Notes to a Math Puzzle Book

Martin Titz

February 27, 2025

Contents

1 Introduction

I once have bought the book [2] by J.J.Clessa ”Math and Logic Puzzles for PC Enthusiasts” in the reprinted Dover edition and now found time to solve the majority of the so called micropuzzles. These are my collected notes.

Many puzzles can be best solved writing a computer program. In most cases I used the programming language Java, while I used the computer algebra system Mathematica for micropuzzles 25 and 55 to solve the minimization problem respective the polynomial equation of degree 4.

2 Errata

The following list is only what I have noticed myself, and is no official list.

- Micropuzzle 41: First example: Replace $321 = 19+2.1^2$ with $21 = 19+2\cdot1^2$
- Micropuzzle 55: Replace in drawing '20m' with '25m'
- Solution to micropuzzle 1: Replace 'perimeter of 144' with 'perimeter of 576'
- Solution to micropuzzle 5: Replace $A0 - 2$ with $A0 = 2$
- Solution to micropuzzle 26: Replace 'school' with 'local'
- Solution to micropuzzle 35: Replace $80 + D = 281$ with $80 + D = 28M$
- Solution to micropuzzle 54: Replace equation (1) with $x^2 + y^2 = 225$
- Solution to micropuzzle 54: The square is missing in $(x + y)^2 - 2xy = 225$
- Solution to micropuzzle 54: Rounding error in result, better 3.781555
- Solution to micropuzzle 55: Replace equation (2) with $\frac{1}{a} + \frac{1}{b} = \frac{1}{h}$
- Solution to micropuzzle 55: In the last sentence it should be 'distance between the walls'

3 Overview

Some of the Java programs require additional Java source files, either using Permutation.java for iterating through all permutations of a given size or using some number-theoretical algorithms like fast square detection, using an algorithm given in [3]. It is also mentioned when the internal Java class BigInteger is used.

In many cases the Java program solves a slightly more general problem, either by calculating a solution for a more general case than the special value given in the problem or by not only finding the first solution. In several cases the special puzzle problem is the default and other values can be given by command line parameters.

	Puzzle	**Used**	**Remarks (uses)**
1	Pythagoras for beginners	C++, Java	NumberTheory.java
2	Flying the Glasgow shuttle	—	
3	A chessboard dilemma	Java	
4	A palindromic puzzle	Java	
5	Digital dexterity	Java	BigInteger
6	A palindromic square	Java	
7	More perfect squares	Java	(also older version)
8	A question of logic	(list cases)	
9	The whole truth and nothing but the truth	(list cases)	
10	Talking turkey	Java	
11	The not-so-perfect square	Java	NumberTheory.java
12	Squaring the cube	Java	Permutation.java
13	A natural mistake	Java	
14	Ten-digit perfect squares	Java	

15	A bad connection		
16	A numerical traverse	—	
17	A problem of check-digits	Java	
18	Ceremonial rice pudding	Java	BigInteger
19	Who's who	Java	Permutation.java
20	Word frustration	—	
21	A positional problem	Java	
22	Generating a specific value	Java	Permutation.java
23	Sums of squares	—	
24	A very prime word	—	
25	Fieldcraft	Java, Mathematica	BigDecimal
26	Opening day at the local	(list cases)	
27	A very charitable dilemma	—	
28	Cows, pigs and horses	(paper and pen)	
29	An exact number of factors	(paper and pen)	
30	Cubes and squares	Java	
31	A question of remainders	Java	NumberTheory.java
32	A problem of prime factors	Java	BigInteger
33	The ladies of the committee	Java	
34	An unusual number	Java	
35	Tadpoles, terrapins, tortoises, and turtles	Java	
36	More cubes and squares	C++, Java	NumberTheory.java
37	Sums of cubes	Java	NumberTheory.java
38	Coconut galore		
39	More trouble with remainders	C++, Java	(as testcase)
40	Ball-bearing pyramids		
41	Sums of primes, etc.	Java	NumberTheory.java
42	The numerate marathon runner	Java	
43	Ten-digit primes	Java	NumberTheory.java
44	Approximations	—	
45	Palindromic cycles	Java	BigInteger
46	Mother and daughter		
47	A catastrophic puzzle	(Venn diagram)	
48	Susan's perfect man		
49	An interesting pair of series	Java	
50	Another pyramid problem	Java	
51	A number and its square	Java	Permutation.java
52	A long-winded fraction	—	
53	An infernal triangle	—	
54	The ladder and the wall	Java	
55	More ladders and walls	Mathematica	
56	Reflections at a corner	(paper and pen)	
57	The census taker	Java	NumberTheory.java
58	A number crossword	—	
59	Another number crossword	—	
60	A series of primes	Java	
61	The hymn board	Java	
62	Three unusual digits	Java	
63	A recurring quotient	Java	
64	A rotating grid	Java	Permutation.java
65	The cocktail party	(paper and pen)	

