
Operational notes








































Document updated on **August 1, 2022**.

The following colors are **not** part of the final product, but serve as highlights in the editing/review process:

- text that needs attention from the Subject Matter Experts: Mirco, Anna,& Jan
- terms that have not yet been defined in the book
- things that need to be checked only at the very final typesetting stage (and it doesn't make sense to do them before)
- text that needs advice from the communications/marketing team: Aaron & Shane
- text that needs to be completed or otherwise edited (by Sylvia)


NB: This PDF only includes the following chapter(s): Arithmetics.

Todo list

14	 Clarinet	5
15	 zero-knowledge proofs	5
16	 played with	6
17	 Update reference when content is finalized	6
18	 methatical	6
19	 numerical	6
20	 a list of additional exercises	6
21	 think about them	6
22	 Pluralize chapter title	13
23	 check if this is already introduced in intro	13
24	 unify addressing the reader	13
25	 unify addressing the reader	13
26	 Move content on binary representation	14
27	 simplify Sage ex.	14
28	 To see that	15
29	 let's	15
30	 "themselves" is more common?	15
31	 you	15
32	 readers	18
33	 Commonwealth countries	18
34	 Add explanation	18
35	 Add more explanation	19
36	 check algorithm floating	24
37	 subtrahend	34
38	 minuend	34
39	 algorithm-floating	36
40	 check algorithm floating	39
41	 Sylvia: I would like to have a separate counter for definitions	42
42	 check reference	48
43	 runtime complexity	48
44	 add reference	48
45	 S: what does “efficiently” mean here?	48
46	 computational hardness assumptions	49
47	 check reference	49
48	 check reference	49
49	 explain last sentence more	50
50	 “equation”?	51
51	 check reference	51
52	 what’s the difference between \mathbb{F}_p^* and \mathbb{Z}_p^* ?	51

53	Legendre symbol	51
54	Euler's formular	51
55	These are only explained later in the text, '4.31'	51
56	are these going to be relevant later? yes, they are used in various snark proof systems	52
57	TODO: theorem: every factor of order defines a subgroup...	52
58	Is there a term for this property?	52
59	a few examples?	54
60	check reference	54
61	TODO: DOUBLE CHECK THIS REASONING.	55
62	Mirco: We can do better than this	56
63	check reference	58
64	add reference	58
65	pseudorandom	58
66	oracle	58
67	check reference	58
68	check reference	61
69	check reference	61
70	check reference	61
71	check reference	61
72	add more examples protocols of SNARK	61
73	check reference	61
74	add reference	62
75	Abelian groups	62
76	codomain	62
77	Check change of wording	62
78	add reference	63
79	Expand on this?	64
80	check reference	64
81	S: are we introducing elliptic curves in section 1 or 2?	65
82	check reference	66
83	check reference	66
84	add reference	66
85	check reference	66
86	write paragraph on exponentiation	67
87	add reference	67
88	check reference	67
89	add reference	67
90	group pairings	67
91	add reference	68
92	check reference	68
93	check reference	71
94	add reference	72
95	TODO: Elliptic Curve asymmetric cryptography examples. Private key, generator,	
96	public key.	74
97	add reference	74
98	maybe remove this sentence?	74
99	affine space	74
100	cusps	75

101	self-intersections	75
102	check reference	76
103	check reference	77
104	jubjub	77
105	check reference	77
106	affine plane	77
107	check reference	78
108	check reference	78
109	check reference	79
110	sign	79
111	more explanation of what the sign is	79
112	check reference	79
113	S: I don't follow this at all	79
114	check reference	80
115	add explanation of how this shows what we claim	80
116	should this def. be moved even earlier?	80
117	chord line	81
118	tangential	81
119	tangent line	81
120	remove Q ?	81
121	where?	82
122	check reference	82
123	check reference	82
124	check reference	82
125	check reference	83
126	check reference	83
127	check reference	84
128	check reference	84
129	check reference	84
130	add term	85
131	add term	85
132	add reference	85
133	cofactor clearing	85
134	add reference	85
135	check reference	85
136	check reference	86
137	add reference	86
138	add reference	86
139	check reference	86
140	check reference	86
141	check reference	86
142	check reference	87
143	check reference	87
144	Explain how	87
145	write example	88
146	check reference	88
147	add reference	88
148	check reference	88

149	 add reference	88
150	 check reference	88
151	 add reference	88
152	 check reference	89
153	 add reference	89
154	 check reference	89
155	 add reference	89
156	 add reference	89
157	 add reference	89
158	 check reference	89
159	 check reference	89
160	 Check if following Alg is floated too far	89
161	 add reference	91
162	 add reference	91
163	 write up this part	91
164	 is the label in L ^A T _E X correct here?	91
165	 check reference	92
166	 check reference	92
167	 check reference	92
168	 check reference	92
169	 check reference	93
170	 check reference	94
171	 check reference	94
172	 check reference	94
173	 check reference	94
174	 check reference	94
175	 add reference	94
176	 check reference	95
177	 check reference	96
178	 check reference	96
179	 check reference	96
180	 check reference	96
181	 check reference	97
182	 check reference	97
183	 check reference	98
184	 either expand on this or delete it	98
185	 add reference	98
186	 check reference	98
187	 check reference	98
188	 check reference	98
189	 check reference	99
190	 check reference	99
191	 check reference	99
192	 check reference	100
193	 check reference	100
194	 check reference	100
195	 add reference	100
196	 add reference	100

197	■ This needs to be written (in Algebra)	101
198	■ add reference	101
199	■ add reference	101
200	■ check reference	101
201	■ towers of curve extensions	101
202	■ check reference	102
203	■ check reference	102
204	■ check reference	102
205	■ check reference	102
206	■ add reference	103
207	■ check reference	103
208	■ S: either add more explanation or move to a footnote	103
209	■ type 3 pairing-based cryptography	103
210	■ add references?	103
211	■ check reference	104
212	■ check reference	104
213	■ check floating of algorithm	105
214	■ add references	106
215	■ check reference	106
216	■ add reference	106
217	■ check reference	106
218	■ check reference	106
219	■ add reference	107
220	■ should all lines of all algorithms be numbered?	107
221	■ check reference	108
222	■ check reference	108
223	■ check reference	108
224	■ check if the algorithm is floated properly	108
225	■ check reference	108
226	■ again?	110
227	■ check reference	111
228	■ circuit	111
229	■ signature schemes	111
230	■ add reference	111
231	■ check reference	111
232	■ check reference	111
233	■ add references	111
234	■ add reference	111
235	■ reference text to be written in Algebra	111
236	■ check reference	112
237	■ check reference	112
238	■ check reference	112
239	■ add reference	113
240	■ algebraic closures	113
241	■ check reference	113
242	■ check reference	113
243	■ check reference	113
244	■ check reference	114

245	check reference	114
246	disambiguate	114
247	add reference	115
248	unify terminology	115
249	check reference	116
250	actually make this a table?	116
251	exercise still to be written?	117
252	add reference	117
253	check reference	117
254	check reference	117
255	add reference	118
256	check reference	118
257	check reference	118
258	check reference	119
259	add reference	120
260	check reference	120
261	check reference	120
262	check reference	121
263	what does this mean? Maybe just delete it	121
264	write up this part	122
265	add reference	122
266	check reference	122
267	cyclotomic polynomial	123
268	Pholaard-rho attack	123
269	todo	123
270	why? Because in this book elliptic curves are only defined for fields of chracteristic > 3	123
271	check reference	123
272	check reference	123
273	what does this mean?	123
274	add reference	123
275	add reference	124
276	check reference	124
277	check reference	124
278	add reference	125
279	add exercise	125
280	check reference	126
281	add reference	126
282	add reference	126
283	add reference	126
284	check reference	127
285	check reference	127
286	add reference	127
287	add reference	127
288	add reference	128
289	check reference	128
290	add reference	128
291	add reference	128
292	finish writing this up	129

293	add reference	129
294	correct computations	129
295	fill in missing parts	129
296	add reference	130
297	check equation	130
298	Chapter 1?	131
299	"rigorous"?	131
300	"proving"?	131
301	Add example	132
302	M: 1:1 correspondence might actually be wrong	132
303	binary tuples	132
304	add reference	133
305	add reference	133
306	check reference	133
307	check reference	133
308	Are we using w and x interchangeably or is there a difference between them?	134
309	check reference	134
310	jubjub	134
311	check reference	134
312	check reference	134
313	check wording	134
314	check reference	134
315	check references	135
316	add reference	135
317	add reference	135
318	check reference	136
319	add reference	136
320	check reference	137
321	check reference	137
322	add reference	138
323	add reference	139
324	Schur/Hadamard product	139
325	add reference	139
326	check reference	139
327	check reference	140
328	add reference	141
329	check reference	142
330	check reference	142
331	check reference	142
332	check reference	142
333	check reference	143
334	add reference	143
335	add reference	144
336	check reference	144
337	check reference	145
338	check reference	145
339	check reference	145
340	add reference	146

341	check reference	148
342	add reference	148
343	check reference	149
344	check reference	149
345	check reference	149
346	Should we refer to R1CS satisfiability (p. 142 here?	150
347	check reference	151
348	add reference	151
349	check reference	151
350	check reference	152
351	check reference	152
352	check reference	153
353	check reference	155
354	add reference	156
355	"by"?	156
356	check reference	156
357	check reference	156
358	add reference	156
359	add reference	156
360	check reference	156
361	add reference	156
362	clarify language	158
363	check reference	159
364	add reference	159
365	check reference	159
366	add reference	159
367	check references	161
368	add references to these languages?	161
369	check reference	164
370	check reference	165
371	check reference	165
372	check reference	166
373	check reference	167
374	check reference	167
375	check reference	169
376	check reference	169
377	check reference	170
378	add reference	170
379	check reference	170
380	add reference	170
381	add reference	170
382	check reference	171
383	check reference	171
384	check reference	171
385	check reference	171
386	add reference	171
387	check reference	172
388	check reference	173

389	■ "constraints" or "constrained"?	173
390	■ check reference	174
391	■ "constraints" or "constrained"?	174
392	■ add reference	174
393	■ "constraints" or "constrained"?	174
394	■ add reference	175
395	■ check references	175
396	■ check reference	175
397	■ add reference	176
398	■ can we rotate this by 90°?	176
399	■ check reference	177
400	■ add reference	177
401	■ add reference	177
402	■ shift	179
403	■ bishift	180
404	■ add reference	181
405	■ check reference	182
406	■ Add example	183
407	■ add reference	184
408	■ add reference	185
409	■ check reference	186
410	■ add reference	186
411	■ add reference	186
412	■ check reference	187
413	■ add reference	187
414	■ add reference	187
415	■ add reference	189
416	■ check reference	190
417	■ check reference	191
418	■ common reference string	191
419	■ simulation trapdoor	191
420	■ check reference	191
421	■ check reference	191
422	■ add reference	192
423	■ check reference	192
424	■ check reference	192
425	■ check reference	192
426	■ "invariable"?	192
427	■ explain why	193
428	■ 4 examples have the same title. Change it to be distinct	193
429	■ check reference	193
430	■ add reference	193
431	■ check reference	193
432	■ add reference	193
433	■ add reference	194
434	■ add reference	195
435	■ check reference	196
436	■ add reference	196

437	add reference	197
438	check reference	197
439	check reference	197
440	add reference	197
441	add reference	197
442	check reference	198
443	add reference	198
444	add reference	198
445	add reference	198
446	check reference	198
447	add reference	198
448	add reference	198
449	add reference	198
450	add reference	198
451	add reference	199
452	add reference	199
453	add reference	199
454	add reference	199
455	check reference	201
456	check reference	201
457	add reference	201
458	add reference	201
459	add reference	201
460	add reference	201
461	add reference	202
462	add reference	202
463	add reference	202
464	add reference	202
465	fix error	202
466	add reference	202
467	check reference	203
468	add reference	203
469	add reference	203
470	add reference	203
471	add reference	204
472	add reference	204
473	add reference	204
474	add reference	204
475	add reference	204
476	add reference	204
477	add reference	204
478	add reference	205

479

MoonMath manual

480

TechnoBob and the Least Scruples crew

481

August 1, 2022

Contents

1	Introduction	5
1.1	Aims and target audience	5
1.2	The Zoo of Zero-Knowledge Proofs	7
	To Do List	9
	Points to cover while writing	9
2	Preliminaries	10
2.1	Preface and Acknowledgements	10
2.2	Purpose of the book	10
2.3	How to read this book	11
2.4	Cryptological Systems	11
2.5	SNARKS	11
2.6	complexity theory	11
2.6.1	Runtime complexity	11
2.7	Software Used in This Book	12
2.7.1	Sagemath	12
3	Arithmetics	13
3.1	Introduction	13
3.1.1	Aims and target audience	13
3.2	Integer arithmetic	13
3.2.1	Integers, natural numbers and rational numbers	13
3.2.2	Euclidean Division	16
3.2.3	The Extended Euclidean Algorithm	19
	Coprime Integers	20
3.3	Modular arithmetic	20
	Congruence	21
	Computational Rules	21
	The Chinese Remainder Theorem	24
	Remainder Class Representation	25
	Modular Inverses	27
3.4	Polynomial arithmetic	30
	Polynomial arithmetic	34
	Euklidean Division	35
	Prime Factors	37
	Lagrange interpolation	38

