\mathbb{Z}/n\mathbb{Z}$ zorgt voor weinig verrassingen. Ieder element in $\mathbb{Z}/n\mathbb{Z}$ kan verkregen worden door de restklasse $\overline{1}$ herhaald bij zichzelf op te tellen, zodat we net als voor de natuurlijke getallen een *additieve voortbrenger* hebben. Anders dan in het eerdere geval is $\overline{1}$ niet noodzakelijk de *enige* additieve voortbrenger van $\mathbb{Z}/n\mathbb{Z}$, zie opgave 1. Een ander verschil met $\mathbb{Z}$ is dat de elementen van $\mathbb{Z}/n\mathbb{Z}$ niet zoals in $\mathbb{Z}$ op een voor de hand liggende wijze op grootte geordend kunnen worden. De natuurlijke ordening in $\mathbb{Z}$ is in $\mathbb{Z}/n\mathbb{Z}$ verdwenen omdat iedere restklassen zowel positieve als negatieve getallen bevat. Hier staat echter tegenover dat $\mathbb{Z}/n\mathbb{Z}$ een eindige verzameling is, zodat men ook kan zeggen dat $\mathbb{Z}/n\mathbb{Z}$ eenvoudiger is omdat het anders dan $\mathbb{Z}$ geen willekeurig 'grote getallen' bevat.

De vermenigvuldiging in $\mathbb{Z}/n\mathbb{Z}$ heeft een verrassende eigenschap die in $\mathbb{Z}$ niet optreedt: het product van twee elementen verschillend van het nulelement $\overline{0} \in \mathbb{Z}/n\mathbb{Z}$ kan gelijk zijn aan $\overline{0}$. Een eenvoudig voorbeeld wordt gegeven door de gelijkheid $\overline{2} \cdot \overline{3} = \overline{0} \in \mathbb{Z}/6\mathbb{Z}$. Elementen $\overline{x} \in \mathbb{Z}/n\mathbb{Z}$ verschillend van $\overline{0}$ die de eigenschap hebben dat er $\overline{y} \neq \overline{0}$ bestaat met $\overline{x} \cdot \overline{y} = \overline{0}$ heten *nuldelers*. Dergelijke elementen komen zoals bekend in $\mathbb{Z}$ niet voor.

Een ander belangrijk verschil tussen $\mathbb{Z}$ en $\mathbb{Z}/n\mathbb{Z}$ is dat $\mathbb{Z}/n\mathbb{Z}$ veel meer *eenheden* kan hebben. Hiermee bedoelen we dat er veel elementen $\overline{x} \in \mathbb{Z}/n\mathbb{Z}$ kunnen zijn waarvoor de vergelijking $\overline{xy} = \overline{1}$ oplosbaar is voor zekere $\overline{y} \in \mathbb{Z}/n\mathbb{Z}$. Zo geldt bijvoorbeeld $\overline{2} \cdot \overline{3} = \overline{1} \in \mathbb{Z}/5\mathbb{Z}$. De corresponderende vergelijking in $\mathbb{Z}$ is zoals bekend slechts oplosbaar voor $x = \pm 1$, zodat $\mathbb{Z}$ precies twee eenheden bevat. Als er $\overline{y}$ bestaat zodat $\overline{xy} = \overline{1}$, dan is hij eenduidig bepaald door $\overline{x}$ (opgave 4). Hij heet de *inverse* van $\overline{x}$ in $\mathbb{Z}/n\mathbb{Z}$. We gaan nu bewijzen dat ieder element $\overline{a} \in \mathbb{Z}/n\mathbb{Z}$ verschillend van het nulelement een nuldeler of een eenheid is.

3.2. Stelling. *Laat a een geheel getal zijn met $\overline{a} \neq \overline{0} \in \mathbb{Z}/n\mathbb{Z}$. Dan geldt*

$$\begin{aligned} \gcd(a, n) = 1 &\implies \overline{a} \text{ is een eenheid in } \mathbb{Z}/n\mathbb{Z}; \\ \gcd(a, n) > 1 &\implies \overline{a} \text{ is een nuldeler in } \mathbb{Z}/n\mathbb{Z}. \end{aligned}$$

Bewijs. Als $\gcd(a, n) = 1$ geldt kunnen we wegens gevolg 2.7 getallen $x, y \in \mathbb{Z}$ vinden zodat $xa + yn = 1$. De vergelijking $xa = 1 - yn$ geeft $\overline{xa} = \overline{1}$ in $\mathbb{Z}/n\mathbb{Z}$, dus $\overline{a}$ is een eenheid in $\mathbb{Z}/n\mathbb{Z}$.

Als $\gcd(a, n) = d > 1$ geldt, dan hebben we $\overline{n/d} \neq \overline{0}$. Uit de vergelijking

$$\overline{a} \cdot (\overline{n/d}) = \overline{(na)/d} = \overline{n} \cdot (\overline{a/d}) = \overline{0}$$

zien we nu dat $\overline{a}$ een nuldeler is in $\mathbb{Z}/n\mathbb{Z}$. □

Uit het bewijs zien we tevens hoe we de inverse van een element $\overline{a}$ kunnen berekenen. Hiertoe moeten we de vergelijking $xa + yn = 1$ oplossen, en we hebben in de paragraaf gezien hoe dit met behulp van deling met rest kan geschieden. In het geval van een berekening van de inverse is het niet nodig de coefficient y te berekenen, zodat we alleen de rij van de x_k in het bewijs van 2.6 bij hoeven te houden, niet de rij van y_k.

3.3. Lemma. Voor elke eenheid $\overline{a}$ in $\mathbb{Z}/n\mathbb{Z}$ geldt

$$\{0, \overline{a}, 2\overline{a}, 3\overline{a}, \ldots, (n-1)\overline{a}\} = \mathbb{Z}/n\mathbb{Z}.$$

Bewijs. Het is voldoende om te laten zien dat voor alle gehele getallen i, j met $1 \leq i < j \leq n-1$ geldt $i\overline{a} \neq j\overline{a}$ in $\mathbb{Z}/n\mathbb{Z}$. Dit laatste volgt door de gelijkheid $i\overline{a} = j\overline{a}$ links en rechts te vemenigvuldigen met de inverse van $\overline{a}$. □

Computerpracticum

Getallen in $\mathbb{Z}/n\mathbb{Z}$ hebben geen eenduidige notatie. Om dit te verhelpen kunnen we afspreken elementen $\overline{x} \in \mathbb{Z}$ altijd te noteren als $\overline{r} \in \mathbb{Z}$, met $r \in \{0, 1, 2, \ldots, n-1\}$ de rest van x bij deling door n. Het nemen van de rest bij deling door n wordt *reduceren modulo* n genoemd en wordt in Sage gedaan met het symbool `%`. Na reduceren modulo n kunnen we zien welke getallen modulo n congruent zijn:

```
sage: n = 12345
sage: x = 8657823; y = 3867963; z = 391667
sage: x % n; y % n; z % n
3978
3978
8972
```

Dus in $\mathbb{Z}/n\mathbb{Z}$ geldt $\overline{x} = \overline{y}$ en niet $\overline{x} = \overline{z}$. Modulair optellen en vermenigvuldigen kan eenvoudig uitgevoerd worden door in $\mathbb{Z}$ op te tellen en daaarna opnieuw te reduceren modulo n.

```
sage: (x+z) % n
605
```

Laten we eens wat grote getallen invoeren. De exacte getallen zijn niet belangrijk, maar zorg dat n ongeveer 50 cijfers heeft, dat a vier cijfers heeft en b ongeveer vijfmiljoen is.

```
sage: n = 78541236548952458436871597526841526798426714856719
sage: a = 9234
sage: b = 5432101
```

Vergelijk eens de volgende commando's. Voer eerst het ene commando uit, wacht tot het klaar is en voer dan het andere uit.

```
sage: (a^b) % n
10146644709565963304550406809266705983320185948313
```

```
sage: power_mod(a, b, n)
10146644709565963304550406809266705983320185948313
```

Bij het eerste commando wordt eerst a^b uitgerekend, een getal met meer dan 20.000.000 cijfers, dat dan gereduceerd wordt modulo n. Bij de tussenstappen van deze berekening wordt gerekend met getallen van miljoenen cijfers. Bij het tweede commando wordt bij elke vermenigvuldigingsstap opnieuw gereduceerd modulo n, waardoor er nooit getallen groter dan n vermenigvuldigd hoeven worden. Dat is dus een stuk sneller.

Tenslotte kan $\mathbb{Z}/n\mathbb{Z}$ *eenheden* bevatten. Volgens Stelling 3.2 is $\overline{a} \neq \overline{0} \in \mathbb{Z}/n\mathbb{Z}$ een eenheid als $\text{ggd}(a, n) = 1$. Volgens Stelling 2.6 zijn er dan namelijk $x, y \in \mathbb{Z}$ met $xa + yn = 1$, waaruit volgt dat $\overline{x} \cdot \overline{a} = \overline{1}$ in $\mathbb{Z}/n\mathbb{Z}$. Met behulp van `xgcd` kunnen we x (en y) uitrekenen.

```
sage: d, x, y = xgcd(a, n)
sage: d
1
```

In dit geval is de ggd van a en n gelijk aan 1. Als in jouw voorbeeld $d \neq 1$, dan is a geen eenheid modulo n. Probeer dan een andere a.

```
sage: x*a+y*n
1
```

Voor de uitvoer van de functie `xgcd` geldt altijd $x \cdot a + y \cdot n = d$. Als $d = 1$, dan geldt dus $x \cdot a = 1 - y \cdot n$. Er volgt dan dat $x \cdot a$ congruent is aan 1 modulo n of, in andere woorden, dat $\overline{x} \cdot \overline{a} = \overline{1}$ in $\mathbb{Z}/n\mathbb{Z}$. We noemen $\overline{x}$ dan de inverse van $\overline{a}$ in $\mathbb{Z}/n\mathbb{Z}$.

```
sage: x
5546538916349566617574612166694104356211182668190 0
sage: x*a % n
1
```

Ook deze methode zit al in Sage voorgeprogrammeerd. Probeer maar eens het volgende.

```
sage: (1/a) % n
5546538916349566617574612166694104356211182668190 0
```

Als men dit probeert met een niet–eenheid, dan volgt een foutmelding.

Opgaven

1. Wat is de inverse van $\overline{19}$ in $\mathbb{Z}/65\mathbb{Z}$? Heeft $\overline{26}$ een inverse in $\mathbb{Z}/65\mathbb{Z}$?

2. Bereken de inverse van $\overline{7}$ in $\mathbb{Z}/(10^{100})\mathbb{Z}$.

3. Neem $m = 2^{100}$ en bereken m^m modulo 7.

4. Een zwemmer met constante snelheid die baantjes van 47 seconden zwemt kijkt bij het aantikken steeds naar de secondenwijzer van de klok. Hij begint om :00. Na het eerste baantje tikt hij aan om :47. en na het tweede baantje om :34. Na hoeveel baantjes tikt hij voor het eerst weer aan om :00? En na hoeveel baantjes tikt hij aan om :01?

5. Laat $\overline{a}$ een eenheid zijn in $\mathbb{Z}/24\mathbb{Z}$. Laat zien dat $x^2 = \overline{1}$.

6. Laat p een priemgetal zijn.

1. Voor welke x in $\mathbb{Z}/p\mathbb{Z}$ geldt $x^2 = \overline{1}$ in $\mathbb{Z}/p\mathbb{Z}$? Hint: gebruik dat $x^2 - 1 = (x-1)(x+1)$.

2. De **stelling van Wilson** (naar John Wilson, 1741–1793) zegt dat voor elk priemgetal p geldt:

$$1 \cdot 2 \cdot 3 \cdot \cdots \cdot (p-1) \equiv -1 \bmod p.$$

   Bewijs deze stelling. Hint: als $x \neq x^{-1}$ kunnen we x en x^{-1} tegen elkaar wegstrepen.