4 Individual Puzzles

4.1 Quickie 16

From the book [2]: "Here's a simple multiplication problem in which each letter represents a different digit. Can you solve it?"

$$\begin{array}{r} \text{IF}\times \\ \text{AT} \\ \hline \text{FIAT} \\ \hline \end{array}$$

The book gives only one solution, but as also 0 is a digit, I found three:

$$\begin{aligned} 21 \times 60 &= 1260 \\ 41 \times 35 &= 1435 \\ 51 \times 30 &= 1530 \end{aligned}$$

4.2 Quickie 17

See also [4], pages 83-84 for a more general treatment. There the general formula for the total number of triangles is given when the big triangle sides each consist of n small triangles (sequence 0, 1, 5, 13, 27, 48, 78, 118, 170,..., M3827 in [15]).

$$\begin{array}{rl} \dfrac{n(n+2)(2n+1)}{8} & \text{for even n} \\ \dfrac{n(n+2)(2n+1)-1}{8} & \text{for odd n} \end{array}$$

It is derived in [4] from a difference pattern, but not proofed for general n.

4.3 Quickie 59

From the book [2]: "Find four consecutive prime numbers that add to 220."

It is $220/4 = 55$ and 55 is not itself a prime number as $55 = 5 \times 11$.
If you add up 4 numbers each less than 55 you get a sum below $4 \times 55 = 220$.
If you add up 4 numbers each greater than 55 you get a sum greater than 220.
Therefore at least one prime must be below 55 and one prime above 55 and as we look for consecutive primes both the primes 53 and 59 must be included. As $220 - 53 - 59 = 108 = 2 \times 54$ one remaining prime must be just below the already found ones and one just above, therefore the unique solution is

$$220 = 47 + 53 + 59 + 61$$

4.4 Micropuzzle 1 – Pythagoras for beginners

From the book [2]: "I'd like you to find the smallest-area Pythagorean triangle whose perimeter is a perfect square and whose area is a perfect cube."

Pythagorean triples are triples (a, b, c) of positive integers with the additional property $a^2 + b^2 = c^2$ like for example $3^2 + 4^2 = 5^2$ or $5^2 + 12^2 = 13^2$.

Generation of Pythagorean triples

Pythagorean triples and their construction are explained in many texts. To give just two examples in well-known books see [16] in section 2.3.4 on page 90 for the generation rule and a short table or [11], section 13.2 with a proof for the generation rule.

Solution of the puzzle

My program Micropuzzle01.java generates all Pythagorean triples up to a given limit for the hypotenuse c and first checks by a fast square detection test implemented in my NumberTheory.java file according to [3], whether the perimeter is a perfect square. If this is true, the program finally checks if the area is a perfect cube and in this case prints the found solution out.