517	4 Algebra	42
518	4.1 Commutative Groups	42
519	Finite groups	44
520	Generators	44
521	The exponential map	45
522	Factor Groups	46
523	Pairings	47
524	4.1.1 Cryptographic Groups	48
525	The discrete logarithm assumption	49
526	The decisional Diffie–Hellman assumption	50
527	The computational Diffie–Hellman assumption	51
528	Cofactor Clearing	52
529	4.1.2 Hashing to Groups	52
530	Hash functions	52
531	Hashing to cyclic groups	53
532	Hashing to modular arithmetics	54
533	Pedersen Hashes	58
534	MimC Hashes	59
535	Pseudorandom Functions in DDH-A groups	59
536	4.2 Commutative Rings	59
537	Hashing to Commutative Rings	62
538	4.3 Fields	62
539	4.3.1 Prime fields	64
540	Square Roots	65
541	Exponentiation	67
542	Hashing into prime fields	67
543	MiMC Hash functions	67
544	4.3.2 Extension Fields	67
545	Hashing into extension fields	71
546	4.4 Projective Planes	71
547	5 Elliptic Curves	74
548	5.1 Elliptic Curve Arithmetics	74
549	5.1.1 Short Weierstraß Curves	74
550	Affine short Weierstraß form	75
551	Affine compressed representation	79
552	Affine group law	80
553	Scalar multiplication	84
554	Projective short Weierstraß form	88
555	Projective Group law	89
556	Coordinate Transformations	91
557	5.1.2 Montgomery Curves	91
558	Affine Montgomery Form	91
559	Affine Montgomery coordinate transformation	93
560	Montgomery group law	94
561	5.1.3 Twisted Edwards Curves	95
562	Twisted Edwards Form	95
563	Twisted Edwards group law	97

564	5.2	Elliptic Curve Pairings	98
565		Embedding Degrees	98
566		Elliptic Curves over extension fields	100
567		Full torsion groups	101
568		Torsion subgroups	103
569		The Weil pairing	105
570	5.3	Hashing to Curves	107
571		Try-and-increment hash functions	108
572	5.4	Constructing elliptic curves	111
573		The Trace of Frobenius	111
574		The j -invariant	113
575		The Complex Multiplication Method	114
576		The <i>BLS6_6</i> pen-and-paper curve	122
577		Hashing to pairing groups	129
578	6	Statements	131
579	6.1	Formal Languages	131
580		Decision Functions	132
581		Instance and Witness	135
582		Modularity	138
583	6.2	Statement Representations	138
584	6.2.1	Rank-1 Quadratic Constraint Systems	139
585		R1CS representation	139
586		R1CS Satisfiability	141
587		Modularity	143
588	6.2.2	Algebraic Circuits	143
589		Algebraic circuit representation	143
590		Circuit Execution	148
591		Circuit Satisfiability	150
592		Associated Constraint Systems	151
593	6.2.3	Quadratic Arithmetic Programs	156
594		QAP representation	156
595		QAP Satisfiability	158
596	7	Circuit Compilers	161
597	7.1	A Pen-and-Paper Language	161
598	7.1.1	The Grammar	161
599	7.1.2	The Execution Phases	163
600		The Setup Phase	163
601		The Prover Phase	165
602	7.2	Common Programing concepts	165
603	7.2.1	Primitive Types	165
604		The base-field type	166
605		The Subtraction Constraint System	169
606		The Inversion Constraint System	170
607		The Division Constraint System	171
608		The boolean Type	172
609		The boolean Constraint System	172

610		The AND operator constraint system	173
611		The OR operator constraint system	173
612		The NOT operator constraint system	174
613		Modularity	175
614		Arrays	178
615		The Unsigned Integer Type	178
616		The uN Constraint System	179
617		The Unsigned Integer Operators	180
618	7.2.2	Control Flow	181
619		The Conditional Assignment	181
620		Loops	183
621	7.2.3	Binary Field Representations	184
622	7.2.4	Cryptographic Primitives	186
623		Twisted Edwards curves	186
624		Twisted Edwards curve constraints	186
625		Twisted Edwards curve addition	187
626	8	Zero Knowledge Protocols	189
627	8.1	Proof Systems	189
628	8.2	The “Groth16” Protocol	190
629		The Setup Phase	192
630		The Prover Phase	197
631		The Verification Phase	200
632		Proof Simulation	202
633	9	Exercises and Solutions	206

Chapter 3

Arithmetics

S: This chapter talks about different types of arithmetic, so I suggest using "Arithmetics" as the chapter title.

Pluralize chapter title

3.1 Introduction

3.1.1 Aims and target audience

The goal of this chapter is to bring a reader with only basic school-level algebra up to speed in arithmetics. We start with a brief recapitulation of basic integer arithmetics, discussing long division, the greatest common divisor and Euclidean division. After that, we introduce modular arithmetics as **the most important** skill to compute our **pen-and-paper examples**. We then introduce polynomials, compute their analogs to integer arithmetics and introduce the important concept of Lagrange interpolation.

check if this is already introduced in intro

3.2 Integer arithmetic

In a sense, integer arithmetic is at the heart of large parts of modern cryptography. Fortunately, most readers will probably remember integer arithmetic from school. It is, however, important that you can confidently apply those concepts to understand and execute computations in the many pen-and-paper examples that form an integral part of the MoonMath Manual. We will therefore recapitulate basic arithmetic concepts to refresh **your** memory and fill any knowledge gaps.

unify addressing the reader

Even though the terms and concepts in this chapter might not appear in the literature on zero-knowledge proofs directly, understanding them is necessary to follow subsequent chapters and beyond: terms like **groups** or **fields** also crop up very frequently in academic papers on zero-knowledge cryptography.

unify addressing the reader

3.2.1 Integers, natural numbers and rational numbers

Integers are also known as **whole numbers**, that is, numbers that can be written without fractional parts. Examples of numbers that are **not** integers are $\frac{2}{3}$, 1.2 and -1280.006 .

Throughout this book, we use the symbol \mathbb{Z} as a shorthand for the set of all **integers**:

$$\mathbb{Z} := \{\dots, -3, -2, -1, 0, 1, 2, 3, \dots\} \quad (3.1)$$

If $a \in \mathbb{Z}$ is an integer, then we write $|a|$ for the **absolute value** of a , that is, the non-negative value of a without regard to its sign:

$$|4| = 4 \quad (3.2)$$

$$|-4| = 4 \quad (3.3)$$

We use the symbol \mathbb{N} for the set of all positive integers, usually called the set of **natural numbers**. Furthermore, we use \mathbb{N}_0 for the set of all non-negative integers. This means that \mathbb{N} does not contain the number 0, while \mathbb{N}_0 does:

$$\mathbb{N} := \{1, 2, 3, \dots\} \quad \mathbb{N}_0 := \{0, 1, 2, 3, \dots\}$$

SB: Talking about the binary representation seems way to complex at this stage, and the concepts introduced here are not used for several chapters. Let $n \in \mathbb{N}_0$ be a non-negative integer and (b_0, b_1, \dots, b_k) a string of **bits** $b_j \in \{0, 1\} \subset \mathbb{N}_0$ for some non negative integer $k \in \mathbb{N}$, such that the following equation holds:

$$n = \sum_{j=0}^k b_j \cdot 2^j \quad (3.4)$$

In this case, we call $\text{Bits}(n) := \langle b_0, b_1, \dots, b_k \rangle$ the **binary representation** of n , say that n is a k -bit number and call $k := |n|_2$ the **bit length** of n . It can be shown, that the binary representation of any non negative integer is unique. We call b_0 the **least significant bit** and b_k the **most significant bit** and define the **Hamming weight** of an integer as the number of 1s in its binary representation.

In addition, we use the symbol \mathbb{Q} for the set of all **rational numbers**, which can be represented as the set of all fractions $\frac{n}{m}$, where $n \in \mathbb{Z}$ is an integer and $m \in \mathbb{N}$ is a natural number, such that there is no other fraction $\frac{n'}{m'}$ and natural number $k \in \mathbb{N}$ with $k \neq 1$ and

$$\frac{n}{m} = \frac{k \cdot n'}{k \cdot m'} \quad (3.5)$$

The sets \mathbb{N} , \mathbb{Z} and \mathbb{Q} have a notion of addition and multiplication defined on them. Most of us are probably able to do many integer computations in our head, but this gets more and more difficult as these increase in complexity. We will frequently invoke the SageMath system (2.7.1) for more complicated computations (We define rings and fields later in this book): **SB:** I would delete lines 12-18 form the Sage example below, unnecessarily confusing at this point

sage: ZZ # A sage notation for the integers

Integer Ring

sage: NN # A sage notation for the natural numbers

Non negative integer semiring

sage: QQ # A sage notation for the rational numbers

Rational Field

sage: ZZ(5) # Get an element from the integers

5

sage: ZZ(5) + ZZ(3)

8

sage: ZZ(5) * NN(3)

Move
content
on binary
representation

simplify
Sage ex.

```

941 15
942 sage: ZZ.random_element(10**50)
943 20086235761044088201572950207179628954288133546338
944 sage: ZZ(27713).str(2) # Binary string representation
945 110110001000001
946 sage: NN(27713).str(2) # Binary string representation
947 110110001000001
948 sage: ZZ(27713).str(16) # Hexadecimal string representation
949 6c41

```

A set of numbers of particular interest to us is the set of **prime numbers**, which are natural numbers $p \in \mathbb{N}$ with $p \geq 2$ that are only divisible by themselves and by 1. All prime numbers apart from the number 2 are called **odd** prime numbers. We use \mathbb{P} for the set of all prime numbers and $\mathbb{P}_{\geq 3}$ for the set of all odd prime numbers. The set of prime numbers \mathbb{P} is an infinite set, and it can be ordered according to size. This means that, for any prime number $p \in \mathbb{P}$, one can always find another prime number $p' \in \mathbb{P}$ with $p < p'$. Consequently, there is no largest prime number. Since prime numbers can be ordered by size, we can write them as follows:

$$2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, \dots \quad (3.6)$$

As the **fundamental theorem of arithmetic** tells us, prime numbers are, in a certain sense, the basic building blocks from which all other natural numbers are composed. To see that, let $n \in \mathbb{N}$ be any natural number with $n > 1$. Then there are always prime numbers $p_1, p_2, \dots, p_k \in \mathbb{P}$, such that the following equation holds:

$$n = p_1 \cdot p_2 \cdot \dots \cdot p_k \quad (3.7)$$

This representation is unique for each natural number (except for the order of the **factors** p_1, p_2, \dots, p_k) and is called the **prime factorization** of n .

Example 1 (Prime Factorization). To see what we mean by the prime factorization of a number, let's look at the number $504 \in \mathbb{N}$. To get its prime factors, we can successively divide it by all prime numbers in ascending order starting with 2:

$$504 = 2 \cdot 2 \cdot 2 \cdot 3 \cdot 3 \cdot 7$$

We can double check our findings invoking Sage, which provides an algorithm for factoring natural numbers:

```

966 sage: n = NN(504)
967 sage: factor(n)
968 2^3 * 3^2 * 7

```

The computation from the previous example reveals an important observation: computing the factorization of an integer is computationally expensive, because we have to divide repeatedly by all prime numbers smaller than the number itself until all factors are prime numbers themselves. From this, an important question arises: how fast can we compute the prime factorization of a natural number? This question is the famous **integer factorization problem** and, as far as we know, there is currently no known method that can factor integers much faster than the naive approach of just dividing the given number by all prime numbers in ascending order.

On the other hand, computing the product of a given set of prime numbers is fast: you just multiply all factors. This simple observation implies that the two processes "prime number

To see that

let's

"themselves is more common?"

you

multiplication" on the one side and its inverse process "natural number factorization" have very different computational costs. The factorization problem is therefore an example of a so-called **one-way function**: an invertible function that is easy to compute in one direction, but hard to compute in the other direction.¹

Exercise 1. What is the absolute value of the integers -123 , 27 and 0 ?

Exercise 2. Compute the factorization of 30030 and double check your results using Sage.

Exercise 3. Consider the following equation:

$$4 \cdot x + 21 = 5.$$

Compute the set of all solutions for x under the following alternative assumptions:

1. The equation is defined over the set of natural numbers.
2. The equation is defined over the set of integers.

Exercise 4. Consider the following equation:

$$2x^3 - x^2 - 2x = -1.$$

Compute the set of all solutions x under the following assumptions:

1. The equation is defined over the set of natural numbers.
2. The equation is defined over the set of integers.
3. The equation is defined over the set of rational numbers.

3.2.2 Euclidean Division

As we know from high school mathematics, integers can be added, subtracted and multiplied, and the result of these operations is guaranteed to always be an integer as well. On the contrary, division (in the commonly understood sense) is not defined for integers, as, for example, 7 divided by 3 will not result in an integer. However, it is always possible to divide any two integers if we consider **division with a remainder**. For example, 7 divided by 3 is equal to 2 with a remainder of 1 , since $7 = 2 \cdot 3 + 1$.