3. Laat de omkering zien: als de bovenstaande congruentie geldt voor een geheel getal p, dan is p een priemgetal.

Figuur 7: Abu Ali Al-Hasan ibn Al-Haytham (965–1039) staat op het Irakese bankbiljet van 10 Dinar. Hij maakte al bijna 800 jaar eerder dan Wilson gebruik van de stelling van Wilson!

College 4

De kleine stelling van Fermat

Met het oog op onze cryptografische toepassing in het volgende college gaan we nu kijken naar machtsverheffingen modulo n. Als we met gewone getallen werken is het niet lastig in te zien dat een getal a^b met a en b niet al te groot al snel onopschrijfbaar groot wordt. Van een getal als 100000^{100000} is bijvoorbeeld de uitkomst een 1 gevolgd door een half miljoen nullen, en hieruit volgt dat een getal als 123456^{123456} in decimale schrijfwijze afgedrukt in deze aantekeningen enige honderden pagina's zou beslaan. Willen we echter slechts weten wat de uitkomst is modulo een getal n dat niet al te groot is, dan kunnen we daar vrij snel achter komen *zonder* eerst de exacte waarde van de uitkomst in $\mathbb{Z}$ te berekenen. De methode die men hiertoe gebruikt kwadrateert herhaald modulo n om hoge machten van een element te maken, en rekent vervolgens de gewenste hoge macht als een produkt van de gemaakte hoge machten uit. Daar na iedere kwadratering of vermenigvuldiging het antwoord eerst modulo n gereduceerd kan worden zijn de getallen die in de tussenstappen optreden nooit groter dan n^2, en als n niet heel erg groot is is het geen probleem om zulke getallen op te schrijven. Op het computerpracticum zal dit uitgebreid aan de orde komen.

In onze toepassing in de volgende paragraaf zullen we machtsverheffen modulo getallen n die van een speciale vorm zijn. Voor deze speciale n zullen we nu een aantal stellingen bewijzen die zeggen dat bepaalde machtsverheffingen modulo n de de elementen van $\mathbb{Z}/n\mathbb{Z}$ niet veranderen. (Zulke stellingen bestaan ook voor willekeurige n en zijn bewezen door Euler. We gaan hier verder niet op in.) Het duidelijkste voorbeeld van deze situatie krijgen we als we voor n een priemgetal nemen.

4.1. Kleine stelling van Fermat. *Laat p een priemgetal zijn. Dan geldt*

$$a^p \equiv a \mod p$$

voor iedere $a \in \mathbb{Z}$. Als $\gcd(a,p) = 1$ *geldt tevens*

$$a^{p-1} \equiv 1 \mod p.$$

Bewijs. De eerste congruentie is gemakkelijk te bewijzen als $a \equiv 0 \mod p$. We nemen daarom verder aan dat a niet deelbaar is door p, en dus $\gcd(a,p) = 1$ geldt. We gaan bewijzen dat $a^{p-1} \equiv 1 \mod p$ geldt. De eerste congruentie volgt dan door het resultaat links en rechts met a te vermenigvuldigen.

Uit lemma 3.3 volgt dat in $\mathbb{Z}/p\mathbb{Z}$ geldt

$$a \cdot 2a \cdot \cdots \cdot (p-1)a \equiv 1 \cdot 2 \cdot \cdots \cdot (p-1) \mod p$$

en dus dat

$$(p-1)!a^{p-1} \equiv (p-1)! \mod p.$$

Het ligt nu voor de hand de factor $(p-1)!$ 'weg te delen', doch bij rekenen modulo n kan dit niet altijd. (Voorbeeld: uit $\overline{2} \cdot \overline{4} = \overline{2} \in \mathbb{Z}/6\mathbb{Z}$ volgt niet $\overline{4} = \overline{1}$.) Omdat $(p-1)!$ product van factoren is die onderling ondeelbaar zijn met p is $(p-1)!$ zelf ook onderling ondeelbaar met p, dus zijn restklasse in $\mathbb{Z}/p\mathbb{Z}$ is een eenheid. Kies nu $x \in \mathbb{Z}$ zodat $(p-1)! \cdot \overline{x} = \overline{1} \in \mathbb{Z}/p\mathbb{Z}$. Dan kunnen we onze congruentie met $\overline{x}$ vermenigvuldigen, en dit levert $a^{p-1}(p-1)! \cdot x \equiv (p-1)! \cdot x \mod p$. Omdat $(p-1)! \cdot x \equiv 1 \mod p$ volgt het gewenste resultaat. □

We zien dat bij rekenen modulo een priemgetal de merkwaardige situatie zich voor kan doen dat een machtsverheffing geen effect heeft. Uit de tweede congruentie van voorafgaande stelling volgt bovendien dat bij machtsverheffen modulo p de exponent altijd gereduceerd kan worden modulo $p-1$. Op deze gronden is het soms mogelijk een machtsverheffing ongedaan te maken door een *tweede* machtsverheffing. Hiermee bedoelen we dat bij rekenen modulo n het soms mogelijk is bij een gegeven exponent e een 'inverse exponent' f te vinden zodat

$$(a^e)^f = a^{ef} \equiv a \mod n$$

geldt voor alle a. Op deze fundamentele eigenschap zullen we de cryptografische toepassing in de volgende paragraaf baseren.

4.2. Stelling. *Laat p een priemgetal zijn en $e > 0$ een getal dat onderling ondeelbaar is met $p - 1$. Dan bestaat er een getal $f > 0$ zodat $ef \equiv 1 \mod p - 1$, en voor dergelijke f geldt*

$$a^{ef} \equiv a \mod p$$

voor alle $a \in \mathbb{Z}$.

Bewijs. Vanwege $\gcd(e, p-1) = 1$ is $\overline{e}$ een eenheid modulo $p - 1$, en dat betekent dat we een exponent $f > 0$ kunnen vinden zodat $ef \equiv 1 \mod p-1$. Schrijven we $ef = 1 + k(p-1)$, dan geldt wegens de voorafgaande stelling voor iedere a onderling ondeelbaar met p dat

$$a^{ef} = a^{1+k(p-1)} = a \cdot (a^{p-1})^k \equiv a \cdot 1^k \mod p = a \mod p.$$

Voor $a \equiv 0 \mod p$ is de congruentie ook waar, want aan beide kanten staat dan $0 \mod p$. □

In de volgende paragraaf willen we voorafgaande stelling gebruiken modulo getallen die geen priemgetallen zijn maar het product van *twee* priemgetallen. Een kleine verandering in de voorgaande stelling geeft de gewenste generalisatie.

4.3. Stelling. *Laat $n = pq$ een product van twee verschillende priemgetallen p en q zijn en $e > 0$ een getal dat onderling ondeelbaar is met $(p-1)(q-1)$. Dan bestaat er een getal $f > 0$ zodat $ef \equiv 1 \mod (p-1)(q-1)$, en voor dergelijke f geldt*

$$a^{ef} \equiv a \mod n$$

voor alle $a \in \mathbb{Z}$.

Bewijs. Vanwege de aanname $\gcd(e, (p-1)(q-1)) = 1$ bestaat er weer $f > 0$ met $ef \equiv 1 \mod (p-1)(q-1)$. In het bijzonder geldt dan $ef \equiv 1 \mod p-1$ en $ef \equiv 1 \mod q - 1$, dus de vorige stelling vertelt ons dat voor alle $a \in \mathbb{Z}$ zowel $a^{ef} \equiv a \mod p$ als $a^{ef} \equiv a \mod q$ geldt. Dan is voor alle a het verschil $a^{ef} - a$ zowel door p als door q deelbaar, en dus door $n = pq$. Dit is precies wat we moesten bewijzen. □

We kunnen de uitspraak van de twee voorgaande stellingen ook iets anders formuleren. Als $n = p$ of $n = pq$ als in de stelling en $e > 0$ van de vereiste

vorm, dan is de exponentiatie-afbeelding

$$\begin{aligned}\mathbb{Z}/n\mathbb{Z} &\longrightarrow \mathbb{Z}/n\mathbb{Z}\\ \overline{x} &\longmapsto \overline{x}^e\end{aligned}$$

een afbeelding die een *inverse* van dezelfde vorm heeft, namelijk verheffen tot de macht f. Deze f is de inverse van e modulo $p-1$ of $(p-1)(q-1)$. In het geval $n = pq$ maakt de *berekening* van f gebruik van het feit dat we de factoren p en q van n kennen en daarmee $(p-1)(q-1)$. Met andere woorden: gegeven n en e moeten we de priemfactorisatie van n kennen om f te berekenen. Hiermee is de basis voor het RSA-cryptosysteem gelegd.

Opgaven

1. Laat n een positief getal zijn en a een getal dat onderling ondeelbaar is met n. Bewijs dat $\overline{a}$ een additieve voortbrenger is van $\mathbb{Z}/n\mathbb{Z}$.

2. Toon aan dat n^{n^n} modulo 5 enkel afhangt van de klasse van n modulo 20. Wat is het laatste cijfer van $2008^{2008^{2008}}$?

3. Laat zien dat ieder element in $\mathbb{Z}/17\mathbb{Z}$ verschillend van $\overline{0}$ een macht is van $\overline{3}$. Bepaal voor welke $\overline{a} \in \mathbb{Z}/17\mathbb{Z}$ de vergelijking $\overline{x}^2 = \overline{a}$ oplosbaar is.

4. Zij $\overline{x} \in \mathbb{Z}/n\mathbb{Z}$ gegeven. Laat zien dat er ten hoogste één element $\overline{y} \in \mathbb{Z}/n\mathbb{Z}$ bestaat zodat $\overline{xy} = \overline{1}$.

5. Vind een f zodat $(x^5)^f \equiv x \mod 17$ voor alle x. Idem modulo 91.

Computerpracticum

In het vorige computerpracticum heb je al kennisgemaakt met het commando `power_mod`.

Opgave 6. Bepaal 123^{1008} modulo 1009 met behulp van `power_mod`. Had je dit resultaat ook al kunnen voorspellen? Hint: `is_prime(1009)`.

Met behulp van de kleine stelling van Fermat kan je het berekenen van machten modulo priemgetallen soms een stuk makkelijker maken.

Opgave 7. Wat is 123^{9072} modulo 1009? Gebruik je verstand en niet de computer!

De kleine stelling van Fermat kan ons ook helpen om te beslissen dat een getal niet priem is.

Opgave 8. Voer de volgende berekening uit:

```
sage: a = 10^70 + 31
sage: power_mod(2, a - 1,  a)
```

Is $10^{70} + 31$ priem? Waarom wel/niet?

Opgave 9. Voer de volgende berekening uit:

```
sage: a = 10^70 + 33
sage: power_mod(2, a - 1,  a)
```

Kan je hieruit concluderen dat $10^{70} + 33$ priem is?

Je kan met behulp van de kleine stelling van Fermat alleen met zekerheid vaststellen dat een getal niet priem is. Wanneer de berekening wel uitkomt betekent dit nog niet dat het getal wel priem is. We onderzoeken dit fenomeen nog wat verder:

```
sage: def mod_test(x, p):
....:     return power_mod(x, p, p) == x
....:
```

De functie `mod_test` controleert of x^p gelijk is aan x modulo p.

```
sage: def volledige_test(p):
....:     for x in range(1, p):
....:         if mod_test(x, p) == False:
....:             return False
....:     return True
....:
```

De functie `volledige_test` voert `mod_test` uit voor elke x van 1 tot en met $p-1$. We kijken nu voor welke getallen onder de 1000 de volledige test lukt, en bekijken gelijk ook of het getal priem is of niet:

```
sage: for x in range(2, 1000):
    if volledige_test(x):
        print x, is_prime(x)
```

We gaan nu onderzoeken of er ook een soort kleine stelling van Fermat bestaat modulo n die niet noodzakelijk priem zijn.