All solutions with hypotenuse $c \leq 1\,000\,000\,000$ ordered by increasing c are

a	b	c	perimeter	area
144	192	240	576	13824
150	360	390	900	27000
9216	12288	15360	36864	56623104
9600	23040	24960	57600	110592000
12960	57600	59040	129600	373248000
113400	119070	164430	396900	6751269000
104976	139968	174960	419904	7346640384
20808	176256	177480	374544	1833767424
103680	194400	220320	518400	10077696000
109350	262440	284310	656100	14348907000
113256	301158	321750	736164	17053975224
127008	435456	453600	1016064	27653197824
37026	610368	611490	1258884	11299742784
158400	784080	799920	1742400	62099136000
589824	786432	983040	2359296	231928233984
191646	1238328	1253070	2683044	118660303944
345744	1361367	1404585	3111696	235342236024
208568	1504926	1519310	3232804	156939702984
614400	1474560	1597440	3686400	452984832000
259920	2462400	2476080	5198400	320013504000
2250000	3000000	3750000	9000000	3375000000000
829440	3686400	3778560	8294400	1528823808000
540225	4033680	4069695	8643600	1089547389000
2205000	3543750	4173750	9922500	3906984375000
363776	5086368	5099360	10549504	925149302784
2343750	5625000	6093750	14062500	6591796875000
2520000	7350000	7770000	17640000	9261000000000
468198	8655336	8667990	17791524	2026205502264
7257600	7620480	10523520	25401600	27653197824000
6981824	8082882	10680770	25745476	28216629768384
6885000	8323200	10801800	26010000	28652616000000
6718464	8957952	11197440	26873856	30091839012864
1331712	11280384	11358720	23970816	7511111368704
6619392	9738144	11774880	28132416	32230296244224
6586272	11176704	12972960	30735936	36806406303744
6635520	12441600	14100480	33177600	41278242816000
6742400	13978440	15519560	36240400	47124116928000

6808050	14760000	16254450	37822500	50243409000000
6998400	16796160	18195840	41990400	58773123072000
677600	18627840	18640160	37945600	6311112192000
3630000	19800000	20130000	43560000	35937000000000
7248384	19274112	20592000	47114496	69853082517504
7719624	23836032	25054920	56610576	92002602345984
16941456	22588608	28235760	67765824	191341954266624
4228200	28383750	28696950	61308900	60006085875000
8128512	27869184	29030400	65028096	113267498287104
8610560	32810400	33921440	75342400	141257958912000
2369664	39063552	39135360	80568576	46283746443264
17114328	36605646	40408830	94128804	313240516147944
26730000	30628125	40651875	98010000	409344890625000
9447840	41990400	43040160	94478400	198359290368000
17647350	42353640	45883110	105884100	373714754427000
10137600	50181120	51194880	111513600	254358061056000
25666875	52650000	58573125	136890000	675680484375000
37748736	50331648	62914560	150994944	949978046398464
6063750	63525000	63813750	133402500	192599859375000
11311650	65499840	66469410	143280900	370455632568000
1306910	71220600	71232590	143760100	46539457173000
12265344	79252992	80196480	171714816	486032604954624
27915408	76097375	81056033	185068816	1062144635427000
22127616	87127488	89893440	199148544	963961798754304
7300800	94770000	95050800	197121600	345948408000000
29600000	92407500	97032500	219040000	1367631000000000
13348352	96315264	97235840	206899456	642825023422464
39321600	94371840	102236160	235929600	1855425871872000
82668600	86802030	119869470	289340100	3587901148629000
76527504	102036672	127545840	306110016	3904305912313344
15169032	128490624	129382920	273042576	974539193577984
4461600	144967680	145036320	294465600	323393900544000
16634880	157593600	158469120	332697600	1310775312384000
75582720	141717600	160613280	377913600	5355700839936000
39361977	199994000	203830727	443186704	3936079614069000
79716150	191318760	207261990	478296900	7625597484987000
153307200	166511250	226338450	546156900	12763686753000000
152100000	168480000	226980000	547560000	12812904000000000
82563624	219544182	234555750	536663556	9063181647017784
144000000	192000000	240000000	576000000	13824000000000000
53084160	235929600	241827840	530841600	6262062317568000
43512500	254880000	258567500	556960000	5545233000000000
34574400	258155520	260460480	553190400	4462786105344000
21060000	262828800	263671200	547560000	2767587264000000
141120000	226800000	267120000	635040000	16003008000000000
205760898	222659136	303174270	731594304	22907271885632064
142876800	277365000	312001800	732243600	19814511816000000
143325000	283920000	318045000	745290000	20346417000000000
23281664	325527552	326359040	675168256	3789411544203264
92588832	317447424	330674400	740710656	14696043104784384
13525200	344760000	345025200	703310400	2331473976000000
189078750	311169600	364111650	864360000	29417779503000000
150000000	360000000	390000000	900000000	27000000000000000
255104784	340139712	425174640	1020419136	43385633879791104