This section introduces division with a remainder for integers, usually called **Euclidean division**. It is an essential technique underlying many concepts in this book. The precise definition is as follows:

Let $a \in \mathbb{Z}$ and $b \in \mathbb{Z}$ be two integers with $b \neq 0$. Then there is always another integer $m \in \mathbb{Z}$ and a natural number $r \in \mathbb{N}$, with $0 \leq r < |b|$ such that the following holds:

$$a = m \cdot b + r \tag{3.8}$$

This decomposition of a given b is called **Euclidean division**, where a is called the **dividend**, b is called the **divisor**, m is called the **quotient** and r is called the **remainder**. It can be shown that both the quotient and the remainder always exist and are unique, as long as the divisor is different from 0 .

¹It should be pointed out, however, that the American mathematician Peter W. Shor developed an algorithm in 1994, which can calculate the prime factorization of a natural number in polynomial time on a quantum computer. The consequence of this is that cryptosystems, which are based on the prime factor problem, are unsafe as soon as practically usable quantum computers become available.

1007 *Notation and Symbols* 1. Suppose that the numbers a , b , m and r satisfy equation (3.8). We can
 1008 then describe the quotient and the remainder of the Euclidean division as follows:

$$a \operatorname{div} b := m, \quad a \operatorname{mod} b := r \quad (3.9)$$

1009 We also say that an integer a is **divisible** by another integer b if $a \operatorname{mod} b = 0$ holds. In this
 1010 case, we write $b|a$, and call the integer $a \operatorname{div} b$ the **cofactor** of b in a .

1011 So, in a nutshell, Euclidean division is the process of dividing one integer by another in a
 1012 way that produces a quotient and a non-negative remainder, the latter of which is smaller than
 1013 the absolute value of the divisor.

1014 *Example 2.* Applying Euclidean division and the notation defined in 3.9 to the dividend -17
 1015 and the divisor 4 , we get the following:

$$-17 \operatorname{div} 4 = -5, \quad -17 \operatorname{mod} 4 = 3 \quad (3.10)$$

1016 $-17 = -5 \cdot 4 + 3$ is the Euclidean division of -17 by 4 . The remainder, by definition, is a
 1017 non-negative number. In this case, 4 does not divide -17 , as the remainder is not zero. The truth
 1018 value of the expression $4|-17$ therefore is **FALSE**. On the other hand, the truth value of $4|12$ is
 1019 **TRUE**, since 4 divides 12 , as $12 \operatorname{mod} 4 = 0$. If we invoke Sage to do the computation for us, we
 1020 get the following:

```

1021 sage: ZZ(-17) // ZZ(4) # Integer quotient      24
1022 -5                                              25
1023 sage: ZZ(-17) % ZZ(4) # remainder             26
1024 3                                              27
1025 sage: ZZ(4).divides(ZZ(-17)) # self divides other 28
1026 False                                         29
1027 sage: ZZ(4).divides(ZZ(12))                   30
1028 True                                          31
```

1029 *Remark 1.* In 3.9, we defined the notation of $a \operatorname{div} b$ and $a \operatorname{mod} b$ in terms of Euclidean di-
 1030 vision. It should be noted, however, that many programming languages (like Python and Sage)
 1031 implement both the operator $(/)$ and the operator $(\%)$ differently. Programmers should be aware
 1032 of this, as the discrepancy between the mathematical notation and the implementation in pro-
 1033 gramming languages might become the source of subtle bugs in implementations of cryptographic
 1034 primitives.

1035 To give an example, consider the the dividend -17 and the divisor -4 . Note that, in contrast
 1036 to the previous example 2, we now have a negative divisor. According to our definition we have
 1037 the following:

$$-17 \operatorname{div} -4 = 5, \quad -17 \operatorname{mod} -4 = 3 \quad (3.11)$$

1038 $-17 = 5 \cdot (-4) + 3$ is the Euclidean division of -17 by -4 (the remainder is, by definition, a
 1039 non-negative number). However, using the operators $(/)$ and $(\%)$ in Sage, we get a different
 1040 result:

```

1041 sage: ZZ(-17) // ZZ(-4) # Integer quotient      32
1042 4                                              33
1043 sage: ZZ(-17) % ZZ(-4) # remainder             34
1044 -1                                             35
1045 sage: ZZ(-17).quo_rem(ZZ(-4)) # not Euclidean division 36
```

(4, -1)

Methods to compute Euclidean division for integers are called **integer division algorithms**. Probably the best known algorithm is the so-called **long division**, which most of us might have learned in school.

In a nutshell, the long division algorithm loops through the digits of the dividend from the left to right, subtracting the largest possible multiple of the divisor (at the digit level) at each stage. The multiples then become the digits of the quotient, and the remainder is the first digit of the dividend.

As long division is the standard method used for pen-and-paper division of multi-digit numbers expressed in decimal notation, we use it throughout this book when we do simple pen-and-paper computations, so readers should become familiar with it. However, instead of defining the algorithm formally, we provide some examples instead, as this will hopefully make the process more clear.

Example 3 (Integer Long Division). To give an example of integer long division algorithm, let's divide the integer $a = 143785$ by the number $b = 17$. Our goal is therefore to find solutions to equation 3.8, that is, we need to find the quotient $m \in \mathbb{Z}$ and the remainder $r \in \mathbb{N}$ such that $143785 = m \cdot 17 + r$. Using a notation that is mostly used in Commonwealth countries, we compute as follows: **SB: I think a more detailed explanation is needed for those unfamiliar with this notation/algorithm**

$$\begin{array}{r}
 8457 \\
 17 \overline{) 143785} \\
 \underline{136} \\
 77 \\
 \underline{68} \\
 98 \\
 \underline{85} \\
 135 \\
 \underline{119} \\
 16
 \end{array}
 \tag{3.12}$$

We calculated $m = 8457$ and $r = 16$, and, indeed, the equation $143785 = 8457 \cdot 17 + 16$ holds. We can double check this invoking Sage:

```

sage: ZZ(143785).quo_rem(ZZ(17))      38
      (8457, 16)                       39
sage: ZZ(143785) == ZZ(8457)*ZZ(17) + ZZ(16) # check 40
True                                   41

```

Exercise 5 (Integer Long Division). Find an $m \in \mathbb{Z}$ and an $r \in \mathbb{N}$ with $0 \leq r < |b|$ such that $a = m \cdot b + r$ holds for the following pairs:

- $(a, b) = (27, 5)$
- $(a, b) = (27, -5)$
- $(a, b) = (127, 0)$
- $(a, b) = (-1687, 11)$
- $(a, b) = (0, 7)$

In which cases are your solutions unique?

1079 *Exercise 6* (Long Division Algorithm). Using the programming language of your choice, write
 1080 an algorithm that computes integer long division and handles all edge cases properly.

1081 *Exercise 7* (Binary Representation). Using the programming language of your choice, write an
 1082 algorithm that computes the binary representation 3.4 of any non-negative integer.

1083 3.2.3 The Extended Euclidean Algorithm

1084 One of the most critical parts of this book is the modular arithmetic, defined in section 3.3,
 1085 and its application in the computations of **prime fields**, defined in section 4.3.1. To be able to
 1086 do computations in modular arithmetic, we have to get familiar with the so-called **Extended**
 1087 **Euclidean Algorithm**, used to calculate the **greatest common divisor** (GCD) of integers.

1088 The greatest common divisor of two non-zero integers a and b is defined as the largest non-
 1089 zero natural number d such that d divides both a and b , that is, $d|a$ as well as $d|b$ are true. We
 1090 use the notation $\gcd(a, b) := d$ for this number. Since the natural number 1 divides any other
 1091 integer, 1 is always a common divisor of any two non-zero integers, but it is not necessarily the
 1092 greatest.

1093 A common method for computing the greatest common divisor is the so-called Euclidean
 1094 Algorithm. However, since we don't need that algorithm in this book, we will introduce the
 1095 Extended Euclidean Algorithm, which is a method for calculating the greatest common divisor
 1096 of two natural numbers a and $b \in \mathbb{N}$, as well as two additional integers $s, t \in \mathbb{Z}$, such that the
 1097 following equation holds:

$$\gcd(a, b) = s \cdot a + t \cdot b \quad (3.13)$$

1098 The pseudocode in algorithm 1 shows in detail how to calculate the greatest common divisor
 1099 and the numbers s and t with the Extended Euclidean Algorithm:

Algorithm 1 Extended Euclidean Algorithm

Require: $a, b \in \mathbb{N}$ with $a \geq b$

procedure EXT-EUCLID(a, b)

$r_0 \leftarrow a$ and $r_1 \leftarrow b$

$s_0 \leftarrow 1$ and $s_1 \leftarrow 0$

$t_0 \leftarrow 0$ and $t_1 \leftarrow 1$

$k \leftarrow 2$

while $r_{k-1} \neq 0$ **do**

$q_k \leftarrow r_{k-2} \text{ div } r_{k-1}$

$r_k \leftarrow r_{k-2} \text{ mod } r_{k-1}$

$s_k \leftarrow s_{k-2} - q_k \cdot s_{k-1}$

$t_k \leftarrow t_{k-2} - q_k \cdot t_{k-1}$

$k \leftarrow k + 1$

end while

return $\gcd(a, b) \leftarrow r_{k-2}$, $s \leftarrow s_{k-2}$ and $t \leftarrow t_{k-2}$

end procedure

Ensure: $\gcd(a, b) = s \cdot a + t \cdot b$

1100 The algorithm is simple enough to be used effectively in pen-and-paper examples. It is
 1101 commonly written as a table where the rows represent the while-loop and the columns
 1102 represent the values of the the array r , s and t with index k . The following example provides a
 1103 simple execution.

Add more
explan-
ation

1104 *Example 4.* To illustrate algorithm 1, we apply it to the numbers $a = 12$ and $b = 5$. Since
 1105 $12, 5 \in \mathbb{N}$ and $12 \geq 5$, all requirements are met, and we compute as follows:

add more
explana-
tion

k	r_k	s_k	t_k
0	12	1	0
1	5	0	1
2	2	1	-2
3	1	-2	5
4	0		

1107 From this we can see that the greatest common divisor of 12 and 5 is $\gcd(12, 5) = 1$ and that
 1108 the equation $1 = (-2) \cdot 12 + 5 \cdot 5$ holds. We can also invoke sage to double check our findings:

```
1109 sage: ZZ(12).xgcd(ZZ(5)) # (gcd(a,b), s, t) 42
1110 (1, -2, 5) 43
```

1111 *Exercise 8* (Extended Euclidean Algorithm). Find integers $s, t \in \mathbb{Z}$ such that $\gcd(a, b) = s \cdot a +$
 1112 $t \cdot b$ holds for the following pairs $(a, b) = (45, 10)$, $(a, b) = (13, 11)$, $(a, b) = (13, 12)$. What
 1113 pairs (a, b) are coprime?

1114 *Exercise 9* (Towards Prime fields). Let $n \in \mathbb{N}$ be a natural number and p a prime number, such
 1115 that $n < p$. What is the greatest common divisor $\gcd(p, n)$?

1116 *Exercise 10.* Find all numbers $k \in \mathbb{N}$ with $0 \leq k \leq 100$ such that $\gcd(100, k) = 5$.

1117 *Exercise 11.* Show that $\gcd(n, m) = \gcd(n + m, m)$ for all $n, m \in \mathbb{N}$.

1118 **Coprime Integers** Coprime integers are integers that do not have a common prime number
 1119 as a factor. As we will see in 3.3 those numbers are important for our purposes because in
 1120 modular arithmetic, computation that involve coprime numbers are substantially different from
 1121 computations on non-coprime numbers 3.3.

1122 The naive way to decide if two integers are coprime would be to divide both number suces-
 1123 sively by all prime numbers smaller then those numbers to see if they share a common prime
 1124 factor. However two integers are coprime if and only if their greatest common divisor is 1 and
 1125 hence computing the \gcd is the preferred method.

1126 *Example 5.* Consider example 4 again. As we have seen, the greatest common divisor of 12
 1127 and 5 is 1. This implies that the integers 12 and 5 are coprime, since they share no divisor other
 1128 then 1, which is not a prime number.

1129 *Exercise 12.* Consider exercise 8 again. Which pairs (a, b) from that exercise are coprime?

1130 3.3 Modular arithmetic

1131 **Modular arithmetic** is a system of integer arithmetic, where numbers “wrap around” when
 1132 reaching a certain value, much like calculations on a clock wrap around whenever the value
 1133 exceeds the number 12. For example, if the clock shows that it is 11 o’clock, then 20 hours
 1134 later it will be 7 o’clock, not 31 o’clock. The number 31 has no meaning on a normal clock that
 1135 shows hours.

1136 The number at which the wrap occurs is called the **modulus**. Modular arithmetic general-
 1137 izes the clock example to arbitrary moduli and studies equations and phenomena that arise in
 1138 this new kind of arithmetic. It is of central importance for understanding most modern crypto

systems, in large parts because modular arithmetic provides the computational infrastructure for algebraic types that have cryptographically useful examples of one-way functions.

Although modular arithmetic appears very different from ordinary integer arithmetic that we are all familiar with, we encourage the interested reader to work through the example and to discover that, once they get used to the idea that this is a new kind of calculations, it will seem much less daunting.