Opgave 10. Vind de kleinste positieve k (als die al bestaat) zodat $a^k \equiv 1$ modulo $n = 1001 = 7 \cdot 11 \cdot 13$ voor alle a die onderling ondeelbaar zijn met n.

Opgave 11. Idem maar modulo $n = 125 = 5 \cdot 5 \cdot 5$. Probeer ook zelf een aantal waarden van n, kan je een formule vinden voor k, gegeven de priemfactorisatie van n?

Door te controleren of de kleine stelling van Fermat misgaat, kunnen we vaak zien dat een getal niet priem is. We hebben ook al gezien dat het goed gaan van dit soort testen geen absolute garantie is voor het priem zijn van het getal. We noemen een getal dat (een aantal) van deze of vergelijkbare testen heeft doorstaan een *pseudopriemgetal*.

Opgave 12. Schrijf een functie die, gegeven een getal n, alle a modulo n vindt waarvan het kwadraat 1 modulo n is.

Opgave 13. Voer de functie uit de vorige opgave uit met als invoer een aantal priemgetallen en een aantal niet-priemgetallen. Wat valt je op? Is dit een geschikte (pseudo-)priemtest? Waarom wel/niet?

Opgave 14. Voer de volgende berekeningen uit.

```
sage: power_mod(2, 1001, 2003)
sage: power_mod(17, 6173, 12347)
sage: power_mod(12, 6173, 12347)
```

Kan je de resultaten verklaren? Denk aan de kleine stelling van Fermat en het resultaat uit de vorige opgave.

Opgave 15. Kies voor `n` je favoriete niet-priemgetal van 12 cijfers en voer de volgende berekeningen uit.

```
sage: power_mod(2, (n - 1)/2, n)
sage: power_mod(3, (n - 1)/2, n)
```

Is jouw getal door de mand gevallen?

College 5

Het RSA-cryptosysteem

Factoriseren van gehele getallen is moeilijk

We hebben nu voldoende wiskunde gezien om het RSA-cryptosysteem te kunnen begrijpen. In dit systeem is de sleutel publiek bekend, en tevens is de manier waarop boodschappen gecodeerd worden geen geheim. De veiligheid van het systeem berust op het volgende.

5.1. Ervaringsfeit. Het factoriseren van gehele getallen is moeilijk.

Deze uitspraak vereist enige uitleg, want op het eerste gezicht lijkt factoriseren bijzonder eenvoudig: het is voldoende na te gaan of er onder de getallen $2, 3, 4, \ldots$ tot aan de wortel van n delers van n zitten. Zo gauw men een deler $n_1 > 1$ tegenkomt schrijft men $n = n_1 n_2$, en omdat n_1 als kleinste deler priem is blijft er een kleiner getal n_2 over om te factoriseren. Vindt men geen priemfactoren tot aan de wortel, dan is het te factoriseren getal priem. Dit is een *algoritme* (=methode) die zeker werkt, maar als probleem heeft dat de hoeveelheid werk voor niet al te grote n al snel zo groot wordt dat een supercomputer er meer tijd voor nodig heeft dan de leeftijd van het heelal. Een dergelijke methode is daarom niet praktisch. Het vak *complexiteitstheorie*, dat op de grens van informatica en wiskunde ligt, houdt zich bezig met de analyse van de snelheid en het geheugengebruik van algoritmen. Heel onnauwkeurig gezegd zijn in deze theorie de 'goede' algoritmes de algoritmes waarvan het opschrijven van de gegevens (in ons geval n) en het berekenen van de gevraagde oplossing (de factorisatie van n) hoeveelheden werk van 'dezelfde orde van grootte' zijn. Dit is een notie die redelijk correspondeert met 'in de praktijk bruikbaar'. Problemen waarvoor 'goede' algoritmen bestaan

zijn optellen, vermenigvuldigen, ggd bepalen en machtsverheffen of inverteren modulo n. Voor andere problemen, zoals machtsverheffen in $\mathbb{Z}$, bestaan geen 'goede' algoritmen vanwege het simpele feit dat het opschrijven van het antwoord al te veel tijd kost.

Voor factorisatie zijn geen algoritmen bekend die 'goed' zijn in de boven gesuggereerde zin. Het is wel zo dat de moderne wiskunde methoden opgeleverd heeft die *veel* beter zijn dan het domweg proberen van delers. Een tamelijk recent succes is de factorisatie in 1990 door A.K. Lenstra, H.W. Lenstra, M. Manasse and J. Pollard van het zogenaamde *negende Fermatgetal*

$$\begin{aligned} 2^{2^9}+1 =& 1340780792994259709957402499820584612747936582059239 \\ & 337772356144372176403007354697680187429816690342769 \\ & 003185818648605085375388281194656994643364900608409 7 \end{aligned}$$

als een product van 3 priemgetallen p_7, p_{49} and p_{99} van 7, 49 en 99 cijfers:

$$\begin{aligned} p_7 =& 2424833, \\ p_{49} =& 7455602825647884208337395736200454918783366342657, \\ p_{99} =& 7416400626275308015247871419019374740599407810975 \\ & 19023905821316144415759504705008092818711693940737. \end{aligned}$$

Hierbij werden vele computers parallel gebruikt over de hele wereld, en de algoritme maakte gebruik van getaltheorie in zogenaamde *algebraïsche getallenlichamen*. Bij getallen van dit formaat ligt nu ongeveer de grens van het mogelijke. In het bijzonder is niemand op dit moment in staat een getal van 200 cijfers zonder kleine factoren, bijvoorbeeld een product van twee priemgetallen van 100 cijfers elk, in factoren te ontbinden. Er zoeken echter veel mensen naar snelle factorisatiealgoritmen, en het is gegeven de vooruitgang in de laatste twintig jaar niet te zeggen waar de grens van het mogelijke over zeg tien jaar zal liggen.

Grote priemgetallen zijn gemakkelijk te maken

Op het eerste gezicht is het misschien verrassend te horen dat het heel snel vast te stellen is of een getal van 200 cijfers een priemgetal is. Hiertoe is

het niet nodig alle mogelijke delers te proberen. Immers, de kleine stelling van Fermat (4.2) vertelt ons dat *als* n een priemgetal is, voor alle $a \in \mathbb{Z}$ de congruentie $a^n \equiv a \mod n$ geldt. Voor vaste a is dit snel te controleren, en een enkele waarde van a waarvoor de congruentie niet geldt is voldoende om te concluderen dat n niet priem is. Het is echter zo dat een dergelijke a ons niets vertelt over de factoren van n. Als willekeurig gekozen waarden van a steeds aan de congruentie voldoen krijgt men al snel het vermoeden dat n priem is. Om een echt *bewijs* van dit feit uit zulke congruenties af te leiden is gebruikt men veel geavanceerdere getaltheorie die voor *cyclotomische lichamen* ontwikkeld is. We kunnen hier niet nader op ingaan, en volstaan met de opmerking dat met de beschikbare algoritmen sinds kort van getallen van meer dan 1000 cijfers primaliteit bewezen kan worden. Het factoriseren van de meeste getallen van dergelijke grootte is met de nu beschikbare kennis volstrekt onmogelijk.

Een gevolg van een en ander is dat het makkelijk is grote priemgetallen te maken. Er is de zogeheten *priemgetalstelling* die 2.1 verscherpt in de zin dat hij niet alleen de uitspraak doet dat er oneindig veel priemen zijn, maar bovendien vertelt hoeveel priemen er ongeveer te vinden zijn tot aan een gegeven groot getal. Dit geeft een indicatie hoe 'dicht' de priemen liggen rond getallen van een gegeven grootte, en het verklaart het volgende.

5.2. Feit. Grote priemgetallen zijn gemakkelijk te maken.

RSA

We zullen nu aangeven hoe men op het feit dat grote priemen makkelijk te maken zijn maar grote getallen niet gemakkelijk te factoriseren een cryptosysteem kan baseren. We nemen aan dat Alice aan Bob een geheime boodschap wil sturen, en dat deze boodschap uit een aantal getallen van niet meer dan 600 cijfers bestaat. Hiertoe kan men bijvoorbeeld alle woorden uit de boodschap 'vernummeren' door een eenvoudige substitutie ($a = 01$, $b = 02$, $c = 03$, enzovoort) toe te passen en vervolgens het verkregen lange getal in blokjes van 600 cijfers op te hakken.

In het RSA-systeem is het zo dat iedere gebruiker een *publieke modulus* $n = pq > 10^{600}$ bezit die hij zelf gemaakt heeft door twee grote priemgetallen p en q van meer dan 300 cijfers te vermenigvuldigen. De factoren p en q vertelt hij aan *niemand*, en dat betekent met de huidige stand van zaken dat niemand anders dan hij achter de waarde van p en q kan komen. Voorts heeft

hij een publieke exponent e die zo gekozen is dat e en $(p-1)(q-1)$ onderling ondeelbaar zijn. Of de ggd van e en $(p-1)(q-1)$ inderdaad 1 is kan de gebruiker met de Euclidische algoritme snel verifiëren

Wil nu Alice aan Bob een geheime boodschap N van 600 cijfers sturen, dan doet zij het volgende. Ze zoekt de publieke modulus n en de publieke exponent e van Bob op en stuurt hem het getal N^e, gereduceerd modulo n. Zoals we zagen is N^e een vreselijk groot getal, maar als we slechts de waarde ervan modulo n willen weten is dat snel uit te rekenen. Wil het systeem nu werken, dan dient het zo te zijn dat behalve Bob niemand anders uit de waarde van $N^e \mod n$ de waarde van $N \mod n$ af kan leiden. (Merk op dat we met $N \mod n$ ook N zelf hebben omdat we $0 < N < n$ kiezen.) Dit berust op het feit dat de enige bekende manier waarmee men praktisch de waarde van $N \mod n$ uit de waarde van N^e af kan leiden bestaat uit het vinden van de 'inverse exponent' f die wegens stelling 4.3 bij e hoort. Andere bekende methodes, zoals het proberen van alle restklassen modulo n, kosten veel meer tijd dan er ooit kan zijn. Het vinden van f bestaat uit het vinden van de inverse van e modulo $(p-1)(q-1)$, en dit kan niemand anders dan Bob, omdat het vinden van de waarde van $(p-1)(q-1)$ uit de waarde van n even moeilijk is als het vinden van p en q (opgave 1). Het voordeel van deze methode is dat er geen geheime sleutels uitgewisseld hoeven te worden en Bob zijn eigen veiligheid kan garanderen door zorgvuldig de waarden van p, q en f geheim te houden.

Er is nog een verfijning mogelijk waarbij Alice tevens *bewijst* dat zij het is die de boodschap verstuurd heeft en niet een bedrieger die pretendeert Alice te zijn. Men kan hierbij denken aan de situatie dat Bob de bank is en Alice een persoon die betalingsopdrachten verstuurt. In deze verfijning hebben we niet alleen de openbare exponent e_B, de geheime inverse exponent f_B en het getal $n = n_B$ van Bob nodig, maar tevens de corresponderende waarden e_A, f_A en n_A voor Alice. Wat Alice doet om de geheime boodschap N te vercijferen is eerst N vervangen door een getal $M < n_A$ met $M \equiv N^{f_A} \mod n_A$. Dan zendt ze M over als voorheen, dat wil zeggen ze stuurt de waarde $M^{e_B} \mod n_B$ naar Bob. Nu kan Bob hier als voorheen de waarde van M uit afleiden, en dus kent hij het getal $N^{f_A} \mod n_A$. Om nu de originele boodschap te verkrijgen neemt Bob de publieke exponent e_A en rekent modulo de publieke modulus n_A de e_A-de macht van $N^{f_A} \mod n_A$ uit. Dit is $N^{e_A f_A} = N \mod n_A$, en dit geeft hem N. Bovendien weet Bob nu dat de boodschap van Alice komt, want niemand anders is in staat een boodschap tot de geheime exponent f_A te verheffen modulo n_A behalve Alice

Figuur 8: De gegevens op magneetstrips van bijvoorbeeld bankpasjes en creditcards zijn eenvoudig te kopiëren. Daarom wordt steeds meer gebruikgemaakt van chips die hun gegevens alleen in gecodeerde vorm vrijgeven. Een voorbeeld daarvan is de EMV-chip die op steeds meer creditcards te vinden is en waarmee in het buitenland al veel betaald wordt. De protocollen van de EMV-chip zijn gebaseerd op het RSA-cryptosysteem (`http://www.emvco.com/`).

zelf.