55080000	438918750	442361250	936360000	12087822375000000
26991954	444958272	445776210	917726436	6005146604871744
161280000	470400000	497280000	1128960000	37933056000000000
16645128	528351750	528613878	1073610756	4397241253887000
29964672	553941504	554751360	1138657536	8299337737273344
8653320	558105600	558172680	1124931600	2414733175296000
115473600	571594320	583141680	1270209600	33002026934976000
177619200	631620000	656119200	1465358400	56093919552000000
464486400	487710720	673505280	1625702400	113267498287104000
179681250	652680000	676961250	1509322500	58637179125000000
446836736	517304448	683569280	1647710464	115575315531300864
54038376	687929718	690048870	1432016964	18587302381428984
265734150	637761960	690908790	1594404900	84737566171467000
440640000	532684800	691315200	1664640000	117361115136000000
429981696	573308928	716636160	1719926784	123256172596690944
85229568	721944576	726958080	1534132224	30765512166211584
423641088	623241216	753592320	1800474624	132015293416341504
422132256	648747008	773995040	1844874304	136928519030145024
421521408	715309056	830269440	1967099904	150759040220135424
60577230	877325400	879414270	1817316900	26572971270321000
199764288	868877250	891545538	1960187076	86785322602824000
424673280	796262400	902430720	2123366400	169075682574336000
139709934	902741112	913488030	1955939076	63060950588303304
202500000	900000000	922500000	2025000000	91125000000000000
38896200	952560000	953353800	1944810000	18525482136000000
296919480	942148350	987828270	2226896100	139871099082429000
431513600	894620160	993251840	2319385600	193020382937088000

4.5 Micropuzzle 5 – Digital dexterity

From the book [2]: "A certain number ends in the digit 'a'. When the 'a' is taken from the end of the number and placed at the beginning, a new number is formed which is 'a' times the original number."

The problem can be solved more general for an arbitrary base B for the number system instead of only base 10.

A positive integer N can be represented in a number system with base $B \geq 2$ with n digits $0 \leq d_i \leq B-1$ for $i = 0, \ldots, n-1$

$$N = \sum_{i=0}^{n-1} d_i B^i \tag{1}$$

Let $d_0 \in \{2, 3, \cdots, B-1\}$. The case of $d_0 = 1$ is trivial, already $N = 1$ would be a solution (here d_0 is the same as a above in formulation of the problem).

Taking away the last digit d_0 of the number N can be expressed as first subtracting d_0 from N and as then the number ends with a 0 in base B, dividing by the base. Adding the digit d_0 then in front is just adding $d_0 B^{n-1}$ to the remaining number. As by condition of the puzzle this has to be equal d_0 times the original number N, the following equations results

$$d_0 \cdot N = d_0 B^{n-1} + \frac{N - d_0}{B} \tag{2}$$

Multiplying both sides with B

$$Bd_0N = d_0B^n + N - d_0$$

and sorting all terms with N to the left side and the other ones to the right side yields

$$(Bd_0 - 1)\, N = d_0\, (B^n - 1)$$

As d_0 is an integer ≥ 2, the two integers $Bd_0 - 1$ and d_0 cannot have a common divisor, therefore $Bd_0 - 1$ must be a divisor of the other factor on the right side, therefore it must be

$$(Bd_0 - 1) \,|\, (B^n - 1) \tag{3}$$

and if this condition is fulfilled for a given n then the original number N can be calculated by

$$N = \frac{d_0 \cdot (B^n - 1)}{Bd_0 - 1} \tag{4}$$

To find the smallest n which fulfills condition (3) you need to find the smallest positive integer n that

$$B^n \equiv 1 \pmod{Bd_0 - 1} \tag{5}$$

This is just the multiplicative order of an integer a modulo m with symbol $\mathrm{ord}_m(a)$, in our case it is $a = B$ and $m = Bd_0 - 1$, see for example [11], section 6.8 or [9], section 6.2.