Congruence In what follows, let $n \in \mathbb{N}$ with $n \geq 2$ be a fixed natural number that we will call the **modulus** of our modular arithmetic system. With such an n given, we can then group integers into classes, by saying that two integers are in the same class, whenever their Euclidean division 3.2.2 by n will give the same remainder. We then say that two numbers are **congruent** whenever they are in the same class.

Example 6. If we choose $n = 12$ as in our clock example, then the integers $-7, 5, 17$ and 29 are all congruent with respect to 12 , since all of them have the remainder 5 if we perform Euclidean division on them by 12 . In the picture of an analog 12-hour clock, starting at 5 o'clock, when we add 12 hours we are again at 5 o'clock, representing the number 17 . On the other hand, when we subtract 12 hours, we are at 5 o'clock again, representing the number -7 .

We can formalize this intuition of what congruence should be into a proper definition utilizing Euclidean division (as explained previously in 3.2): Let $a, b \in \mathbb{Z}$ be two integers and $n \in \mathbb{N}$ a natural number, such that $n \geq 2$. Then a and b are said to be **congruent with respect to the modulus** n , if and only if the following equation holds

$$a \bmod n = b \bmod n \quad (3.14)$$

If, on the other hand, two numbers are not congruent with respect to a given modulus n , we call them **incongruent** w.r.t. n .

A **congruence** is then nothing but an equation "up to congruence", which means that the equation only needs to hold if we take the modulus on both sides. In which case we write

$$a \equiv b \pmod{n} \quad (3.15)$$

Exercise 13. Which of the following pairs of numbers are congruent with respect to the modulus 13 : $(5, 19), (13, 0), (-4, 9), (0, 0)$.

Exercise 14. Find all integers x , such that the congruence $x \equiv 4 \pmod{6}$ is satisfied.

Computational Rules Having defined the notion of a congruence as an equation "up to a modulus", a follow up question is if we can manipulate a congruence similar to an equation. Indeed we can almost apply the same substitution rules to a congruency then to an equation, with the main difference being that for some non-zero integer $k \in \mathbb{Z}$, the congruence $a \equiv b \pmod{n}$ is equivalent to the congruence $k \cdot a \equiv k \cdot b \pmod{n}$ only, if k and the modulus n are coprime 3.2.3. The following list gives a set of useful rules:

Suppose that integers $a_1, a_2, b_1, b_2, k \in \mathbb{Z}$ are given. Then the following arithmetic rules hold for congruencies:

- $a_1 \equiv b_1 \pmod{n} \Leftrightarrow a_1 + k \equiv b_1 + k \pmod{n}$ (compatibility with translation)
- $a_1 \equiv b_1 \pmod{n} \Rightarrow k \cdot a_1 \equiv k \cdot b_1 \pmod{n}$ (compatibility with scaling)
- $\gcd(k, n) = 1$ and $k \cdot a_1 \equiv k \cdot b_1 \pmod{n} \Rightarrow a_1 \equiv b_1 \pmod{n}$

- 1177 • $k \cdot a_1 \equiv k \cdot b_1 \pmod{k \cdot n} \Rightarrow a_1 \equiv b_1 \pmod{n}$
- 1178 • $a_1 \equiv b_1 \pmod{n}$ and $a_2 \equiv b_2 \pmod{n} \Rightarrow a_1 + a_2 \equiv b_1 + b_2 \pmod{n}$ (compatibil-
- 1179 ity with addition)
- 1180 • $a_1 \equiv b_1 \pmod{n}$ and $a_2 \equiv b_2 \pmod{n} \Rightarrow a_1 \cdot a_2 \equiv b_1 \cdot b_2 \pmod{n}$ (compatibility
- 1181 with multiplication)

1182 Other rules, such as compatibility with subtraction, follow from the rules above. For example,
 1183 compatibility with subtraction follows from compatibility with scaling by $k = -1$ and compat-
 1184 ibility with addition.

1185 Another property of congruencies, not known in the traditional arithmetic of integers is
 1186 **Fermat's Little Theorem**. In simple words, it states that, in modular arithmetic, every number
 1187 raised to the power of a prime number modulus is congruent to the number itself. Or, to be more
 1188 precise, if $p \in \mathbb{P}$ is a prime number and $k \in \mathbb{Z}$ is an integer, then:

$$k^p \equiv k \pmod{p}, \quad (3.16)$$

1189 If k is coprime to p , then we can divide both sides of this congruence by k and rewrite the
 1190 expression into the equivalent form

$$k^{p-1} \equiv 1 \pmod{p} \quad (3.17)$$

1191 The following sage code computes example effects of Fermat's little theorem and highlights the
 1192 effects of the exponent k being coprime and not coprime to p :

```

1193 sage: ZZ(137).gcd(ZZ(64))                                     44
1194 1                                                            45
1195 sage: ZZ(64)^ZZ(137) % ZZ(137) == ZZ(64) % ZZ(137)        46
1196 True                                                         47
1197 sage: ZZ(64)^ZZ(137-1) % ZZ(137) == ZZ(1) % ZZ(137)        48
1198 True                                                         49
1199 sage: ZZ(1918).gcd(ZZ(137))                                   50
1200 137                                                         51
1201 sage: ZZ(1918)^ZZ(137) % ZZ(137) == ZZ(1918) % ZZ(137)     52
1202 True                                                         53
1203 sage: ZZ(1918)^ZZ(137-1) % ZZ(137) == ZZ(1) % ZZ(137)      54
1204 False                                                         55

```

1205 Let's compute an example that contains most of the concepts described in this section:

Example 7. Assume that we consider the modulus 6 and that our task is to solve the following congruence for $x \in \mathbb{Z}$

$$7 \cdot (2x + 21) + 11 \equiv x - 102 \pmod{6}$$

As many rules for congruencies are more or less same as for integers, we can proceed in a similar way as we would if we had an equation to solve. Since both sides of a congruence contain ordinary integers, we can rewrite the left side as follows: $7 \cdot (2x + 21) + 11 = 14x + 147 + 11 = 14x + 158$. We can therefore rewrite the congruence into the equivalent form

$$14x + 158 \equiv x - 102 \pmod{6}$$

In the next step we want to shift all instances of x to left and every other term to the right. So we apply the "compatibility with translation" rules two times. In a first step we choose $k = -x$

and in a second step we choose $k = -158$. Since “compatibility with translation” transforms a congruence into an equivalent form, the solution set will not change and we get

$$\begin{aligned} 14x + 158 &\equiv x - 102 \pmod{6} \Leftrightarrow \\ 14x - x + 158 - 158 &\equiv x - x - 102 - 158 \pmod{6} \Leftrightarrow \\ 13x &\equiv -260 \pmod{6} \end{aligned}$$

If our congruence would just be a normal integer equation, we would divide both sides by 13 to get $x = -20$ as our solution. However, in case of a congruence, we need to make sure that the modulus and the number we want to divide by are coprime first – only then will we get an equivalent expression (See rule XXX). So we need to find the greatest common divisor $\gcd(13, 6)$. Since 13 is prime and 6 is not a multiple of 13, we know that $\gcd(13, 6) = 1$, so these numbers are indeed coprime. We therefore compute

$$13x \equiv -260 \pmod{6} \Leftrightarrow x \equiv -20 \pmod{6}$$

Our task is now to find all integers x , such that x is congruent to -20 with respect to the modulus 6. So we have to find all x such

$$x \bmod 6 = -20 \bmod 6$$

Since $-4 \cdot 6 + 4 = -20$ we know $-20 \bmod 6 = 4$ and hence we know that $x = 4$ is a solution to this congruence. However, 22 is another solution since $22 \bmod 6 = 4$ as well, and so is -20 . In fact, there are infinitely many solutions given by the set

$$\{\dots, -8, -2, 4, 10, 16, \dots\} = \{4 + k \cdot 6 \mid k \in \mathbb{Z}\}$$

Putting all this together, we have shown that the every x from the set $\{x = 4 + k \cdot 6 \mid k \in \mathbb{Z}\}$ is a solution to the congruence $7 \cdot (2x + 21) + 11 \equiv x - 102 \pmod{6}$. We double ckeck for, say, $x = 4$ as well as $x = 4 + 12 \cdot 6 = 76$ using sage:

```

1209 sage: (ZZ(7) * (ZZ(2) * ZZ(4) + ZZ(21)) + ZZ(11)) % ZZ(6) == (ZZ 56
1210       (4) - ZZ(102)) % ZZ(6)
1211 True 57
1212 sage: (ZZ(7) * (ZZ(2) * ZZ(76) + ZZ(21)) + ZZ(11)) % ZZ(6) == ( 58
1213       ZZ(76) - ZZ(102)) % ZZ(6)
1214 True 59

```

Readers who had not been familiar with modular arithmetic until now and who might be discouraged by how complicated modular arithmetic seems at this point, should keep two things in mind. First, computing congruencies in modular arithmetic is not really more complicated than computations in more familiar number systems (e.g. rational numbers), it is just a matter of getting used to it. Second, once we introduce the idea of remainder class representations 3.3, computations become conceptually cleaner and more easy to handle.

Exercise 15. Consider the modulus 13 and find all solutions $x \in \mathbb{Z}$ to the following congruence $5x + 4 \equiv 28 + 2x \pmod{13}$

Exercise 16. Consider the modulus 23 and find all solutions $x \in \mathbb{Z}$ to the following congruence $69x \equiv 5 \pmod{23}$

Exercise 17. Consider the modulus 23 and find all solutions $x \in \mathbb{Z}$ to the following congruence $69x \equiv 46 \pmod{23}$

- 1227 *Exercise 18.* Let a, b, k be integers, such that $a \equiv b \pmod{n}$ holds. Show $a^k \equiv b^k \pmod{n}$.
 1228 *Exercise 19.* Let a, n be integers, such that a and n are not coprime. For which $b \in \mathbb{Z}$ does the
 1229 congruence $a \cdot x \equiv b \pmod{n}$ have a solution x and how does the solution set look in that
 1230 case?

1231 **The Chinese Remainder Theorem** We have seen how to solve congruencies in modular
 1232 arithmetic. However, one question that remains is how to solve systems of congruencies with
 1233 different moduli? The answer is given by the **Chinese remainder theorem**, which states that
 1234 for any $k \in \mathbb{N}$ and coprime natural numbers $n_1, \dots, n_k \in \mathbb{N}$ as well as integers $a_1, \dots, a_k \in \mathbb{Z}$, the
 1235 so-called **simultaneous congruences**

$$\begin{aligned} x &\equiv a_1 \pmod{n_1} \\ x &\equiv a_2 \pmod{n_2} \\ &\dots \\ x &\equiv a_k \pmod{n_k} \end{aligned} \tag{3.18}$$

1236 has a solution, and all possible solutions of this congruence system are congruent modulo the
 product $N = n_1 \cdot \dots \cdot n_k$.² In fact, the following algorithm computes the solution set:

Algorithm 2 Chinese Remainder Theorem

Require: $k \in \mathbb{Z}$, $j \in \mathbb{N}_0$ and $n_0, \dots, n_{k-1} \in \mathbb{N}$ coprime

procedure CONGRUENCE-SYSTEMS-SOLVER(a_0, \dots, a_{k-1})

$N \leftarrow n_0 \cdot \dots \cdot n_{k-1}$

while $j < k$ **do**

$N_j \leftarrow N / n_j$

$(-, s_j, t_j) \leftarrow EXT - EUCLID(N_j, n_j)$

$$\triangleright 1 = s_j \cdot N_j + t_j \cdot n_j$$

end while

$x' \leftarrow \sum_{j=0}^{k-1} a_j \cdot s_j \cdot N_j$

$x \leftarrow x' \bmod N$

return $\{x + m \cdot N \mid m \in \mathbb{Z}\}$

end procedure

Ensure: $\{x + m \cdot N \mid m \in \mathbb{Z}\}$ is the complete solution set to 3.18.

check
algorithm
floating

1237

Example 8. To illustrate how to solve simultaneous congruences using the Chinese remainder theorem, let's look at the following system of congruencies:

$$\begin{aligned} x &\equiv 4 \pmod{7} \\ x &\equiv 1 \pmod{3} \\ x &\equiv 3 \pmod{5} \\ x &\equiv 0 \pmod{11} \end{aligned}$$

Clearly all moduli are coprime and we have $N = 7 \cdot 3 \cdot 5 \cdot 11 = 1155$, as well as $N_1 = 165$, $N_2 = 385$, $N_3 = 231$ and $N_4 = 105$. From this we calculate with the Extended Euclidean Algorithm

$$\begin{aligned} 1 &= 2 \cdot 165 + -47 \cdot 7 \\ 1 &= 1 \cdot 385 + -128 \cdot 3 \\ 1 &= 1 \cdot 231 + -46 \cdot 5 \\ 1 &= 2 \cdot 105 + -19 \cdot 11 \end{aligned}$$

²This is the classical Chinese remainder theorem as it was already known in ancient China. Under certain circumstances, the theorem can be extended to non-coprime moduli n_1, \dots, n_k but this is beyond the scope of this book. Interested readers should consult XXX [add references](#)

so we have $x = 4 \cdot 2 \cdot 165 + 1 \cdot 1 \cdot 385 + 3 \cdot 1 \cdot 231 + 0 \cdot 2 \cdot 105 = 2398$ as one solution. Because $2398 \bmod 1155 = 88$ the set of all solutions is $\{\dots, -2222, -1067, 88, 1243, 2398, \dots\}$. We can invoke Sage's computation of the Chinese Remainder Theorem (CRT) to double check our findings:

```
1242 sage: CRT_list([4,1,3,0], [7,3,5,11]) 60
1243 88 61
```

Remainder Class Representation As we have seen in various examples before, computing congruencies can be cumbersome and solution sets are large in general. It is therefore advantageous to find some kind of simplification for modular arithmetic.