Opgave

1. Stel dat we behalve de waarde van $n = pq$ ook de waarde van $(p-1)(q-1)$ kennen. Laat zien dat we hieruit gemakkelijk de waarden van p en q af kunnen leiden.

Computerpracticum

We gaan modulair rekenen gebruiken om boodschappen te coderen en decoderen. Boodschappen bestaan uit tekst, terwijl modulair rekenen met getallen werkt. De functie `texttonum` kan tekst in een getal omzetten en de functie `numtotext` kan zo'n getal weer naar de oorspronkelijke tekst terugvertalen. Iedere letter wordt in een getal van twee cijfers omgezet ($a = 01,\ b = 02, \ldots, z = 26,\ A = 27, \ldots$, etc.) en zo ontstaat vanzelf uit een tekst een groot getal. Deze functies staan in het bestand `sageinit` dat geladen moet worden voordat deze functies gebruikt kunnen worden. Om deze functies te kunnen gebruiken moet het volgende commando worden ingevoerd.

```
sage: execfile("sageinit")
```

Bijvoorbeeld:

```
sage: texttonum('L')
38
sage: texttonum('e')
5
sage: texttonum('Leiden')
380509040514
```

De getallen van de losse letters worden achter elkaar gezet (waarbij we met twee cijfers werken, dus 01 invullen voor `a` en niet 1).

We zullen nu zien hoe we Bob volgens het RSA–systeem gecodeerde boodschappen kunnen sturen, die alleen hij kan decoderen. Bob begint met het maken van twee priemgetallen. Gebruik zelf gerust andere priemgetallen.

```
sage: p=next_prime(5799415324)
sage: q=next_prime(8941579412)
sage: p; q
5799415327
8941579447
```

Uit deze twee priemgetallen berekent Bob vervolgens getallen n en e. Omdat e moet voldoen aan $\gcd(e,(p-1)(q-1)) = 1$ probeert Bob een paar waarden, totdat zo'n e gevonden is.

```
sage: n=p*q; n
51855932892519984169
sage: e=21561237
sage: while gcd(e,(p-1)*(q-1)) > 1:
....:     e = e+1
....:
sage: e
21561239
```

Het getal n is de publieke modulus en het getal e de publieke exponent. Deze twee getallen maakt Bob aan iedereen bekend, die hem gecodeerde boodschappen wil sturen. Voor hemzelf berekent Bob ook nog de inverse exponent van e. Omdat $\gcd(e,(p-1)(q-1))$ één is, is e modulo $(p-1)(q-1)$ een eenheid en kan een inverse worden berekend:

```
sage: f=(1/e)%((p-1)*(q-1)); f
43048528428912454031
```

Deze f houdt Bob samen met de priemgetallen p en q geheim.

We laten nu Alice de boodschap 'LAPP-Top' coderen:

```
sage: b=texttonum('LAPP-Top'); b
3827424275461516
sage: c=power_mod(b,e,n); c
43874599464765961781
```

Dit getal c stuurt Alice naar Bob. Ieder ander mag dit getal weten, maar zal het niet kunnen decoderen, zolang f onbekend is. Bob kent f wel en decodeert de boodschap als volgt:

```
sage: d=power_mod(c,f,n); d; numtotext(d)
3827424275461516
'LAPP-Top'
```

De truc zit hem er dus in, dat de e–de macht nemen modulo n (door Alice) ongedaan kan worden gemaakt met de f–de macht nemen modulo n (door Bob). Omdat n en e algemeen bekend worden gemaakt, is het voor afluisteraar Eve voldoende om f te weten, om ook de boodschap te decoderen. In dit geval kan Eve dit omdat n klein is:

```
factor(n)
5799415327 * 8941579447
```

Hiermee heeft Eve de priemgetallen p en q bepaald en kan vervolgens net als Bob boven deed f uitrekenen. Wil Bob dus voorkomen dat een ander de voor hem bestemde berichten decodeert, dan zal hij p en q zo groot moeten kiezen, dat een ander n niet kan factoriseren. Dat is geen probleem. Met `next_prime` kan Bob eenvoudig en snel twee priemgetallen van driehonderd cijfers maken. Het product n heeft dan zeshonderd cijfers en iemand moet meer geluk hebben dan er in het heelal aanwezig is, om deze n nog te kunnen factoriseren (met de op dit moment op aarde bekende methoden).

Opgaven

Opgave 2. Stel Bob heeft publieke modulus $n = 1022117$ en publieke exponent $e = 5$.

a) Hoe ziet nu de vercijferde vorm van de boodschap $N =$ 'RSA' $= 444527$ eruit die we aan Bob kunnen sturen?

b) Kraak vervolgens deze code door n te factoriseren en de inverse exponent f van e te berekenen. Controleer de kraak door het vercijferde bericht met behulp van f te decoderen.

Opgave 3. Stel Bob heeft publieke modulus $n = 15241580725499173$ en publieke exponent $e = 1291$. Kraak deze code en ontcijfer het gecodeerde bericht 4974560039093858.

Opgave 4. Gegeven zijn vier personen B_1, B_2, B_3 en B_4. Zij hebben ieder een publieke modulus n_i, publieke exponent e_i en een gecodeerde boodschap c_i $(i = 1, 2, 3, 4)$. De precieze waarden zijn als volgt.

$n1 = 198723412031172002265844037518961615973007117550485870706016515639019$
$928693192999569999695413713988403222384504058111424624528848576558212$
$432988039309167577809742053687865845916524957685158779793894 03$

$e1 = 44310441721180166418378635699 9$

$c1 = 101940698891229399448513698171984371575583686747327920414537383537205$
$196838598381881242511900378291232944793109907926787679032181122886810$
$511814031013668482363919097902228624458911112798857172782354 92$

$n2 = 913438449622001104724817144250163334288413307256323100879810149627555$
$923353443121924827225869857133506010389200252072830678650841252252944$
$062080388412550027853707916082313372591763767082648682455308 9$

$e2 = 882293940804389818582978254793$

$c2 = 167091276459601490675171996464238595059930193923008702198431395671536$
$703557215063178200832274290013064227318323279678475584992085906203406$
$270186274695727907104222980499091057347132447273199466222043 3$

$n3 = 899298305077293021301990442604666796973145266939277594720089875739285$
$156971643169220375418121267687304793051765074306137521594333699440063$
$856114020508274658993641019675516088206955396727026162925544 57$

$e3 = 882293940804389818582978254793$

$c3 = 251789205880461380086641998598981328988953271903203360045548345431940$
$656645285701685969294386880012349855868366370571988397904126907069094$
$992828405494074313595156452782906756270361807469348043352263 09$

$$
\begin{aligned}
n4 &= 162178563231038010102673405887815669935833214220125596940813967629759\\
&\quad 144914170132483644699938793493514527659576486157723312137833022641090\\
&\quad 478600195582216179909720544517667284048235204630210968827293 61\\
e4 &= 882293940804389818582978254793\\
c4 &= 138638511949422690254719181425807283446703191888100806095505781343043\\
&\quad 829294223285548238330322577963308408137556893318421777772110273531797\\
&\quad 126417604013818953672516702032846490796423405087919811189559 74
\end{aligned}
$$

Met de volgende opdracht worden alle bovenstaande waarden automatisch toegekend, zodat je ze niet zelf in hoeft te typen.

```
sage: execfile("opgave")
```

Omdat n_1, n_2, n_3 en n_4 nu te groot zijn om te factoriseren, lukt het niet een van de codes te kraken en dus ook niet een van de berichten c_1, c_2, c_3 of c_4 te ontcijferen.

In het bestand `opgave` is ook een functie `orakel()` voorgeprogrammeerd die één van de geheime priemgetallen lekt.

```
sage: p=orakel();
```

De opdracht luidt nu: zoek uit welke van de vier codes je kan kraken met het gekregen priemgetal en decodeer de boodschap.

College 6 en 7

Secure Computation: Alice en Bob en de Privacy Ring

Introductie

Bij *secure computation* gaat het niet om het beveiligen van een communicatiekanaal tegen kwaadwillende buitenstaanders, maar om een wezenlijk ander probleem. Het gaat namelijk bijvoorbeeld om partijen die elkaar niet of onvoldoende vertrouwen en toch willen samenwerken, en daarbij dus zo min mogelijk geheime informatie willen uitwisselen. Kort gezegd, er zijn n partijen $P_1, \ldots, P_n$, en elke P_i heeft zijn eigen geheime informatie x_i. Secure computation behelst het probleem om voor een gegeven functie f de waarde

$$y := f(x_1, \ldots, x_n)$$

in n variabelen uit te rekenen zodat alle partijen de correcte waarde y te weten komen, terwijl de individuele waarden x_i geheim blijven. Iets preciezer, iedere partij P_i leert niets meer over de waarden x_j van de overige partijen P_j dan hij uit de waarde y en zijn eigen waarde x_i kan afleiden. Meer in het algemeen kan er geëist worden dat dit zelfs geldt voor (zeer kwaadwillende) samenspanning van twee of meer partijen. Het punt van secure computation is natuurlijk dat de partijen dit kunnen afhandelen *zonder* gebruik te maken van een *trusted third party*, een buitenstaander die ze allen zouden moeten kunnen vertrouwen.

Een van de vele voorbeelden is verkiezingen. Namelijk, bij verkiezingen willen we wél de correcte uitslag weten (welke kandidaat heeft de meeste

stemmen gekregen?), maar doorgaans willen we dat individuele stemmen *geheim* kunnen blijven. Hier is de functie f bijvoorbeeld de functie die gegeven de individuele stemmen (de x_i's) de kandidaat y aanwijst met de meeste stemmen. Een ander praktisch voorbeeld is *privacy-beschermende data-mining*. Iedere partij heeft dan zijn eigen geheime database, en het doel is om statistisch onderzoek te doen op die gezamelijke databases, maar zonder de individuele databases aan elkaar te openbaren.

Een meer theoretisch voorbeeld van secure computation is het *zero knowledge bewijs*. Met behulp van cryptografie is het mogelijk om een scepticus te overtuigen van de waarheid van zekere typen beweringen *zonder daarbij het bewijs prijs te geven*. Hierbij heeft er maar één partij geheime informatie, namelijk het bewijs. De functie f komt in wezen neer op een systematische manier om een vermeend bewijs voor de onderhavige stelling te controleren. De functie-waarde y is dan 0 als het bewijs incorrect is, en 1 als het bewijs correct is. Naast encryptie en digitale handtekeningen vormen zero knowledge en andere basistechnieken uit secure computation belangrijke bouwstenen bij het ontwerp van cryptografische toepassingen.