With this, the solution of the puzzle can be done in the following two steps:

1. Calculate $n = \mathrm{ord}_{Bd_0-1}(B)$. A naive implementation is sufficient, as B is the number system base, $d_0 < B$ and thus $Bd_0 - 1$ is small in this case. For a better algorithm see [3], algorithm 1.4.3.
2. Calculate N by the division (4), using high precision integer arithmetic as the powers B^n will soon become huge numbers well out of the range of normal integer arithmetic with int or long data types.

Example 1: $B = 10$, $d_0 = 4$: Here is $Bd_0 - 1 = 39$ and as the smallest exponent to be found is $n = 6$ for which $39|\,(10^n - 1)$, hence $\mathrm{ord}_{39}(10) = 6$ and

$$N = \frac{4 \cdot \left(10^6 - 1\right)}{39} = \frac{4 \cdot 999999}{39} = 102564$$

Indeed it is

$$4 \cdot 102564 = 410256$$

Example 2: $B = 4, d_0 = 2$: Here is $Bd_0 - 1 = 7$ and $7|\left(4^3 - 1\right)$, hence

$$N = \frac{2 \cdot \left(4^3 - 1\right)}{7} = \frac{2 \cdot 63}{7} = 18$$

Therefore

$$2 \cdot (102)_4 = (210)_4$$

For verification it is

$$\begin{aligned} (102)_4 &= 1 \cdot 4^2 + 0 \cdot 4^1 + 2 \cdot 4^0 = 16 + 2 = 18 \\ (210)_4 &= 2 \cdot 4^2 + 1 \cdot 4^1 + 0 \cdot 4^0 = 32 + 4 = 36 \end{aligned}$$

The Java program Micropuzzle05.java (comments removed) solves the puzzle in this way for a given base $B \geq 2$ of the number system, using the BigInteger arbitrary precision library class:

```
import java.math.BigInteger;
import java.text.NumberFormat;
import java.util.Locale;

public class Micropuzzle05
{
    public static int ord(int a, int m)
    {
        int r = 1;
        long power = a % m;
        while (power != 1) {
            ++r;
            power = (power * a) % m;
        }
        return r;
    }

    public static BigInteger leastSolution(int base, int d0)
    {
        if (d0 < 2 || d0 >= base) {
            System.err.println("Digit d0 must be in the range of 2.." + (base-1));
            return BigInteger.ZERO;
        }
        int m = base * d0 - 1;
        int n = ord(base, m);
        BigInteger bigFactor = BigInteger.valueOf(base).pow(n).subtract(BigInteger.ONE);
        return bigFactor.divide(BigInteger.valueOf(m)).multiply(BigInteger.valueOf(d0));
    }

    public static void main(String[] args)
    {
        int base = 10;
        if (args.length > 0) {
            base = Integer.parseInt(args[0]);
            if (base < 2 || args.length >= 2) {
                System.err.println("Command line error");
                return;
            }
            System.out.println("Results for base " + base);
            System.out.println();
        }
        for (int digit = 2; digit < base; ++digit) {
            BigInteger originalNumber = leastSolution(base, digit);
            System.out.println("digit " + digit + ": " +
                    NumberFormat.getNumberInstance(Locale.US).format(originalNumber));
        }
    }
}
```

- The simple ord method uses a local variable of type long to avoid integer overflow at the multiplication for too large arguments.
- The number format is explicitly using US locale settings to make the result reproducible.

The results for the base $B = 10$ of the usual decimal system are the solution of the original puzzle:

d_0	n	N
2	18	105,263,157,894,736,842
3	28	1,034,482,758,620,689,655,172,413,793
4	6	102,564
5	42	102,040,816,326,530,612,244,897,959,183,673,469,387,755
6	58	1,016,949,152,542,372,881,355,932,203,389,830,508,474,576,271,186,440,677,966
7	22	1,014,492,753,623,188,405,797
8	13	1,012,658,227,848
9	44	10,112,359,550,561,797,752,808,988,764,044,943,820,224,719

The solutions for all digits begin with the digit 1. Is there a reason for it?