Fortunately, this is possible and relatively straightforward once we identify each set of numbers with equal remainder with that remainder itself and call it the **remainder class** or **residue class** representation in modulo n arithmetic.

It then follows from the properties of Euclidean division that there are exactly n different remainder classes for every modulus n and that integer addition and multiplication can be projected to a new kind of addition and multiplication on those classes.

Roughly speaking, the new rules for addition and multiplication are then computed by taking any element of the first remainder class and some element of the second, then add or multiply them in the usual way and see which remainder class the result is contained in. The following example makes this abstract description more concrete:

Example 9 (Arithmetic modulo 6). Choosing the modulus $n = 6$, we have six remainder classes of integers which are congruent modulo 6 (they have the same remainder when divided by 6) and when we identify each of those remainder classes with the remainder, we get the following identification:

$$\begin{aligned} 0 &:= \{\dots, -6, 0, 6, 12, \dots\} \\ 1 &:= \{\dots, -5, 1, 7, 13, \dots\} \\ 2 &:= \{\dots, -4, 2, 8, 14, \dots\} \\ 3 &:= \{\dots, -3, 3, 9, 15, \dots\} \\ 4 &:= \{\dots, -2, 4, 10, 16, \dots\} \\ 5 &:= \{\dots, -1, 5, 11, 17, \dots\} \end{aligned}$$

Now to compute the new addition law of those remainder class representatives, say $2 + 5$, one chooses arbitrary elements from both classes, say 14 and -1 , adds those numbers in the usual way and then looks at the remainder class of the result.

So we get $14 + (-1) = 13$, and 13 is in the remainder class (of) 1. Hence we find that $2 + 5 = 1$ in modular 6 arithmetic, which is a more readable way to write the congruence $2 + 5 \equiv 1 \pmod{6}$.

Applying the same reasoning to all remainder classes, addition and multiplication can be transferred to the representatives of the remainder classes. The results for modulus 6 arithmetic are summarized in the following addition and multiplication tables:

+	0	1	2	3	4	5	·	0	1	2	3	4	5
0	0	1	2	3	4	5	0	0	0	0	0	0	0
1	1	2	3	4	5	0	1	0	1	2	3	4	5
2	2	3	4	5	0	1	2	0	2	4	0	2	4
3	3	4	5	0	1	2	3	0	3	0	3	0	3
4	4	5	0	1	2	3	4	0	4	2	0	4	2
5	5	0	1	2	3	4	5	0	5	4	3	2	1

1267 This way, we have defined a new arithmetic system that contains just 6 numbers and comes with
 1268 its own definition of addition and multiplication. We call it **modular 6 arithmetic** and write
 1269 the associated type as \mathbb{Z}_6 .

1270 To see why such an identification of a remainder class with its remainder is useful and
 1271 actually simplifies congruence computations a lot, let's go back to the congruence from example
 1272 7 again:

$$7 \cdot (2x + 21) + 11 \equiv x - 102 \pmod{6} \quad (3.19)$$

1273 As shown in example 7, the arithmetic of congruencies can deviate from ordinary arithmetic:
 1274 For example, division needs to check whether the modulus and the dividend are coprimes, and
 1275 solutions are not unique in general.

We can rewrite this congruence as an **equation** over our new arithmetic type \mathbb{Z}_6 by **projecting onto the remainder classes**. In particular, since $7 \bmod 6 = 1$, $21 \bmod 6 = 3$, $11 \bmod 6 = 5$ and $102 \bmod 6 = 0$ we have

$$\begin{aligned} 7 \cdot (2x + 21) + 11 \equiv x - 102 \pmod{6} \text{ over } \mathbb{Z} \\ \Leftrightarrow 1 \cdot (2x + 3) + 5 = x \text{ over } \mathbb{Z}_6 \end{aligned}$$

1276 We can use the multiplication and addition table above to solves the equation on the right like
 1277 we would solve normal integer equations:

$$\begin{aligned} 1 \cdot (2x + 3) + 5 &= x \\ 2x + 3 + 5 &= x && \# \text{ addition-table: } 3 + 5 = 2 \\ 2x + 2 &= x && \# \text{ add 4 and } -x \text{ on both sides} \\ 2x + 2 + 4 - x &= x + 4 - x && \# \text{ addition-table: } 2 + 4 = 0 \\ x &= 4 \end{aligned}$$

1278 As we can see, despite the somewhat unfamiliar rules of addition and multiplication, solving
 1279 congruencies this way is very similar to solving normal equations. And, indeed, the solution
 1280 set is identical to the solution set of the original congruence, since 4 is identified with the set
 1281 $\{4 + 6 \cdot k \mid k \in \mathbb{Z}\}$.

1282 We can invoke Sage to do computations in our modular 6 arithmetic type. This is particularly
 1283 useful to double-check our computations:

```
1284 sage: Z6 = Integers(6)                                     62
1285 sage: Z6(2) + Z6(5)                                       63
1286 1                                                         64
1287 sage: Z6(7) * (Z6(2) * Z6(4) + Z6(21)) + Z6(11) == Z6(4) - Z6(102) 65
1288 True                                                       66
```

1289 *Remark 2 (k-bit modulus).* In cryptographic papers, we sometimes read phrases like “[...] using
 1290 a 4096-bit modulus”. This means that the underlying modulus n of the modular arithmetic used
 1291 in the system has a binary representation with a length of 4096 bits. In contrast, the number 6
 1292 has the binary representation 110 and hence our example 9 describes a 3-bit modulus arithmetic
 1293 system.

1294 *Exercise 20.* Define \mathbb{Z}_{13} as the the arithmetic modulo 13 analog to example 9. Then consider
 1295 the congruence from exercise 15 and rewrite it into an equation in \mathbb{Z}_{13} .

Modular Inverses As we know, integers can be added, subtracted and multiplied so that the result is also an integer, but this is not true for the division of integers in general: for example, $3/2$ is not an integer anymore. To see why this is, from a more theoretical perspective, let us consider the definition of a multiplicative inverse first. When we have a set that has some kind of multiplication defined on it and we have a distinguished element of that set that behaves neutrally with respect to that multiplication (doesn't change anything when multiplied with any other element), then we can define **multiplicative inverses** in the following way:

Let S be our set that has some notion $a \cdot b$ of multiplication and a **neutral element** $1 \in S$, such that $1 \cdot a = a$ for all elements $a \in S$. Then a **multiplicative inverse** a^{-1} of an element $a \in S$ is defined as follows:

$$a \cdot a^{-1} = 1 \quad (3.20)$$

Informally speaking, the definition of a multiplicative inverse means that it “cancels” the original element to give 1 when they are multiplied.

Numbers that have multiplicative inverses are of particular interest, because they immediately lead to the definition of division by those numbers. In fact, if a is number such that the multiplicative inverse a^{-1} exists, then we define **division** by a simply as multiplication by the inverse:

$$\frac{b}{a} := b \cdot a^{-1} \quad (3.21)$$

Example 10. Consider the set of rational numbers, also known as fractions, \mathbb{Q} . For this set, the neutral element of multiplication is 1, since $1 \cdot a = a$ for all rational numbers. For example, $1 \cdot 4 = 4$, $1 \cdot \frac{1}{4} = \frac{1}{4}$, or $1 \cdot 0 = 0$ and so on.

Every rational number $a \neq 0$ has a multiplicative inverse, given by $\frac{1}{a}$. For example, the multiplicative inverse of 3 is $\frac{1}{3}$, since $3 \cdot \frac{1}{3} = 1$, the multiplicative inverse of $\frac{5}{7}$ is $\frac{7}{5}$, since $\frac{5}{7} \cdot \frac{7}{5} = 1$, and so on.

Example 11. Looking at the set \mathbb{Z} of integers, we see that with respect to multiplication the neutral element is the number 1 and we notice that no integer other than 1 or -1 has a multiplicative inverse, since the equation $a \cdot x = 1$ has no integer solutions for $a \neq 1$ or $a \neq -1$.

The definition of multiplicative inverse works verbatim for addition as well where it is called the additive inverse. In the case of integers, the neutral element with respect to addition is 0, since $a + 0 = a$ for all integers $a \in \mathbb{Z}$. The additive inverse always exist and is given by the negative number $-a$, since $a + (-a) = 0$.

Example 12. Looking at the set \mathbb{Z}_6 of residual classes modulo 6 from example 9, we can use the multiplication table to find multiplicative inverses. To do so, we look at the row of the element and then find the entry equal to 1. If such an entry exists, the element of that column is the multiplicative inverse. If, on the other hand, the row has no entry equal to 1, we know that the element has no multiplicative inverse.

For example in \mathbb{Z}_6 the multiplicative inverse of 5 is 5 itself, since $5 \cdot 5 = 1$. We can also see that 5 and 1 are the only elements that have multiplicative inverses in \mathbb{Z}_6 .

Now, since 5 has a multiplicative inverse in modulo 6 arithmetic, we can divide by 5 in \mathbb{Z}_6 , since we have a notation of multiplicative inverse and division is nothing but multiplication by the multiplicative inverse. For example

$$\frac{4}{5} = 4 \cdot 5^{-1} = 4 \cdot 5 = 2$$

From the last example, we can make the interesting observation that while 5 has no multiplicative inverse as an integer, it has a multiplicative inverse in modular 6 arithmetic.

1334 This raises the question which numbers have multiplicative inverses in modular arithmetic.
 1335 The answer is that, in modular n arithmetic, a number r has a multiplicative inverse, if and only
 1336 if n and r are coprime. Since $\gcd(n, r) = 1$ in that case, we know from the Extended Euclidean
 1337 Algorithm that there are numbers s and t , such that

$$1 = s \cdot n + t \cdot r \quad (3.22)$$

1338 If we take the modulus n on both sides, the term $s \cdot n$ vanishes, which tells us that $t \bmod n$ is the
 1339 multiplicative inverse of r in modular n arithmetic.

1340 *Example 13* (Multiplicative inverses in \mathbb{Z}_6). In the previous example, we looked up multiplica-
 1341 tive inverses in \mathbb{Z}_6 from the lookup-table in Example 9. In real world examples, it is usually
 1342 impossible to write down those lookup tables, as the modulus is way too large, and the sets
 1343 occasionally contain more elements than there are atoms in the observable universe.

1344 Now, trying to determine that $2 \in \mathbb{Z}_6$ has no multiplicative inverse in \mathbb{Z}_6 without using the
 1345 lookup table, we immediately observe that 2 and 6 are not coprime, since their greatest common
 1346 divisor is 2. It follows that equation 3.22 has no solutions s and t , which means that 2 has no
 1347 multiplicative inverse in \mathbb{Z}_6 .

1348 The same reasoning works for 3 and 4, as neither of these are coprime with 6. The case
 1349 of 5 is different, since $\gcd(6, 5) = 1$. To compute the multiplicative inverse of 5, we use the
 1350 Extended Euclidean Algorithm and compute the following:

k	r_k	s_k	$t_k = (r_k - s_k \cdot a) \div b$
0	6	1	0
1	5	0	1
2	1	1	-1
3	0	.	.

1352 We get $s = 1$ as well as $t = -1$ and have $1 = 1 \cdot 6 - 1 \cdot 5$. From this, it follows that $-1 \bmod 6 =$
 1353 5 is the multiplicative inverse of 5 in modular 6 arithmetic. We can double check using Sage:

1354 **sage:** `ZZ(6).xgcd(ZZ(5))` 67
 1355 `(1, 1, -1)` 68

At this point, the attentive reader might notice that the situation where the modulus is a prime number is of particular interest, because we know from exercise 9 that in these cases all remainder classes must have modular inverses, since $\gcd(r, n) = 1$ for prime n and any $r < n$. In fact, Fermat's little theorem provides a way to compute multiplicative inverses in this situation, since in case of a prime modulus p and $r < p$, we get the following:

$$\begin{aligned} r^p &\equiv r \pmod{p} \Leftrightarrow \\ r^{p-1} &\equiv 1 \pmod{p} \Leftrightarrow \\ r \cdot r^{p-2} &\equiv 1 \pmod{p} \end{aligned}$$

1356 This tells us that the multiplicative inverse of a residue class r in modular p arithmetic is pre-
 1357 cisely r^{p-2} .