We gaan hier kijken naar een cryptografisch probleem dat wel iets weg heeft van een "zero-knowledge bewijs." Het is een enigszins gecomponeerd probleem, maar de oplossing is wel degelijk gebaseerd op een echt bestaande methode voor zero knowledge bewijzen. De oplossing zoals die hier wordt gepresenteerd maakt gebruik van allerlei elementaire begrippen uit bijvoorbeeld algebra, getaltheorie en combinatoriek, en die zullen we in detail behandelen. Ook zullen we enige andere toepassingen van die begrippen bekijken, zoals *fouten-corrigerende codes* (die betrouwbare satelliet-communicatie mogelijk maken, en ook bij de cd en de dvd een grote rol spelen) en *secret sharing*, een fundamentele techniek bij secure computation.

Een prikkelend bewijs van expertise

Alice en Bob zijn gestrand op een verder onbewoond maar zonnig eiland, en hebben behalve elkaar ook een complete versie van de Encyclopedia Brittanica (of, vooruit, wireless internet access zodat ze Wikipedia kunnen raadplegen) tot de beschikking. Daarmee gaan ze zich verpozen totdat ze door de reddingsploeg worden opgehaald.

Bob beweert tegen Alice dat hij encyclopedische kennis heeft op een aantal gebieden. En Alice ontvangt die bewering met enige scepsis, en vermoedt een zonnesteek. Natuurlijk kan Alice hem erover ondervragen, met de ency-

clopedie in de hand. Maar om het vermaak te vergroten en een nog grotere indruk te maken op Alice, beweert Bob dat hij een expert is op het onderwerp Kunst en Cultuur *of* op het onderwerp Wiskunde en Natuurwetenschappen, dat wil zeggen, hij claimt expertise in *tenminste* één van de twee. Hij wil echter *niet* prijsgeven wat het geval is: de één, de ander of wellicht beide.

Kunnen Alice en Bob het zó spelen dat de sceptische Alice aan het eind overtuigd is dat Bob inderdaad een expert is in minstens één van de twee maar geen idee heeft wat precies het geval is? En dat Bob zo goed als zeker door de mand valt als hij liegt?

Laten we de twee onderwerpen A en B noemen. Alice destilleert heel eerlijk 100 moeilijke vragen over A en 100 moeilijke vragen over B uit de encyclopedie, en geeft die weer op twee van 0 tot en met 99 genummerde lijsten. Laten we even voor het gemak aannemen dat het vragen zijn waarbij gokken naar het antwoord geen zin heeft als je het antwoord niet reeds weet. Een wat moeizamer alternatief voor Alice is het samenstellen van 100 verschillende multiple-choice examens per onderwerp.

In de eerste stap van het protocol kiest Bob een willekeurig getal x in het bereik $0, 1, \ldots, 99$. Zeg dat Bob geen expert is in A. Dan schrijft hij

$$A : x$$

op een speelkaart, en deponeert de kaart in de schoenendoos. Dit gebeurt zó dat Alice ziet dat er één kaart in verdwijnt, maar niet kan zien wat er op die kaart geschreven staat.

Bob krijgt nu de lijsten en zoekt vraag $A : x$ op. Vervolgens mag hij even spieken in de encyclopedie, terwijl Alice zich afwendt. Bob zoekt natuurlijk razendsnel het antwoord op vraag x over onderwerp A af. Maar hij krijgt niet veel tijd van Alice, dus het lukt hem niet om daarnaast nog veel andere antwoorden op te zoeken.

Vervolgens kiest Alice een *challenge* e in het bereik $0, 1, 2, \ldots, 99$ en maakt deze aan Bob bekend. Bob berekent nu

$$y = e - x \pmod{100},$$

schrijft

$$B : y$$

op een speelkaart, en deponeert die op gelijke wijze in de schoenendoos. Bob schudt de doos hevig, en opent deze. De twee speelkaarten komen tevoorschijn, en Bob beantwoordt netjes vraag $A : x$ *en* vraag $B : y$. Alice

controleert de antwoorden in de encyclopedie, en verifieert ook of

$$e = x + y \pmod{100}.$$

Als dat allemaal klopt, accepteert Alice de claim van Bob. De crux hierachter is dat een vraag over een onderwerp waarop Bob geen expert is, volstrekt willekeurig over het rijtje met 100 verschillende vragen over dat onderwerp verdeeld is. Bob zal maar weinig van die antwoorden weten, en die tijdelijke toegang tot de encyclopedie zal hem ook niet veel geholpen hebben. Dus, bijvoorbeeld, als hij in totaal 5 vragen had kunnen beantwoorden op dat onderwerp, was hij met 95 procent kans door de mand gevallen. Door met een nog groter aantal vragen te werken kan de betrouwbaarheid natuurlijk verder vergroot worden.

Opgave 1. Geef een goede reden waarom Bob zich dient vast te leggen op één van de twee onderwerpen tesamen met een vraag-nummer voordat hij van Alice de challenge e ontvangt.

Het wordt interessanter wanneer Bob claimt expert te zijn op het gebied van bijvoorbeeld, een meerderheid van een groter aantal gegegeven onderwerpen, zeg tenminste 6 uit 11, en hij Alice daarvan wil kunnen overtuigen zonder prijs te geven welke onderwerpen. Om dat op te kunnen lossen, hebben ze speciale *codes* nodig. Die gaan we nu bestuderen.

Interpolerende combinatorische codes

6.1. Definitie. Zij V een verzameling. De cardinaliteit, dat wil zeggen het aantal elementen, van V wordt genoteerd met $|V|$. Een verzameling heet eindig als $|V| < \infty$. De notatie $v \in V$ betekent dat v een element is van V.

6.2. Definitie. De doorsnede, respectievelijk de vereniging, van twee verzamlingen V en W wordt genoteerd met $V \cap W$, respectievelijk $V \cup W$. Als V een deelverzameling is van W dan wordt dit genoteerd met $V \subset W$.

6.3. Definitie. Als V een eindige verzameling is met n elementen, dan wordt het aantal verschillende deelverzamelingen met precies k elementen genoteerd met $\binom{n}{k} (= \frac{n!}{k!(n-k)!})$.

6.4. Definitie. Zij V en W niet-lege verzamelingen. Het Carthesisch product $V \times W$ van V en W is de verzameling bestaande alle mogelijke geordende paren (v, w) met $v \in V$ en $w \in W$.

Het volgende lemma is triviaal.

6.5. Lemma. Als V en W eindige verzamelingen zijn, dan $|V \times W| = |V| \cdot |W|$.

6.6. Definitie. Zij n een positief geheel getal en zij V een niet-lege verzameling. Dan is V^n de verzameling die verkregen wordt door $n-1$ maal het Carthesisch product van V met zich zelf te nemen. V^n bestaat dus uit alle mogelijke geordende rijtjes van de vorm $(v_1, v_2, \ldots, v_n)$ met $v_i \in V$ voor iedere $i \in \{1, \ldots, n\}$.

Zij V een niet-lege verzameling en zij n een positief geheel getal. Schrijf $I := \{1, \ldots, n\}$ (I voor *index*), en neem $(v_i)_{i \in I}$ als een verkorte notatie voor het element $(v_1, \ldots, v_n) \in V^n$. Als A een niet-lege deelverzameling is van I en $\overline{v} \in V^n$ met $\overline{v} = (v_1, \ldots, v_n)$, dan is $\overline{v}_A$ gedefinieerd als het geordende rijtje $(v_i)_{i \in A}$. Met andere woorden, $\overline{v}_A$ wordt verkregen uit $\overline{v}$ door alleen die coördinaten te behouden met $i \in A$. Bijvoorbeeld, als $\overline{v} = (v_1, v_2, v_3, v_4, v_5) \in V^5$ en $A = \{2, 3, 5\}$, dan $\overline{v}_A = (v_2, v_3, v_5) \in V^3$. p

We gaan nu kijken naar een klasse deelverzamelingen C met een speciale combinatorische structuur.

6.7. Definitie. Zij V een niet-lege verzameling en zij t en n positieve gehele getallen met $1 \leq t < n$. Zij C een niet-lege deelverzameling van V^n. Dan heeft C de $(t+1)$-interpolatie eigenschap als voor iedere niet-lege deelverzameling A van I met $|A| = t+1$ en voor iedere $w \in V^{t+1}$ er een unieke $c \in C$ bestaat zodanig dat $c_A = w$.

Bijvoorbeeld,

$$C = \{(0,0,0), (0,1,1), (1,0,1), (1,1,0)\} \subset \{0,1\}^3$$

is 2-interpolerend. Namelijk, op posities 1 en 2 komen alle 4 mogelijke paren $(x, y) \in \{0,1\}^2$ precies één keer voor in een element van C. Dat geldt ook voor posities 1 en 3, en ook voor 2 en 3.

Hier is een aanstekelijker voorbeeld, gebaseerd op het bekende meetkundige gegeven dat door ieder tweetal verschillende punten precies één lijn gaat. Stel dat $V = \mathbb{R}$, de reële getallen en laten we het reële twee-dimensionale vlak $\mathbb{R}^2$ beschouwen, met daarop een gegeven assenstelsel. Dan worden de coordinaten van een punt P in dat vlak gegeven door een paar reële getallen $(x, y) \in \mathbb{R}^2$. Elke lijn L is van de vorm

$$L = \{(x, y) \in \mathbb{R}^2 : ax + b = y\}$$

voor zekere constanten $a, b \in \mathbb{R}$. We noteren een zo'n lijn ook wel als

$$L : aX + b = Y.$$

Beschouw nu de lijnen L in het vlak met uitzondering van de strikt vertikale. Dat wil zeggen, we laten de lijnen $L : X = c$ buiten beschouwing.

Zij $x_1, \ldots, x_n \in \mathbb{R}$ vaste, verschillende getallen. Voor elke gegeven lijn $L : aX + b = Y$ met $a \neq 0$ kunnen we bij de gegeven x_i's corresponderende y_i's vinden zodanig dat $(x_i, y_i) \in L$ voor $i = 1, \ldots, n$.

Als we nu C definiëren als de verzameling van alle vectoren

$$y = (y_1, \ldots, y_n) \in \mathbb{R}^n$$

met de eigenschap dat

$$(x_1, y_1), \ldots, (x_n, y_n) \in L$$

voor een zekere lijn

$$L : aX + b = Y$$

met $a \neq 0$, dan heeft C de 2-interpolatie eigenschap.

Opgave 2. Waarom hebben we de strikt vertikale lijnen uitgesloten bij de constructie van C?

Opgave 3. Ga na dat C inderdaad 2-interpolerend is.

6.8. Definitie. Als de verzameling V eindig is, dan heet $C \subset V^n$ een *code ter lengte* n.

Opgave 4. Bewijs het volgende. Als $C \subset V^n$ een $(t+1)$-interpolerende code is, dan $|C| = |V|^{t+1}$.

Het voorbeeld gebaseerd op lijnen in het vlak leidt ook tot codes die 2-interpolerend zijn over $\mathbb{Z}/p\mathbb{Z}$, de (ring van de) restklassen modulo p met p een *priemgetal*. Dat gaan we als volgt na. Laten we het begrip "punt" opvatten als een element $P = (x, y) \in (\mathbb{Z}/p\mathbb{Z})^2$, en "lijn" als een verzameling

$$L = \{(x, y) \in (\mathbb{Z}/p\mathbb{Z})^2 : ax + b = y\}$$

voor zekere vaste $a, b \in \mathbb{Z}/p\mathbb{Z}$.

Zij $(x, y), (x', y') \in (\mathbb{Z}/p\mathbb{Z})^2$ twee verschillende punten in het "eindige twee-dimensionale vlak" over $\mathbb{Z}/p\mathbb{Z}$. Als $x = x' = c$ dan $y \neq y'$ en gaat er precies één lijn door die twee punten, namelijk de strikt vertikale lijn $L : X = c$.

Neem nu aan dat $x \neq x'$. De twee punten liggen op een lijn $L : aX+b = Y$ dan en slechts dan als

$$ax + b = y,$$
$$ax' + b = y'.$$

Door deze twee vergelijkingen van elkaar af te trekken, ontstaat de conditie

$$a(x - x') = y - y'.$$

Aangezien $x - x' \neq 0$ en p een priemgetal is, heeft $x - x'$ gegarandeerd een multiplicatieve inverse in $\mathbb{Z}/p\mathbb{Z}$, met andere woorden, de fractie $\frac{1}{x-x'}$ is goed gedefinieerd. Dus deze conditie is equivalent met

$$a = \frac{y - y'}{x - x'}.$$

Nu volgt verder dat er moet gelden $b = y - \frac{y-y'}{x-x'} \cdot x$, ofwel

$$b = \frac{xy' - x'y}{x - x'}.$$

Deze condities op a en b zijn ook voldoende (zo gaan we eenvoudig na door middel van substitutie), en daarom is er precies één lijn L die door de twee gegeven punten gaat.

Overigens, op grond van *lineaire algebra* hadden we dit onmiddellijk in kunnen zien, daar de *determinant* van de matrix M met de rijen $(x\ 1)$ en $(x'\ 1)$ gelijk is aan $x - x'$, een inverteerbaar element. Dus is er een unieke oplossing voor de vergelijking

$$M(a, b)^T = (y, y')^T.$$

Maar dat is weer een ander verhaal.

Bij de constructie van een 2-interpolerende code over $\mathbb{Z}/p\mathbb{Z}$ zonderen we vervolgens de strikt vertikale lijnen weer uit. In het reële geval konden we de lengte n willekeurig groot nemen. Hier hebben we echter $n \leq p$, omdat voor $n > p$ er twee punten onder de n punten moeten zijn die beide op dezelfde

strikt vertikale lijn liggen. Dat volgt uiteraard uit het *Laatjes Principe*: als je $n+1$ duiven verdeelt over n laatjes, dan is er minstens één laatje met minstens twee duiven.

Als we in het voorbeeld boven geen priemgetal p nemen, maar een willekeurig geheel getal $m > 1$, dan gaat het voorbeeld ook door, zo lang maar de grootste gemene deler van m en $(x - x')$ gelijk is aan 1, zodat $(x - x')$ een multiplicatieve inverse heeft in $\mathbb{Z}/m\mathbb{Z}$.

Opgave 5. Laat zien dat door de punten

$$(0,0), (2,1) \in (\mathbb{Z}/4\mathbb{Z})^2$$

geen enkele lijn $L \subset (\mathbb{Z}/4\mathbb{Z})^2$ gaat.

Opgave 6. Stel dat we geen priemgetal p nemen maar een willekeurig geheel getal $m > 1$ in de constructie van de 2-interpolerende code. Laat zien dat $n \leq p_0$, waar p_0 de kleinste priemdeler van m is.

Opgave 7. Voor lengte 3 is er een gemakkelijkere constructie die 2-interpolerend is en die niet alleen over $\mathbb{R}$ en over $\mathbb{Z}/p\mathbb{Z}$ werkt, maar ook over $\mathbb{Z}/n\mathbb{Z}$ met $n > 1$ een willekeurig geheel getal :

$$C = \{((x,y,z) : x + y = z, (x,y,z) \in (\mathbb{Z}/n\mathbb{Z})^3\}.$$

Laat zien dat deze code inderdaad 2-interpolerend is. Het allereerste voorbeeld dat we gaven is hier een speciaal geval van.

Langrange-interpolatie

te manier om codes te construeren die $(t+1)$-interpolerend zijn met $t > 1$, is door gebruik te maken van *polynomen*. Laten we eerst over de reële getallen $\mathbb{R}$ werken.

Een polynoom is een veelterm van de vorm

$$f(X) = a_0 + a_1 X + a_2 X^2 + \ldots + a_t X^t,$$

waarbij t een niet-negatief geheel getal is en de coefficienten $a_1, a_1, \ldots, a_t \in \mathbb{R}$. We identificeren X^0 met 1, en we mogen voor a_0 dus ook schrijven $a_0 X^0$. Het polynoom met $t = 0$ en $a_0 = 0$, dat wil zeggen $f(X) = 0$, is het *nulpolynoom*. We noteren de verzameling van alle polynomen met $\mathbb{R}[X]$, de *polynoom-ring (in één variable X) over* $\mathbb{R}$.

Zij $f(X) \in \mathbb{R}[X]$, met $f(X)$ ongelijk aan het nul-polynoom. De *graad* van $f(X)$, genoteerd $\deg(f(X))$, is gelijk aan de grootste i waarvoor $a_i \neq 0$. In het bijzonder, als $a_t \neq 0$, dan is de graad t. Een constant polynoom $f(X) = c$ met $0 \neq c \in \mathbb{R}$ heeft dus graad 0.

Zo is bijvoorbeeld

$$f(X) = a_0 + a_1 X$$

met $a_1 \neq 0$ een polynoom van graad 1. Als we f op natuurlijke wijze opvatten als een functie

$$f : \mathbb{R} \to \mathbb{R},$$
$$x \mapsto f(x)$$

dan is de grafiek van f natuurlijk een lijn in het vlak. De grafiek van de tweedegraads polynoom-functie

$$f(X) = a_0 + a_1 X + a_2 X^2$$

met $a_2 \neq 0$ is bijvoorbeeld een parabool.

Polynomen kunnen opgeteld en vermenigvuldigd worden op de voordehand liggende manier:

$$(a_0 + a_1 X + a_2 X^2 + \ldots + a_t X^t) + (b_0 + b_1 X + b_2 X^2 + \ldots + b_t X^t) =$$
$$(a_0 + b_0) + (a_1 + b_1)X + \ldots + (a_t + b_t)X^t,$$

en

$$(a_0 + a_1 X + a_2 X^2 + \ldots + a_t X^t) \cdot (b_0 + b_1 X + b_2 X^2 + \ldots + b_{t'} X^{t'}) =$$
$$\sum_{d=0}^{t+t'} \left(\sum_{i+j=d} a_i b_j \right) X^d.$$

Bijvoorbeeld,

$$(X + X^5) + (1 + 3X^2) = 1 + X + 3X^2 + X^5,$$
$$(-1 + X^3 - 10X^6) + (4 - 2X^3 + 10X^6) = 3 - X^3,$$
$$(3 + X^2 + 4X^5) - (3 + X^2 + 4X^5) = (3 + X^2 + 4X^5) + (-3 - X^2 - 4X^5) = 0$$
$$3 \cdot (4X^4 + X^5) = 12X^4 + 3X^5,$$
$$0 \cdot (X - 3X^{11}) = 0,$$

$$(5 + X^4 - X^7)(1 - 2X^2) = 5 - 10X^2 + X^4 - 2X^6 + X^7 + 2X^9.$$

Wat betreft de graad, als $f(X), g(X) \in \mathbb{R}[X]$ (beide niet het nul-polynoom) dan volgt onmiddellijk dat

$$\deg(f(X) \cdot g(X)) = \deg(f(X)) + \deg(g(X))$$

en

$$\deg(f(X) + g(X)) \leq \max\{\deg(f(X)), \deg(g(X))\}.$$

Als de graden verschillend zijn, dan is er gelijkheid in deze laatste expressie. Als de graden gelijk zijn, dan kunnen de "hoogste termen" elkaar eventueel uitdoven, en zakt de graad.

Als $f(z) = 0$ voor zekere $z \in \mathbb{R}$, dan heet z een *nulpunt* van f. Merk op dat niet alle polynomen in $\mathbb{R}[X]$ een nulpunt $z \in \mathbb{R}$ hebben. Bijvoorbeeld, voor $f(X) = X^2 + 1$ geldt dat $f(x) \geq 1$ voor alle $x \in \mathbb{R}$, en derhalve heeft deze $f(X)$ geen nulpunt in $\mathbb{R}$.

We bewijzen nu eerst de volgende fundamentele stelling over nulpunten van polynomen.

6.9. Stelling. *Zij $f(X) \in \mathbb{R}[X]$ een polynoom van graad > 0. Dan is het aantal nulpunten van $f(X)$ ten hoogste deg$(f(X))$.*

Bewijs. Het is voldoende de volgende claim te bewijzen: als $g(X) \in \mathbb{R}[X]$ en $g(z) = 0$ voor zekere $z \in \mathbb{R}$ dan kunnen we schrijven

$$g(X) = (X - z) \cdot h(X)$$

voor zekere $h(X) \in \mathbb{R}[X]$.

Voordat we de claim rechtvaardigen laten we zien dat de bewering in de propositie volgt uit de claim. We bewijzen dat met *inductie*. Allereerst, de propositie is geldig voor alle polynomen van graad 1: $a_0 + a_1X = 0$ met $a_1 \neq 0$ heeft uiteraard precies één nulpunt, namelijk

$$z = -\frac{a_0}{a_1}.$$

Stel nu als *inductie-hypothese* dat de propositie geldt voor alle polynomen van graad ten hoogste n voor zeker geheel getal $n \geq 1$. We laten zien dat de propositie dan ook geldt voor alle polynomen van graad $n+1$. Door deze redenering herhaald "mentaal" toe te passen vanaf $n = 1$, wordt duidelijk dat de propositie dan ook geldt voor *alle* graden n.

Neem nu een willekeurig polynoom $f(X)$ van graad $n+1$. Als het geen nulpunt heeft, dan zijn we klaar met dat polynoom en nemen we er één die wel een nulpunt heeft. Uit het feit dat $f(X) = (X - z) \cdot h(X)$ volgt nu dat het aantal nulpunten van $f(X)$ gelijk is aan 1 plus het aantal nulpunten van $h(X)$. Immers, $f(u) = 0$ dan en slechts dan als $u - z = 0$ of $h(u) = 0$. Maar $h(X)$ heeft graad n (gebruik hiervoor de eerdere opmerking over de graad van het product van polynomen), en heeft dus ten hoogste n nulpunten volgens de inductie-hypothese. We concluderen dat $f(X)$ zelf ten hoogste $n+1$ nulpunten heeft.

Rest nog het bewijs van de claim. Zij $g(X) \in \mathbb{R}[X]$ met

$$g(X) = b_0 + b_1 X + \ldots + b_r X^r.$$

en zij $z \in \mathbb{R}$ een nulpunt. Zet

$$Y = X - z$$

en schrijf

$$g(Y + z) = b_0 + b_1 \cdot (Y + z) + \ldots + b_r \cdot (Y + z)^r.$$

Door haakjes weg te werken zien we voorts onmiddellijk dat

$$g(Y + z) = d_0 + d_1 \cdot Y + \ldots + d_r \cdot Y^r$$

voor zekere $d_0, d_1, \ldots, d_r \in \mathbb{R}$. Aangezien

$$0 = g(0 + z) = d_0,$$

hebben we dus

$$g(Y + z) = d_1 \cdot Y + \ldots + d_r \cdot Y^r$$

en na terug substitueren van $Y = X - z$ en het buiten-haakjes-halen van een term $(X - z)$ volgt

$$g(X) = (X - z) \cdot (d_1 + d_2 \cdot (X - z) + \ldots + d_r \cdot (X - z)^{r-1}),$$

als gewenst. □

6.10. Stelling (Interpolatie Stelling van Lagrange over $\mathbb{R}$). *Gegeven $t+1$ verschillende punten $P_1, \ldots, P_{t+1} \in \mathbb{R}^2$, waarvan er geen twee op dezelfde vertikale lijn liggen. Dan is er een uniek polynoom $f(X) \in \mathbb{R}[X]$ van graad ten hoogste t zodanig dat $P_1, \ldots, P_{t+1}$ alle op de grafiek van f liggen.*

Figuur 9: Joseph Louis Lagrange (1736-1813) op een Franse postzegel.