Example 14 (Modular 5 arithmetic). To see the unique properties of modular arithmetic when the modulus is a prime number, we will replicate our findings from example 9, but this time for the prime modulus 5. For $n = 5$ we have five equivalence classes of integers which are

congruent modulo 5. We write this as follows:

$$\begin{aligned} 0 &:= \{\dots, -5, 0, 5, 10, \dots\} \\ 1 &:= \{\dots, -4, 1, 6, 11, \dots\} \\ 2 &:= \{\dots, -3, 2, 7, 12, \dots\} \\ 3 &:= \{\dots, -2, 3, 8, 13, \dots\} \\ 4 &:= \{\dots, -1, 4, 9, 14, \dots\} \end{aligned}$$

1358 Addition and multiplication can be transferred to the equivalence classes, in a way exactly
1359 parallel to Example 9. This results in the following addition and multiplication tables:

	+	0	1	2	3	4		·	0	1	2	3	4
	0	0	1	2	3	4		0	0	0	0	0	0
	1	1	2	3	4	0		1	0	1	2	3	4
1360	2	2	3	4	0	1		2	0	2	4	1	3
	3	3	4	0	1	2		3	0	3	1	4	2
	4	4	0	1	2	3		4	0	4	3	2	1

1361 Calling the set of remainder classes in modular 5 arithmetic with this addition and multiplication
1362 \mathbb{Z}_5 , we see some subtle but important differences to the situation in \mathbb{Z}_6 . In particular, we see
1363 that in the multiplication table, every remainder $r \neq 0$ has the entry 1 in its row and therefore
1364 has a multiplicative inverse. In addition, there are no non-zero elements such that their product
1365 is zero.

1366 To use Fermat's little theorem in \mathbb{Z}_5 for computing multiplicative inverses (instead of using
1367 the multiplication table), let's consider $3 \in \mathbb{Z}_5$. We know that the multiplicative inverse is given
1368 by the remainder class that contains $3^{5-2} = 3^3 = 3 \cdot 3 \cdot 3 = 4 \cdot 3 = 2$. And indeed $3^{-1} = 2$, since
1369 $3 \cdot 2 = 1$ in \mathbb{Z}_5 .

1370 We can invoke Sage to do computations in our modular 5 arithmetic type to double-check
1371 our computations:

```

1372 sage: Z5 = Integers(5)                                     69
1373 sage: Z5(3) ** (5-2)                                       70
1374 2                                                         71
1375 sage: Z5(3) ** (-1)                                       72
1376 2                                                         73
1377 sage: Z5(3) ** (5-2) == Z5(3) ** (-1)                   74
1378 True                                                       75

```

Example 15. To understand one of the principal differences between prime number modular arithmetic and non-prime number modular arithmetic, consider the linear equation $a \cdot x + b = 0$ defined over both types \mathbb{Z}_5 and \mathbb{Z}_6 . Since in \mathbb{Z}_5 every non-zero element has a multiplicative inverse, we can always solve these equations in \mathbb{Z}_5 , which is not true in \mathbb{Z}_6 . To see that, consider the equation $3x + 3 = 0$. In \mathbb{Z}_5 we have the following:

$$\begin{aligned} 3x + 3 &= 0 && \# \text{ add 2 and on both sides} \\ 3x + 3 + 2 &= 2 && \# \text{ addition-table: } 2 + 3 = 0 \\ 3x &= 2 && \# \text{ divide by 3 (which equals multiplication by 2)} \\ 2 \cdot (3x) &= 2 \cdot 2 && \# \text{ multiplication-table: } 2 \cdot 2 = 4 \\ x &= 4 \end{aligned}$$

So in the case of our prime number modular arithmetic, we get the unique solution $x = 4$. Now consider \mathbb{Z}_6 :

$$\begin{array}{ll} 3x + 3 = 0 & \# \text{ add 3 and on both sides} \\ 3x + 3 + 3 = 3 & \# \text{ addition-table: } 3 + 3 = 0 \\ 3x = 3 & \# \text{ division not possible (no multiplicative inverse of 3 exists)} \end{array}$$

So, in this case, we cannot solve the equation for x by dividing by 3. And, indeed, when we look at the multiplication table of \mathbb{Z}_6 (Example 9), we find that there are three solutions $x \in \{1, 3, 5\}$, such that $3x + 3 = 0$ holds true for all of them.

Exercise 21. Consider the modulus $n = 24$. Which of the integers 7, 1, 0, 805, -4255 have multiplicative inverses in modular 24 arithmetic? Compute the inverses, in case they exist.

Exercise 22. Find the set of all solutions to the congruence $17(2x + 5) - 4 \equiv 2x + 4 \pmod{5}$. Then project the congruence into \mathbb{Z}_5 and solve the resulting equation in \mathbb{Z}_5 . Compare the results.

Exercise 23. Find the set of all solutions to the congruence $17(2x + 5) - 4 \equiv 2x + 4 \pmod{6}$. Then project the congruence into \mathbb{Z}_6 and try to solve the resulting equation in \mathbb{Z}_6 .

3.4 Polynomial arithmetic

A polynomial is an expression consisting of variables (also-called indeterminates) and coefficients that involves only the operations of addition, subtraction and multiplication. All coefficients of a polynomial must have the same type, e.g. being integers or rational numbers etc. To be more precise an *univariate polynomial* is an expression

$$P(x) := \sum_{j=0}^m a_j x^j = a_m x^m + a_{m-1} x^{m-1} + \cdots + a_1 x + a_0, \quad (3.23)$$

where x is called the **indeterminate**, each a_j is called a **coefficient**. If R is the type of the coefficients, then the set of all **univariate³ polynomials with coefficients in R** is written as $R[x]$. We often simply use **polynomial** instead of univariate polynomial, write $P(x) \in R[x]$ for a polynomial and denote the constant term a_0 as $P(0)$.

A polynomial is called the **zero polynomial** if all coefficients are zero and a polynomial is called the **one polynomial** if the constant term is 1 and all other coefficients are zero.

Given an univariate polynomial $P(x) = \sum_{j=0}^m a_j x^j$ that is not the zero polynomial, we call the non-negative integer $\deg(P) := m$ the *degree* of P and define the degree of the zero polynomial to be $-\infty$, where $-\infty$ (negative infinity) is a symbol with the properties that $-\infty + m = -\infty$ and $-\infty < m$ for all non-negative integers $m \in \mathbb{N}_0$. In addition, we write

$$Lc(P) := a_m \quad (3.24)$$

and call it the **leading coefficient** of the polynomial P . We can restrict the set $R[x]$ of all polynomials with coefficients in R , to the set of all such polynomials that have a degree that does not exceed a certain value. If m is the maximum degree allowed, we write $R_{\leq m}[x]$ for the set of all polynomials with a degree less than or equal to m .

³in our context the term univariate means that the polynomial contains a single variable only

Example 16 (Integer Polynomials). The coefficients of a polynomial must all have the same type. The set of polynomials with integer coefficients is written as $\mathbb{Z}[x]$. Examples of such polynomials are:

$$\begin{array}{ll}
 P_1(x) = 2x^2 - 4x + 17 & \# \text{ with } \deg(P_1) = 2 \text{ and } Lc(P_1) = 2 \\
 P_2(x) = x^{23} & \# \text{ with } \deg(P_2) = 23 \text{ and } Lc(P_2) = 1 \\
 P_3(x) = x & \# \text{ with } \deg(P_3) = 1 \text{ and } Lc(P_3) = 1 \\
 P_4(x) = 174 & \# \text{ with } \deg(P_4) = 0 \text{ and } Lc(P_4) = 174 \\
 P_5(x) = 1 & \# \text{ with } \deg(P_5) = 0 \text{ and } Lc(P_5) = 1 \\
 P_6(x) = 0 & \# \text{ with } \deg(P_6) = -\infty \text{ and } Lc(P_6) = 0 \\
 P_7(x) = (x-2)(x+3)(x-5)
 \end{array}$$

In particular, every integer can be seen as an integer polynomial of degree zero. P_7 is a polynomial, because we can expand its definition into $P_7(x) = x^3 - 4x^2 - 11x + 30$, which is a polynomial of degree 3 and leading coefficient 1. The following expressions are not integer polynomials:

$$\begin{aligned}
 Q_1(x) &= 2x^2 + 4 + 3x^{-2} \\
 Q_2(x) &= 0.5x^4 - 2x \\
 Q_3(x) &= 2^x
 \end{aligned}$$

In particular Q_1 is not an integer polynomial, because the expression x^{-2} has a negative exponent, Q_2 is not an integer polynomial because the coefficient 0.5 is not an integer and Q_3 is not an integer polynomial because the indeterminate appears in the exponent of a coefficient.

We can invoke Sage to do computations with polynomials. To do so, we have to specify the symbol for the indeterminate and the type for the coefficients (For the definition of rings see 4.2). Note, however that Sage defines the degree of the zero polynomial to be -1 .

```

1413 sage: Zx = ZZ['x'] # integer polynomials with indeterminate x 76
1414 sage: Zt.<t> = ZZ[] # integer polynomials with indeterminate t 77
1415 sage: Zx 78
1416 Univariate Polynomial Ring in x over Integer Ring 79
1417 sage: Zt 80
1418 Univariate Polynomial Ring in t over Integer Ring 81
1419 sage: p1 = Zx([17,-4,2]) 82
1420 sage: p1 83
1421 2*x^2 - 4*x + 17 84
1422 sage: p1.degree() 85
1423 2 86
1424 sage: p1.leading_coefficient() 87
1425 2 88
1426 sage: p2 = Zt(t^23) 89
1427 sage: p2 90
1428 t^23 91
1429 sage: p6 = Zx([0]) 92
1430 sage: p6.degree() 93
1431 -1 94

```

Example 17 (Polynomials over \mathbb{Z}_6). Recall the definition of modular 6 arithmetics \mathbb{Z}_6 as defined in example 9. The set of all polynomials with indeterminate x and coefficients in \mathbb{Z}_6 is symbolized as $\mathbb{Z}_6[x]$. Example of polynomials from $\mathbb{Z}_6[x]$ are:

$$\begin{array}{ll}
 P_1(x) = 2x^2 - 4x + 5 & \# \text{ with } \deg(P_1) = 2 \text{ and } Lc(P_1) = 2 \\
 P_2(x) = x^{23} & \# \text{ with } \deg(P_2) = 23 \text{ and } Lc(P_2) = 1 \\
 P_3(x) = x & \# \text{ with } \deg(P_3) = 1 \text{ and } Lc(P_3) = 1 \\
 P_4(x) = 3 & \# \text{ with } \deg(P_4) = 0 \text{ and } Lc(P_4) = 3 \\
 P_5(x) = 1 & \# \text{ with } \deg(P_5) = 0 \text{ and } Lc(P_5) = 1 \\
 P_6(x) = 0 & \# \text{ with } \deg(P_6) = -\infty \text{ and } Lc(P_6) = 0 \\
 P_7(x) = (x-2)(x+3)(x-5) &
 \end{array}$$

Just like in the previous example, P_7 is a polynomial. However, since we are working with coefficients from \mathbb{Z}_6 now the expansion of P_7 is computed differently, as we have to invoke addition and multiplication in \mathbb{Z}_6 as defined in XXX. We get the following:

$$\begin{aligned}
 (x-2)(x+3)(x-5) &= (x+4)(x+3)(x+1) && \# \text{ additive inverses in } \mathbb{Z}_6 \\
 &= (x^2 + 4x + 3x + 3 \cdot 4)(x+1) && \# \text{ bracket expansion} \\
 &= (x^2 + 1x + 0)(x+1) && \# \text{ computation in } \mathbb{Z}_6 \\
 &= x^3 + x^2 + x^2 + x && \# \text{ bracket expansion} \\
 &= x^3 + 2x^2 + x
 \end{aligned}$$

1432 Again, we can use Sage to do computations with polynomials that have their coefficients in \mathbb{Z}_6
 1433 (For the definition of rings see 4.2). To do so, we have to specify the symbol for the indertemi-
 1434 nate and the type for the coefficients:

```

1435 sage: Z6 = Integers(6)                                     95
1436 sage: Z6x = Z6['x']                                       96
1437 sage: Z6x                                                 97
1438 Univariate Polynomial Ring in x over Ring of integers modulo 6 98
1439 sage: p1 = Z6x([5, -4, 2])                                99
1440 sage: p1                                                  100
1441 2*x^2 + 2*x + 5                                           101
1442 sage: p1 = Z6x([17, -4, 2])                               102
1443 sage: p1                                                  103
1444 2*x^2 + 2*x + 5                                           104
1445 sage: Z6x(x-2)*Z6x(x+3)*Z6x(x-5) == Z6x(x^3 + 2*x^2 + x) 105
1446 True                                                    106

```

1447 Given some element from the same type as the coefficients of a polynomial, the polyno-
 1448 mial can be evaluated at that element, which means that we insert the given element for every
 1449 occurrence of the indeterminate x in the polynomial expression.