Bewijs. We construeren het gevraagde polynoom en bewijzen vervolgens dat het uniek is. Laten we eerst de coordinaten van de gegeven punten P_i opschrijven:

$$P_1 = (x_1, y_1), \ldots, P_{t+1} = (x_{t+1}, y_{t+1}),$$

met $x_i \neq x_j$ als $i \neq j$. De punten $P_1, \ldots, P_{t+1}$ liggen alle op de grafiek van f precies wanneer $f(x_i) = y_i$ voor $i = 1, \ldots, t+1$.

We zullen het gevraagde polynoom $f(X)$ uit eenvoudige bouwstenen construeren. Voor $i = 1, \ldots, t+1$, zij $k_i(X) \in \mathbb{R}[X]$ het polynoom met de eigenschappen

- $k_i(x_i) = 1$.
- $k_i(x_j) = 0$ als $j \neq i$.
- De graad van $k_i(X)$ is t.

Deze polynomen $k_i(X)$ zijn eenvoudig te construeren, namelijk:

$$k_i(X) = \left(\prod_{r \neq i} \frac{1}{x_i - x_r} \right) \cdot \prod_{r \neq i} (X - x_r).$$

Merk op dat dit inderdaad een polynoom is in $\mathbb{R}[X]$: de linkerterm is een element van $\mathbb{R}$, en de rechterterm is een product van polynomen van graad 1. We gaan de geclaimde eigenschappen na. Ten eerste,

$$k_i(x_i) = \prod_{r \neq i} \frac{x_i - x_r}{x_i - x_r} = \prod_{r \neq i} 1 = 1,$$

want geen van de noemers is 0.

Ten tweede, als $j \neq i$, dan

$$k_i(x_j) = \left(\prod_{r \neq i} \frac{1}{x_i - x_r} \right) \cdot \prod_{r \neq i} (x_j - x_r) = 0,$$

want $(x_j - x_j) = 0$ komt voor in het product aan de rechterkant van de vermenigvuldiging, en de linkerkant is goed-gedefinieerd (en ongelijk aan 0).

Ten derde, de graad van de linkerterm is 0 want het is een constant polynoom ongelijk aan het nul-polynoom, en de graad van rechterterm is t, want dat is een product van t polynomen van graad 1. De graad van $k_i(X)$ is dus $0 + t = t$.

Definieer nu

$$f(X) = \sum_{j=1}^{t+1} y_j \cdot k_j(X) \in \mathbb{R}[X].$$

Dan is $f(X)$ het gevraagde polynoom. Allereerst is de graad van $f(X)$ ten hoogste gelijk aan het maximum van de graden van de termen in de sommatie, en die zijn ieder t. Dus, $\deg(f(X)) \leq t$. Voorts,

$$f(x_i) = \sum_{1 \leq j \leq t+1} y_j \cdot k_j(x_i) = y_i \cdot k_i(x_i) + \sum_{j \neq i} y_j \cdot k_j(x_i) = y_i + 0 = y_i.$$

Dus alle P_i liggen inderdaad op de grafiek van f.

Rest nog uniciteit. Stel dat er twee verschillende polynomen $f(X)$ en $g(X)$ bestaan met $f(X) \neq g(X)$ en die beide voldoen. Dat wil zeggen, $f(X) = a_0 + a_1 X + \ldots + a_t X^t$ en $g(X) = b_0 + b_1 X + \ldots + b_t X^t$ met $a_i \neq b_i$ voor zekere i, en de $t + 1$ punten P_i liggen zowel op de grafiek van f als op de grafiek van g. Beschouw dan het polynoom

$$h(X) = f(X) - g(X) \in \mathbb{R}[X].$$

Uit de aanname volgt dat $h(X)$ niet het nul-polynoom is en het is duidelijk dat $\deg(h(X)) \leq t$. Voorts,

$$h(x_i) = f(x_i) - g(x_i) = 0 - 0 = 0.$$

Dus, de $t+1$ verschillende x_i's zijn alle nulpunten van h. Vanwege de eerdere stelling over het aantal nulpunten van een polynoom volgt dat $h(X)$ niet een polynoom van graad groter dan 0 kan zijn. De enige mogelijkheid is dat $h(X) = c$, een constant polynoom. Er geldt dan $c = 0$, want $h(X)$ heeft nulpunten. We concluderen dat $h(X)$ het nul-polynoom is, en dat geeft een tegenspraak. □

$(t+1)$-Interpolerende codes

Het zal niet verbazen dat de Interpolatie Stelling van Lagrange stelling *ook* geldt indien we in plaats van polynomen over de reële getallen, polynomen over $\mathbb{F}_p := \mathbb{Z}/p\mathbb{Z}$ (p een priemgetal) beschouwen, want alle stappen in het bewijs zijn dan nog steeds geldig. Het is wel essentieel dat p priem is, want anders is niet gegarandeerd dat alle $x_j - x_i$ met $j \neq i$ inverteerbaar zijn.

We kunnen de stelling alleen niet meer in termen van grafieken formuleren, want die hebben niet de betekenis als in het reële geval. Bij het rekenen met de polynomen werken we uiteraard nu niet meer over $\mathbb{R}$, maar gebruiken in plaats daarvan de optelling, aftrekking, en vermenigvuldiging en deling in $\mathbb{F}_p$ die eerder behandeld is. De overige definities, zoals graad en nulpunt zijn geheel analoog, en we noteren de polynoom-ring over $\mathbb{F}_p$ met $\mathbb{F}_p[X]$.

6.11. Stelling (Interpolatie Stelling van Lagrange over $\mathbb{F}_p$). *Zij elementen $x_1, \ldots, x_{t+1} \in \mathbb{F}_p$ gegeven, met $x_i \neq x_j$ als $i \neq j$ (dus $p \geq t+1$). Zij voorts willekeurige elementen $y_1, \ldots, y_{t+1} \in \mathbb{F}_p$ gegeven.*

Dan is er een uniek polynoom $f(X) \in \mathbb{F}_p[X]$ zodanig dat

$$deg(f(X)) \leq t$$

$$f(x_1) = y_1, \ldots, f(x_{t+1}) = y_{t+1}.$$

Er zijn overigens interessante en zeer belangrijke generalizaties van deze stelling, maar die voeren te ver om er nu op in te gaan. Wél kunnen we nu eindelijk $(t+1)$-interpolerende codes van lengte n maken. Kies een priemgetal $p \geq n$ en neem $V = \mathbb{F}_p$. Zij t een geheel getal met $1 \leq t < n$ en zij $x_1, \ldots, x_n \in \mathbb{F}_p$ met $x_i \neq x_j$ als $i \neq j$.

Definieer

$$C = \{(f(x_1), \ldots, f(x_n)) \in \mathbb{F}_p^n : f(X) \in \mathbb{F}_p[X], \deg(f(X)) \leq t\}.$$

6.12. Stelling. *De code C van lengte n over $\mathbb{F}_p$ is $(t+1)$-interpolerend.*

Bewijs. Kies $A \subset \{1, \ldots, n\}$ met $|A| = t+1$ willekeurig en kies $w \in \mathbb{F}_p^{t+1}$ willekeurig. Vanwege Lagrange Interpolatie is er een uniek polynoom $g(X) \in \mathbb{F}_p[X]$ met $\deg(g(X)) \leq t$ zodanig dat

$$(g(x_i))_{i \in A} = w.$$

Dus is er een unieke $c \in C$ met $c_A = w$, namelijk

$$c = (g(x_1), \ldots, g(x_n)).$$

□

De oplossing van het t-uit-n expertise probleem

We zijn nu klaar om het expertise probleem op te lossen, waarbij Bob claimt dat hij expert is in een meerderheid van een aantal gegegeven onderwerpen maar niet wil prijsgegeven in welke precies hij een expert is.

Stel dat het aantal onderwerpen bijvoorbeeld 11 is, en Bob claimt expert te zijn in minstens 6 onderwerpen. Samen leggen Alice en Bob zich dan vast op een 6-interpolerende code van lengte 12 over, zeg, $\mathbb{F}_{101}$ (merk op dat 101 een priemgetal is). Per onderwerp maakt Alice een lijst met 101 moeilijke vragen.

Stel dat $Z \subset \{1, \ldots, 11\}$ met $|Z| = 5$ de onderwerpen representeert waarop Bob geen expert is. In het protocol legt Bob zich nu reeds op 5 vragen vast met behulp van de speelkaarten voordat hij de challenge ontvangt, één voor ieder van de onderwerpen in Z. De encyclopedie-inspectie is verder als vanouds. Gegeven de challenge e, berekent Bob het unieke codewoord $c = (c_0, c_1, \ldots, c_{11}) \in C$ zodanig dat $c_0 = e$ en $c_Z = w$, waar w de vector van vragen-nummers is waarop Bob zich reeds heeft vastgelegd. Hij deponeert de resterende kaarten in de schoenendoos, en beantwoordt uiteindelijk alle 11 vragen: vraag c_1 over het eerste onderwerp tot en met vraag c_{11} over het 11-de onderwerp. Alice doet dan de voor de hand liggende controles: of er een codewoord is consistent met de kaarten in de schoenendoos en de challenge, en of alle antwoorden correct zijn.

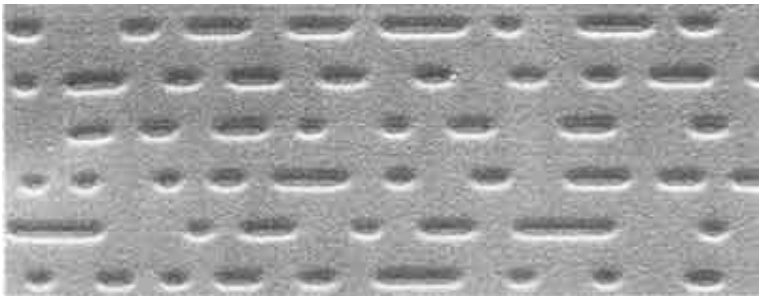
Opgave 8. Beargumenteer dat indien Bob liegt, hij over ten minste één onderwerp waarop hij geen expert is een volstrekt willekeurige vraag uit de lijst met 101 vragen over dat onderwerp voor zijn kiezen krijgt.

Opgave 9. Stel dat het aantal onderwerpen 15 is en de expertise claim betreft 8 onderwerpen. Hoe beïnvloedt dat de keuze van de code?

Foutencorrigerende codes

Een laatste toepassing van interpolerende codes is *foutcorrectie*. De reden dat een krasje op een cd of een dvd bij het afspelen geen storing oplevert is niet dat er een heel dikke beschermlaag zit op het oppervlak van de schijf, maar dat de informatie op een slimme, zo zuinig mogelijke manier redundant gerepresenteerd is dat het wegvallen of het veranderen van sommige informatie-symbolen getolereerd kan worden en de oorspronkelijke informatie desondanks gereconstrueerd kan worden. Behalve praktische toepassingen als cd-technologie en satelliet-communicatie hebben dergelijke codes talrijke theoretische toepassingen in de cryptografie en daarbuiten.

De basisgedachte van foutencorrigerende codes is elegant en eenvoudig, en komt in wezen op het volgende neer. Laten we ons de kaart van continentaal Europa in gedachten nemen, met alleen de hoofdsteden. Stel er ontsnapt een cavia uit een hoofdstad, en alleen zijn huidige locatie wordt op de kaart weergegeven. Uit welke stad was die afkomstig? Het is vast een realistische aanname dat cavia's hooguit enkele kilometers van huis af kunnen dwalen, en zeker *minder dan de helft van de kortste afstand tussen enig tweetal hoofsteden*. Dus, onder deze aannames is de stad die het dichtsbijzijnd is uniek,



Figuur 10: Close-up van een cd.

en kwam de cavia daar ook vandaan. Bijvoorbeeld, als de locatie van de ontsnapte cavia niet al te ver buiten Madrid ligt, dan moet hij oorspronkelijk ook uit Madrid gekomen zijn.