1450 To be more precise, let $P \in R[x]$, with $P(x) = \sum_{j=0}^m a_j x^j$ be a polynomial with a coefficient
 1451 of type R and let $b \in R$ be an element of that type. Then the **evaluation** of P at b is given as
 1452 follows:

$$P(b) = \sum_{j=0}^m a_j b^j \quad (3.25)$$

Example 18. Consider the integer polynomials from example 16 again. To evaluate them at given points, we have to insert the point for all occurrences of x in the polynomial expression. Inserting arbitrary values from \mathbb{Z} , we get:

$$\begin{aligned} P_1(2) &= 2 \cdot 2^2 - 4 \cdot 2 + 17 = 17 \\ P_2(3) &= 3^{23} = 94143178827 \\ P_3(-4) &= -4 = -4 \\ P_4(15) &= 174 \\ P_5(0) &= 1 \\ P_6(1274) &= 0 \\ P_7(-6) &= (-6-2)(-6+3)(-6-5) = -264 \end{aligned}$$

1453 Note, however, that it is not possible to evaluate any of those polynomial on values of different
1454 type. For example, it is not strictly correct to write $P_1(0.5)$, since 0.5 is not an integer. We can
1455 verify our computations using Sage:

```
1456 sage: Zx = ZZ['x'] 107
1457 sage: p1 = Zx([17, -4, 2]) 108
1458 sage: p7 = Zx(x-2)*Zx(x+3)*Zx(x-5) 109
1459 sage: p1(ZZ(2)) 110
1460 17 111
1461 sage: p7(ZZ(-6)) == ZZ(-264) 112
1462 True 113
```

Example 19. Consider the polynomials with coefficients in \mathbb{Z}_6 from example again. To evaluate them at given values from \mathbb{Z}_6 , we have to insert the point for all occurrences of x in the polynomial expression. We get the following:

$$\begin{aligned} P_1(2) &= 2 \cdot 2^2 - 4 \cdot 2 + 5 = 2 - 2 + 5 = 5 \\ P_2(3) &= 3^{23} = 3 \\ P_3(-4) &= P_3(2) = 2 \\ P_5(0) &= 1 \\ P_6(4) &= 0 \end{aligned}$$

```
1463
1464 sage: Z6 = Integers(6) 114
1465 sage: Z6x = Z6['x'] 115
1466 sage: p1 = Z6x([5, -4, 2]) 116
1467 sage: p1(Z6(2)) == Z6(5) 117
1468 True 118
```

1469 *Exercise 24.* Compare both expansions of P_7 from $\mathbb{Z}[x]$ and from $\mathbb{Z}_6[x]$ in example 16 and
1470 example 19, and consider the definition of \mathbb{Z}_6 as given in example 9. Can you see how the
1471 definition of P_7 over \mathbb{Z} projects to the definition over \mathbb{Z}_6 if you consider the residue classes of
1472 \mathbb{Z}_6 ?

Polynomial arithmetic Polynomials behave like integers in many ways. In particular, they can be added, subtracted and multiplied. In addition, they have their own notion of Euclidean division. Informally speaking, we can add two polynomials by simply adding the coefficients of the same index, and we can multiply them by applying the distributive property, that is, by multiplying every term of the left factor with every term of the right factor and adding the results together.

To be more precise let $\sum_{n=0}^{m_1} a_n x^n$ and $\sum_{n=0}^{m_2} b_n x^n$ be two polynomials from $R[x]$. Then the **sum** and the **product** of these polynomials is defined as follows:

$$\sum_{n=0}^{m_1} a_n x^n + \sum_{n=0}^{m_2} b_n x^n = \sum_{n=0}^{\max(\{m_1, m_2\})} (a_n + b_n) x^n \quad (3.26)$$

$$\left(\sum_{n=0}^{m_1} a_n x^n \right) \cdot \left(\sum_{n=0}^{m_2} b_n x^n \right) = \sum_{n=0}^{m_1+m_2} \sum_{i=0}^n a_i b_{n-i} x^n \quad (3.27)$$

A rule for polynomial subtraction can be deduced from these two rules by first multiplying the **subtrahend** with (the polynomial) -1 and then add the result to the **minuend**.

Regarding the definition of the degree of a polynomial, we see that the degree of the sum is always the maximum of the degrees of both summands, and the degree of the product is always the degree of the sum of the factors, since we defined $-\infty + m = -\infty$ for every integer $m \in \mathbb{Z}$.

Example 20. To give an example of how polynomial arithmetic works, consider the following two integer polynomials $P, Q \in \mathbb{Z}[x]$ with $P(x) = 5x^2 - 4x + 2$ and $Q(x) = x^3 - 2x^2 + 5$. The sum of these two polynomials is computed by adding the coefficients of each term with equal exponent in x . This gives the following:

$$\begin{aligned} (P + Q)(x) &= (0 + 1)x^3 + (5 - 2)x^2 + (-4 + 0)x + (2 + 5) \\ &= x^3 + 3x^2 - 4x + 7 \end{aligned}$$

The product of these two polynomials is computed by multiplication of each term in the first factor with each term in the second factor. We get the following:

$$\begin{aligned} (P \cdot Q)(x) &= (5x^2 - 4x + 2) \cdot (x^3 - 2x^2 + 5) \\ &= (5x^5 - 10x^4 + 25x^2) + (-4x^4 + 8x^3 - 20x) + (2x^3 - 4x^2 + 10) \\ &= 5x^5 - 14x^4 + 10x^3 + 21x^2 - 20x + 10 \end{aligned}$$

```

sage: Zx = ZZ['x']
sage: P = Zx([2, -4, 5])
sage: Q = Zx([5, 0, -2, 1])
sage: P+Q == Zx(x^3 +3*x^2 -4*x +7)
True
sage: P*Q == Zx(5*x^5 -14*x^4 +10*x^3+21*x^2-20*x +10)
True

```

Example 21. Let us consider the polynomials of the previous example but interpreted in modular 6 arithmetic. So we consider $P, Q \in \mathbb{Z}_6[x]$ again with $P(x) = 5x^2 - 4x + 2$ and $Q(x) = x^3 - 2x^2 +$

5. This time we get the following:

$$\begin{aligned}(P + Q)(x) &= (0 + 1)x^3 + (5 - 2)x^2 + (-4 + 0)x + (2 + 5) \\ &= (0 + 1)x^3 + (5 + 4)x^2 + (2 + 0)x + (2 + 5) \\ &= x^3 + 3x^2 + 2x + 1\end{aligned}$$

$$\begin{aligned}(P \cdot Q)(x) &= (5x^2 - 4x + 2) \cdot (x^3 - 2x^2 + 5) \\ &= (5x^2 + 2x + 2) \cdot (x^3 + 4x^2 + 5) \\ &= (5x^5 + 2x^4 + 1x^2) + (2x^4 + 2x^3 + 4x) + (2x^3 + 2x^2 + 4) \\ &= 5x^5 + 4x^4 + 4x^3 + 3x^2 + 4x + 4\end{aligned}$$

1495

1496	<code>sage: Z6x = Integers(6) ['x']</code>	126
1497	<code>sage: P = Z6x([2, -4, 5])</code>	127
1498	<code>sage: Q = Z6x([5, 0, -2, 1])</code>	128
1499	<code>sage: P+Q == Z6x(x^3 +3*x^2 +2*x +1)</code>	129
1500	<code>True</code>	130
1501	<code>sage: P*Q == Z6x(5*x^5 +4*x^4 +4*x^3+3*x^2+4*x +4)</code>	131
1502	<code>True</code>	132

1503 *Exercise 25.* Compare the sum $P + Q$ and the product $P \cdot Q$ from the previous two examples
 1504 20 and 21 and consider the definition of \mathbb{Z}_6 as given in example 9. How can we derive the
 1505 computations in $\mathbb{Z}_6[x]$ from the computations in $\mathbb{Z}[x]$?

1506 **Euklidean Division** The arithmetic of polynomials share a lot of properties with the arith-
 1507 metic of integers and as a consequence the concept of Euclidean division and the algorithm of
 1508 long division is also defined for polynomials. Recalling the Euclidean division of integers 3.2.2,
 1509 we know that, given two integers a and $b \neq 0$, there is always another integer m and a natural
 1510 number r with $r < |b|$ such that $a = m \cdot b + r$ holds.

1511 We can generalize this to polynomials whenever the leading coefficient of the dividend
 1512 polynomial has a notion of multiplicative inverse. In fact, given two polynomials A and $B \neq 0$
 1513 from $R[x]$ such that $Lc(B)^{-1}$ exists in R , there exist two polynomials Q (the quotient) and P (the
 1514 remainder), such that the following equation holds:

$$A = Q \cdot B + P \tag{3.28}$$

1515 and $\deg(P) < \deg(B)$. Similarly to integer Euclidean division, both Q and P are uniquely
 1516 defined by these relations.

1517 *Notation and Symbols 2.* Suppose that the polynomials A, B, Q and P satisfy equation 3.28. We
 1518 often use the following notation to describe the quotient and the remainder polynomials of the
 1519 Euclidean division:

$$A \operatorname{div} B := Q, \quad A \operatorname{mod} B := P \tag{3.29}$$

1520 We also say that a polynomial A is divisible by another polynomial B if $A \operatorname{mod} B = 0$ holds. In
 1521 this case, we also write $B|A$ and call B a *factor* of A .

Algorithm 3 Polynomial Euclidean Algorithm

Require: $A, B \in R[x]$ with $B \neq 0$, such that $Lc(B)^{-1}$ exists in R

procedure POLY-LONG-DIVISION(A, B)

$$Q \leftarrow 0$$
$$P \leftarrow A$$
$$d \leftarrow \deg(B)$$
$$c \leftarrow Lc(B)$$
while $\deg(P) \geq d$ **do**
$$S := Lc(P) \cdot c^{-1} \cdot x^{\deg(P)-d}$$
$$Q \leftarrow Q + S$$
$$P \leftarrow P - S \cdot B$$
end while**return** (Q, P)

end procedure

Ensure: $A = Q \cdot B + P$

Analogously to integers, methods to compute Euclidean division for polynomials are called **polynomial division algorithms**. Probably the best known algorithm is the so-called **polynomial long division**.

This algorithm works only when there is a notion of division by the leading coefficient of B . It can be generalized, but we will only need this somewhat simpler method in what follows.

Example 22 (Polynomial Long Division). To give an example of how the previous algorithm works, let us divide the integer polynomial $A(x) = x^5 + 2x^3 - 9 \in \mathbb{Z}[x]$ by the integer polynomial $B(x) = x^2 + 4x - 1 \in \mathbb{Z}[x]$. Since B is not the zero polynomial and the leading coefficient of B is 1, which is invertible as an integer, we can apply algorithm 1. Our goal is to find solutions to equation XXX, that is, we need to find the quotient polynomial $Q \in \mathbb{Z}[x]$ and the remainder polynomial $P \in \mathbb{Z}[x]$ such that $x^5 + 2x^3 - 9 = Q(x) \cdot (x^2 + 4x - 1) + P(x)$. Using a notation that is mostly used in anglophone countries, we compute as follows:

algorithm-floating

$$X^2 + 4X - 1) \overline{\begin{array}{r} X^3 - 4X^2 + 19X - 80 \\ X^5 + 2X^3 - 9 \\ -X^5 - 4X^4 + X^3 \\ \hline -4X^4 + 3X^3 \\ 4X^4 + 16X^3 - 4X^2 \\ \hline 19X^3 - 4X^2 \\ -19X^3 - 76X^2 + 19X \\ \hline -80X^2 + 19X - 9 \\ 80X^2 + 320X - 80 \\ \hline 339X - 89 \end{array}} \quad (3.30)$$

We therefore get $Q(x) = x^3 - 4x^2 + 19x - 80$ as well as $P(x) = 339x - 89$ and indeed we have $x^5 + 2x^3 - 9 = (x^3 - 4x^2 + 19x - 80) \cdot (x^2 + 4x - 1) + (339x - 89)$, which we can double check invoking Sage:

```
sage: Zx = ZZ['x']
```

```
sage: A = Zx([-9, 0, 0, 2, 0, 1])
```

```
sage: B = Zx([-1, 4, 1])
```



```

1540 sage: Q = Zx([-80, 19, -4, 1]) 136
1541 sage: P = Zx([-89, 339]) 137
1542 sage: A == Q*B + P 138
1543 True 139

```

Example 23. In the previous example, polynomial division gave a non-trivial (non-vanishing, i.e non-zero) remainder. Of special interest are divisions that don't give a remainder. Such divisors are called factors of the dividend.

For example, consider the integer polynomial P_7 from example 16 again. As we have shown, it can be written both as $x^3 - 4x^2 - 11x + 30$ and as $(x - 2)(x + 3)(x - 5)$. From this, we can see that the polynomials $F_1(x) = (x - 2)$, $F_2(x) = (x + 3)$ and $F_3(x) = (x - 5)$ are all factors of $x^3 - 4x^2 - 11x + 30$, since division of P_7 by any of these factors will result in a zero remainder.

Exercise 26. Consider the polynomial expressions $A(x) := -3x^4 + 4x^3 + 2x^2 + 4$ and $B(x) = x^2 - 4x + 2$. Compute the Euclidean division of A by B in the following types:

- 1553 1. $A, B \in \mathbb{Z}[x]$
- 1554 2. $A, B \in \mathbb{Z}_6[x]$
- 1555 3. $A, B \in \mathbb{Z}_5[x]$

Now consider the result in $\mathbb{Z}[x]$ and in $\mathbb{Z}_6[x]$. How can we compute the result in $\mathbb{Z}_6[x]$ from the result in $\mathbb{Z}[x]$?