6.13. Definitie. Zij n een positief geheel getal en zij V een eindige, niet-lege verzameling. De Hamming-afstand tussen $v = (v_1, \ldots, v_n) \in V^n$ en $w = (w_1, \ldots, w_n) \in V^n$, genoteerd $d(v, w)$, is gelijk aan het aantal coordinaten i met $v_i \neq w_i$.

Er geldt dus $0 \leq d(v, w) \leq n$.

6.14. Definitie. De minimum-afstand $d(C)$ van een code C is het minimum van $d(c, c')$ genomen over alle paren $c, c' \in C$ met $c \neq c'$.

Als C een triviale code is, dat wil zeggen, $C = \{c\}$, dan zeggen we dat $d(C) = 0$. De minimum-afstand van een code is dus "de korste afstand binnen de code."

6.15. Definitie. Zij r een niet-negatief geheel getal en zij $v \in V^n$. De bol $B(v; r)$ met straal r en middelpunt v is de verzameling van elementen $v' \in V^n$ met de eigenschap dat $d(v, v') \leq r$.

Een bol om v bestaat dus uit alle elementen die daar dicht genoeg bij liggen.

Opgave 10. Zij $v \in V^n$. Laat zien dat het aantal elementen in een bol $B(v; r)$ met straal r gelijk is aan

$$\sum_{i=0}^{r} \binom{n}{i} \cdot (|V| - 1)^i.$$

Stel nu dat we een code C over V hebben, met lengte n en minimum-afstand d. Zij $c \in C$. Als c in minder dan $\frac{d}{2}$ coordinaten gewijzigd wordt, dan bevindt het resulterende woord $\tilde{c}$ zich op afstand minder dan $\frac{d}{2}$ van c. Dat betekent dat c het *unieke* dichtstbijzijnde codewoord is van $\tilde{c}$. Als namelijk $c' \in C$ ook op afstand minder dan $\frac{d}{2}$ van $\tilde{c}$ zou liggen, dan zou de afstand tussen c en c' minder dan

$$\frac{d}{2} + \frac{d}{2} = d$$

zijn: met $< \frac{d}{2}$ wijzigingen in c reizen we van c naar $\tilde{c}$, en met $< \frac{d}{2}$ wijzigingen in $\tilde{c}$ reizen we van $\tilde{c}$ naar c'. Dus, in totaal zouden we minder dan d wijzigingen in c hoeven aanbrengen om van c naar c' te reizen, en dat is strijdig

met de minimum-afstand. Hieraan ontlenen codes hun nut met betrekking tot fouten-correctie. De informatie die bijvoorbeeld op een cd opgeslagen moet worden, wordt eerst in kleine sectoren verdeeld. Ieder van die sectoren kan dan een kleine hoeveelheid, zeg ten hoogste W, verschillende waarden aannemen. Als we nu een code nemen met

$$|C| \geq W,$$

dan kunnen met ieder van die waarden een codewoord associëren. Elk van die codewoorden wordt dan op de cd opgeslagen. Die codewoorden zijn dan wel "langer" dan de oorspronkelijke informatie, maar als krassen en beschadigingen nu binnen de perken blijven, dan kan de cd speler wel beschadigde woorden decoderen tot de oorspronkelijke.

6.16. Stelling. *Zij $C \subset V^n$ een $t+1$-interpolerende code. Dan $d(C) = n-t$ en $|C| = |V|^{t+1}$.*

Bewijs. Zij $c, c' \in C$ met $c \neq c'$. Stel dat

$$c_A = c'_A$$

voor zekere $A \subset \{1, \ldots, n\}$. Dan moet gelden dat $|A| \leq t$; anders geldt $c = c'$ wegens de interpolatie-eigenschap. Derhalve verschillen c en c' in ten minste $n - t$ coordinaten. De bewering over $|C|$ hebben we al eerder bewezen. □

We hebben eerder gezien dat zulke codes gemaakt kunnen worden met behulp van polynomen en Lagrange Interpolatie. Voor polynoom codes zijn er elegante en zeer efficiënte methoden om te decoderen, dat wil zeggen de fouten te corrigeren, maar het voert te ver om die hier te behandelen.

Bij "goede" codes zijn zowel $|C|$ als ook $d(C)$ "groot". Dat is gelijk de moeilijkheid bij het vinden ervan, want als we C steeds groter maken, dan komen de code-woorden uiteindelijk "dichter op elkaar" te zitten. Een karakteristieke vraag over codes is dan ook de volgende. Voor gegeven V, n en d, wat is de grootst mogelijke waarde van $|C|$ waarbij $C \subset V^n$ een code met minimum-afstand ten minste d is?

Hier is een eenvoudig telprincipe op grond waarvan we een ondergrens op die maximale waarde kunnen afleiden. Stel dat we een code C met $d(C) \geq d$ hebben. Kunnen we een conditie vinden die impliceert dat C nog niet maximaal is? Zo ja, dan kunnen we aan een code die aan die conditie voldoet

natuurlijk nog minstens 'e'en woord toevoegen zonder dat de minimum afstand verkleint.

Sla om elke $c \in C$ een bol met straal $d-1$. En beschouw de vereniging

$$\bigcup_{c \in C} B(c; d-1) \subset V^n$$

van al die bollen. Als het nu zo is dat die vereniging niet gelijk is aan V^n, dan is er dus ten minste één $v \in V^n$ die niet in die vereniging ligt. Die v ligt niet in enige bol, en dus geldt dat

$$d(c, v) \geq d$$

voor alle $c \in C$. Dat betekent dat we v kunnen toevoegen:

$$C' := C \cup \{v\}.$$

Deze nieuwe code C' bevat één woord meer maar nog steeds geldt $d(C') \geq d$. Dit proces kan in principe natuurlijk herhaald worden, net zo lang totdat een maximale code is verkregen.

Andersom gezegd, als $\overline{C}$ een maximale code met minimum afstand ten minste d is, dan moeten we uiteraard hebben

$$\bigcup_{c \in \overline{C}} B(c; d-1) = V^n,$$

anders zouden we nog een woord kunnen toevoegen en dat is strijdig met de maximaliteit van $\overline{C}$. In het bijzonder moet dus gelden

$$\sum_{c \in C} |B(c; d-1)| = |C| \cdot \left(\sum_{i=1}^{d-1} \binom{n}{i} (|V|-1)^i \right) \geq |V|^n.$$

Voor een maximale code $\overline{C}$ met minimum afstand ten minste d hebben we derhalve de beroemde ondergrens

$$|\overline{C}| \geq \frac{|V|^n}{\left(\sum_{i=0}^{d-1} \binom{n}{i} (|V|-1)^i \right)}.$$

Een eenvoudige *bovengrens* kunnen we als volgt afleiden. Zij C een code over V van lengte n met minimum-afstand d. Beschouw de code C'' die uit

C verkregen wordt door, zeg, de eerste $d-1$ coördinaten buiten beschouwing te laten. Dan is C'' een code over V van lengte $n-d+1$. C'' heeft net zoveel elementen als C: als er $c, c' \in C$ met $c \neq c'$ zouden zijn die na weglating van die coöordinaten hetzelfde element van C'' zouden opleveren, dan zou de afstand tussen c en c' in C ten hoogste $d-1$ geweest zijn, een tegenspraak. Er volgt:

$$|C| \leq |V|^{n-d+1},$$

want de verzameling $|V|^{n-d+1}$ moet groot genoeg zijn om die $|C''|$ $(=|C|)$ verschillende elementen te kunnen bevatten.

Secret sharing

We behandelen nu een toepassing van codes op *secret sharing*, een belangrijk onderwerp in de cryptografie, met name in de *secure computation*, en tegenwoordig ook in *zero knowledge bewijzen*.

Secret sharing behelst in het bijzonder methoden om geheime informatie zó in n stukken "op te knippen" dat een willekeurige collectie van t deelstukken, de *shares*, niets prijsgeeft over de geheime informatie, terwijl een willekeurige collectie van meer dan t zulke shares unieke reconstructie van de geheime informatie mogelijk maakt.

De originele motivatie in jaren 70 van de 20e eeuw voor secret sharing was het tegengaan van *single-point-of-failure*. Geheime informatie moet namelijk ergens opgeslagen worden, en indien een hacker zich daartoe toegang weet te verschaffen, dan wordt die geheime informatie daarmee gecompromitteerd. Met secret sharing kan de geheime informatie over *meerdere* locaties uitgesmeerd worden, zodanig dat de informatie zelfs nog geheim blijft voor een hacker die in t locaties inbreekt. Om de informatie te reconstrueren zijn de shares uit meer dan t locaties echter weer voldoende.

Zij C een (publiek bekende) $(t+1)$-interpolerende code van lengte $n+1$ over een verzameling symbolen V. Laten we zeggen dat de geheime informatie wordt gerepresenteerd met een symbool $s \in V$. Ga dan als volgt te werk:

- Kies een volstrekt willekeurig codewoord $c \in C$ zodanig dat $c_0 = s$. Met volstrekt willekeurig wordt hier bedoeld: kies c volgens de uniforme kansverdeling op C, geconditioneerd op $c_0 = s$.

- Definieer de shares s_i als $s_i := c_i$, $i = 1, \ldots, n$.

6.17. Propositie. *Zij $A \subset \{1, \ldots, n\}$. Als $|A| \leq t$, dan is $(s_i)_{i \in A}$ volgens de uniforme kansverdeling op $V^{|A|}$ verdeeld (en geeft dus "geen informatie" over het secret s). Als $|A| > t$ dan kan het secret s uniek worden gereconstrueerd uit $(s_i)_{i \in A}$. We zeggen dat de methode* t-privacy *biedt en* $(t+1)$-reconstructie.

Bewijs. Voor t-privacy is het voldoende om het geval $|A| = t$ te beschouwen. Dan is er voor elke mogelijke keus $\tilde{s} \in V$ en voor elke $w = (v_i)_{i \in A}$ precies één codewoord $c \in C$ met

$$c_0 = \tilde{s},$$

$$c_A = w.$$

Onafhankelijk van de geheime informatie, heerst er dus op $(s_i)_{i \in A}$ de uniforme kansverdeling op V^t. Daarom is het onmogelijk om de shares van twee verschillende secrets te onderscheiden, althans vanuit het "gezichtspunt van A."

Als daarentegen $|A| > t$, dan is er een uniek codewoord c met $c_A = (c_i)_{i \in A}$. Daaruit kan c_0 worden afgeleid.

□

Opgave 11. Zij C een $(t+1)$-interpolerende code over V van lengte $n+1$, en beschouw de bovenstaande secret sharing methode. Zij $\sigma = (s_1, \ldots, s_n) \in V^n$ een vector van n shares voor het secret $s \in V$. Zij $x \in V^n$ zodanig dat $d(\sigma, x) \leq t$. Laat zien dat als $t < \frac{n}{3}$, het secret s uniek uit x gereconstrueerd kan worden.

Opgave 12. Ga uit van een interpolerende code gebaseerd op de polynoom constructie en verwoord de bovenstaande secret sharing methode in termen van polynomen (Shamir's secret sharing scheme).

Opgave 13. Zij $\ell \geq 1$ een geheel getal en zij C een $(t+\ell)$-interpolerende code over V van lengte $n + \ell$. Ontwerp een variant waarin de geheime informatie s een willekeurige waarde in V^ℓ kan aannemen in plaats van in V, en waarin elke collectie van t van de n shares geen informatie geeft over s, terwijl elke collectie van $t + \ell$ van de n shares tot unieke reconstructie van s leidt, dat wil zeggen: secret in V^ℓ, t-privacy en $(t + \ell)$-reconstructie.