Exercise 27. Show that the polynomial $B(x) = 2x^4 - 3x + 4 \in \mathbb{Z}_5[x]$ is a factor of the polynomial $A(x) = x^7 + 4x^6 + 4x^5 + x^3 + 2x^2 + 2x + 3 \in \mathbb{Z}_5[x]$ that is show $B|A$. What is $B \text{ div } A$?

Prime Factors Recall that the fundamental theorem of arithmetic 3.7 tells us that every natural number is the product of prime numbers. In this chapter we will see that something similar holds for univariate polynomials $R[x]$, too⁴.

The polynomial analog to a prime number is a so-called an **irreducible polynomial**, which is defined as a polynomial that cannot be factored into the product of two non-constant polynomials using Euclidean division. Irreducible polynomials are for polynomials what prime numbers are for integer: They are the basic building blocks from which all other polynomials can be constructed. To be more precise, let $P \in R[x]$ be any polynomial. Then there are always irreducible polynomials $F_1, F_2, \dots, F_k \in R[x]$, such that the following holds:

$$P = F_1 \cdot F_2 \cdot \dots \cdot F_k . \quad (3.31)$$

This representation is unique, except for permutations in the factors and is called the **prime factorization** of P . Moreover each factor F_i is called a **prime factor** of P .

Example 24. Consider the polynomial expression $P = x^2 - 3$. When we interpret P as an integer polynomial $P \in \mathbb{Z}[x]$, we find that this polynomial is irreducible, since any factorization other than $1 \cdot (x^2 - 3)$, must look like $(x - a)(x + a)$ for some integer a , but there is no integers a with $a^2 = 3$.

```

1575 sage: Zx = ZZ['x'] 140
1576 sage: p = Zx(x^2-3) 141

```

⁴Strictly speaking this is not true for polynomials over arbitrary types R . However in this book we assume R to be a so-called unique factorization domain for which the content of this section holds.

```

1577 sage: p.factor()
1578 x^2 - 3

```

On the other hand interpreting P as a polynomial $P \in \mathbb{Z}_6[x]$ in modulo 6 arithmetic, we see that P has two factors $F_1 = (x - 3)$ and $F_2 = (x + 3)$, since $(x - 3)(x + 3) = x^2 - 3x + 3x - 3 \cdot 3 = x^2 - 3$.

Points where a polynomial evaluates to zero are called **roots** of the polynomial. To be more precise, let $P \in R[x]$ be a polynomial. Then a root is a point $x_0 \in R$ with $P(x_0) = 0$ and the set of all roots of P is defined as follows:

$$R_0(P) := \{x_0 \in R \mid P(x_0) = 0\} \quad (3.32)$$

The roots of a polynomial are of special interest with respect to its prime factorization, since it can be shown that for any given root x_0 of P the polynomial $F(x) = (x - x_0)$ is a prime factor of P .

Finding the roots of a polynomial is sometimes called **solving the polynomial**. It is a hard problem and has been the subject of much research throughout history.

It can be shown that if m is the degree of a polynomial P , then P can not have more than m roots. However, in general, polynomials can have less than m roots.

Example 25. Consider the integer polynomial $P_7(x) = x^3 - 4x^2 - 11x + 30$ from example 16 again. We know that its set of roots is given by $R_0(P_7) = \{-3, 2, 5\}$.

On the other hand, we know from example 24 that the integer polynomial $x^2 - 3$ is irreducible. It follows that it has no roots, since every root defines a prime factor.

Example 26. To give another example, consider the integer polynomial $P = x^7 + 3x^6 + 3x^5 + x^4 - x^3 - 3x^2 - 3x - 1$. We can invoke Sage to compute the roots and prime factors of P :

```

1597 sage: Zx = ZZ['x']
1598 sage: p = Zx(x^7 + 3*x^6 + 3*x^5 + x^4 - x^3 - 3*x^2 - 3*x - 1)
1599
1600 sage: p.roots()
1601 [(1, 1), (-1, 4)]
1602 sage: p.factor()
1603 (x - 1) * (x + 1)^4 * (x^2 + 1)

```

We see that P has the root 1 and that the associated prime factor $(x - 1)$ occurs once in P and that it has the root -1 , where the associated prime factor $(x + 1)$ occurs 4 times in P . This gives the following prime factorization:

$$P = (x - 1)(x + 1)^4(x^2 + 1)$$

Exercise 28. Show that if a polynomial $P \in R[x]$ of degree $\deg(P) = m$ has less than m roots, it must have a prime factor F of degree $\deg(F) > 1$.

Exercise 29. Consider the polynomial $P = x^7 + 3x^6 + 3x^5 + x^4 - x^3 - 3x^2 - 3x - 1 \in \mathbb{Z}_6[x]$. Compute the set of all roots of $R_0(P)$ and then compute the prime factorization of P .

Lagrange interpolation One particularly useful property of polynomials is that a polynomial of degree m is completely determined on $m + 1$ evaluation points, which implies that we can uniquely derive a polynomial of degree m from a set S :

$$S = \{(x_0, y_0), (x_1, y_1), \dots, (x_m, y_m) \mid x_i \neq x_j \text{ for all indices } i \text{ and } j\} \quad (3.33)$$

1611 Polynomials therefore have the property that $m + 1$ pairs of points (x_i, y_i) for $x_i \neq x_j$ are enough
 1612 to determine the set of pairs $(x, P(x))$ for all $x \in R$. This “few too many” property of polynomials
 1613 is used in many places, like for example in erasure codes. It is also of importance in snarks and
 1614 we therefore need to understand a method to actually compute a polynomial from a set of points.
 1615 If the coefficients of the polynomial we want to find have a notion of multiplicative inverse,
 1616 it is always possible to find such a polynomial using a method called **Lagrange interpolation**,
 1617 which works as follows: Given a set like 3.33, a polynomial P of degree m with $P(x_i) = y_i$ for
 1618 all pairs (x_i, y_i) from S is given by the following algorithm:

 check
algorithm
floating

Algorithm 4 Lagrange Interpolation

Require: R must have multiplicative inverses

Require: $S = \{(x_0, y_0), (x_1, y_1), \dots, (x_m, y_m) \mid x_i, y_i \in R, x_i \neq x_j \text{ for all indices } i \text{ and } j\}$
procedure LAGRANGE-INTERPOLATION(S)

 for $j \in (0 \dots m)$ **do**

$$l_j(x) \leftarrow \prod_{i=0; i \neq j}^m \frac{x - x_i}{x_j - x_i} = \frac{(x - x_0)}{(x_j - x_0)} \dots \frac{(x - x_{j-1})}{(x_j - x_{j-1})} \frac{(x - x_{j+1})}{(x_j - x_{j+1})} \dots \frac{(x - x_m)}{(x_j - x_m)}$$

end for

$$P \leftarrow \sum_{j=0}^m y_j \cdot l_j$$

return P
end procedure
Ensure: $P \in R[x]$ with $\deg(P) = m$
Ensure: $P(x_j) = y_j$ for all pairs $(x_j, y_j) \in S$

Example 27. Let us consider the set $S = \{(0, 4), (-2, 1), (2, 3)\}$. Our task is to compute a polynomial of degree 2 in $\mathbb{Q}[x]$ with coefficients from the rational numbers \mathbb{Q} . Since \mathbb{Q} has multiplicative inverses, we can use the Lagrange interpolation algorithm from 4, to compute the polynomial.

$$\begin{aligned} l_0(x) &= \frac{x - x_1}{x_0 - x_1} \cdot \frac{x - x_2}{x_0 - x_2} = \frac{x + 2}{0 + 2} \cdot \frac{x - 2}{0 - 2} = -\frac{(x + 2)(x - 2)}{4} \\ &= -\frac{1}{4}(x^2 - 4) \\ l_1(x) &= \frac{x - x_0}{x_1 - x_0} \cdot \frac{x - x_2}{x_1 - x_2} = \frac{x - 0}{-2 - 0} \cdot \frac{x - 2}{-2 - 2} = \frac{x(x - 2)}{8} \\ &= \frac{1}{8}(x^2 - 2x) \\ l_2(x) &= \frac{x - x_0}{x_2 - x_0} \cdot \frac{x - x_1}{x_2 - x_1} = \frac{x - 0}{2 - 0} \cdot \frac{x + 2}{2 + 2} = \frac{x(x + 2)}{8} \\ &= \frac{1}{8}(x^2 + 2x) \\ P(x) &= 4 \cdot \left(-\frac{1}{4}(x^2 - 4)\right) + 1 \cdot \frac{1}{8}(x^2 - 2x) + 3 \cdot \frac{1}{8}(x^2 + 2x) \\ &= -x^2 + 4 + \frac{1}{8}x^2 - \frac{1}{4}x + \frac{3}{8}x^2 + \frac{3}{4}x \\ &= -\frac{1}{2}x^2 + \frac{1}{2}x + 4 \end{aligned}$$

1619 And, indeed, evaluation of P on the x -values of S gives the correct points, since $P(0) = 4$,
 1620 $P(-2) = 1$ and $P(2) = 3$. Sage provides the following function:

```

1621 sage: Qx = QQ['x'] 150
1622 sage: S=[(0,4),(-2,1),(2,3)] 151
1623 sage: Qx.lagrange_polynomial(S) 152
1624 -1/2*x^2 + 1/2*x + 4 153

```

Example 28. To give another example more relevant to the topics of this book, let us consider the same set $S = \{(0,4), (-2,1), (2,3)\}$ as in the previous example. This time, the task is to compute a polynomial $P \in \mathbb{Z}_5[x]$ from this data. Since we know from example 14 that multiplicative inverses exist in \mathbb{Z}_5 , algorithm 4 applies and we can compute a unique polynomial of degree 2 in $\mathbb{Z}_5[x]$ from S . We can use the lookup tables from example 14 for computation in \mathbb{Z}_5 and get the following:

$$\begin{aligned}
l_0(x) &= \frac{x-x_1}{x_0-x_1} \cdot \frac{x-x_2}{x_0-x_2} = \frac{x+2}{0+2} \cdot \frac{x-2}{0-2} = \frac{(x+2)(x-2)}{-4} = \frac{(x+2)(x+3)}{1} \\
&= x^2 + 1 \\
l_1(x) &= \frac{x-x_0}{x_1-x_0} \cdot \frac{x-x_2}{x_1-x_2} = \frac{x-0}{-2-0} \cdot \frac{x-2}{-2-2} = \frac{x}{3} \cdot \frac{x+3}{1} = 2(x^2 + 3x) \\
&= 2x^2 + x \\
l_2(x) &= \frac{x-x_0}{x_2-x_0} \cdot \frac{x-x_1}{x_2-x_1} = \frac{x-0}{2-0} \cdot \frac{x+2}{2+2} = \frac{x(x+2)}{3} = 2(x^2 + 2x) \\
&= 2x^2 + 4x \\
P(x) &= 4 \cdot (x^2 + 1) + 1 \cdot (2x^2 + x) + 3 \cdot (2x^2 + 4x) \\
&= 4x^2 + 4 + 2x^2 + x + x^2 + 2x \\
&= 2x^2 + 3x + 4
\end{aligned}$$

1625 And, indeed, evaluation of P on the x -values of S gives the correct points, since $P(0) = 4$,
1626 $P(-2) = 1$ and $P(2) = 3$. We can doublecheck our findings using Sage:

```

1627 sage: F5 = GF(5) 154
1628 sage: F5x = F5['x'] 155
1629 sage: S=[(0,4),(-2,1),(2,3)] 156
1630 sage: F5x.lagrange_polynomial(S) 157
1631 2*x^2 + 3*x + 4 158

```

1632 *Exercise 30.* Consider modular 5 arithmetic from example 14 and the set $S = \{(0,0), (1,1), (2,2), (3,2)\}$.
1633 Find a polynomial $P \in \mathbb{Z}_5[x]$ such that $P(x_i) = y_i$ for all $(x_i, y_i) \in S$.

1634 *Exercise 31.* Consider the set S from the previous example. Why is it not possible to apply
1635 algorithm 4 to construct a polynomial $P \in \mathbb{Z}_6[x]$, such that $P(x_i) = y_i$ for all $(x_i, y_i) \in S$?

Bibliography

- Jens Groth. On the size of pairing-based non-interactive arguments. *IACR Cryptol. ePrint Arch.*, 2016:260, 2016. URL <http://eprint.iacr.org/2016/260>.
- P.W. Shor. Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings 35th Annual Symposium on Foundations of Computer Science*, pages 124–134, 1994. doi: 10.1109/SFCS.1994.365700.
- David Fifield. The equivalence of the computational diffie–hellman and discrete logarithm problems in certain groups, 2012. URL <https://web.stanford.edu/class/cs259c/finalpapers/dlp-cdh.pdf>.
- Torben Pryds Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In Joan Feigenbaum, editor, *Advances in Cryptology — CRYPTO '91*, pages 129–140, Berlin, Heidelberg, 1992. Springer Berlin Heidelberg. ISBN 978-3-540-46766-3. URL <https://fmouhart.epheme.re/Crypto-1617/TD08.pdf>.
- Martin Albrecht, Lorenzo Grassi, Christian Rechberger, Arnab Roy, and Tyge Tiessen. Mimc: Efficient encryption and cryptographic hashing with minimal multiplicative complexity. *Cryptology ePrint Archive, Report 2016/492*, 2016. <https://ia.cr/2016/492>.