Yes, there is:

It is $2 \leq d_0 < B$. If $n = 1$ then $B^1 - 1 < Bd_0 - 1$, therefore (3) cannot hold. This shows that is must be $n \geq 2$ for any solution of the problem.

The equation (4) can now be written in another form

$$\begin{aligned} N &= \frac{d_0 \cdot (B^n - 1)}{Bd_0 - 1} \\ &= B^{n-1} \frac{d_0 \left(B - \frac{1}{B^{n-1}}\right)}{Bd_0 \left(1 - \frac{1}{Bd_0}\right)} \\ &= B^{n-1} \frac{d_0 B \left(1 - \frac{1}{B^n}\right)}{Bd_0 \left(1 - \frac{1}{Bd_0}\right)} \\ &= B^{n-1} \frac{1 - \frac{1}{B^n}}{1 - \frac{1}{Bd_0}} \end{aligned}$$

As $2 \leq d_0 < B$, both d_0 and B are positive numbers, and it is

$$B^n > Bd_0 \Rightarrow \frac{1}{B^n} < \frac{1}{Bd_0} \Rightarrow -\frac{1}{B^n} > -\frac{1}{Bd_0} \Rightarrow 1 - \frac{1}{B^n} > 1 - \frac{1}{Bd_0}$$

This gives the bounds

$$1 - \frac{1}{Bd_0} < 1 - \frac{1}{B^n} < 1$$

and dividing all parts of these inequalities by the positive number $1 - \frac{1}{Bd_0}$ gives

$$1 < \frac{1 - \frac{1}{B^n}}{1 - \frac{1}{Bd_0}} < \frac{1}{1 - \frac{1}{Bd_0}} \tag{6}$$

As $d_0 \geq 2$ and integer $B > d_0$ is must be $B \geq d_0 + 1$ and therefore $B \geq 3$. With this the right side of (6) can further be estimated by

$$\frac{1}{1 - \frac{1}{Bd_0}} < \frac{1}{1 - \frac{1}{3 \cdot 2}} = \frac{1}{\frac{5}{6}} = \frac{6}{5}$$

Conclusion: The number (4) has the form of B^{n-1} multiplied with a factor between 1 and 6/5. This explains why the first digit of the solution N is always a 1 and the number n is the number of digits of N, both of this within the number system with base B.

The equation (4) can be written as

$$N = \frac{d_0 \cdot (B^n - 1)}{Bd_0 - 1} = \frac{d_0 B^n}{Bd_0 - 1} - \frac{d_0}{Bd_0 - 1} \tag{7}$$

and because of $d_0 \geq 2$ and $B \geq 3$ it is $Bd_0 - 1 = d_0 + (B-1)d_0 - 1 > d_0$ and so

$$0 < \frac{d_0}{Bd_0 - 1} < 1 \tag{8}$$

Two often useful functions are the floor and ceiling functions which are defined for all real values x as

$$\begin{aligned} \lfloor x \rfloor &= \text{the greatest integer less than or equal to } x \\ \lceil x \rceil &= \text{the least integer greater than or equal to } x \end{aligned}$$

so it is for example $\lfloor 3.14 \rfloor = 3$, $\lceil 3.14 \rceil = 4$, but $\lfloor -3.14 \rfloor = -4$, $\lceil -3.14 \rceil = -3$, see for this and many more details the book [10].

Note that for positive integers a and b the calculation of $\left\lfloor \frac{a}{b} \right\rfloor$ is just the integer quotient of a by b with discarding of any possible remainder. This is just what the division operator / is doing in programming languages like C, C++ and Java for integer arguments (for positive a and b).

Using the floor function it follows from (7) by the facts that N is an integer and that by (8) the last term in (7) must be between 0 and 1 that

$$N = \left\lfloor \frac{d_0 B^n}{Bd_0 - 1} \right\rfloor \tag{9}$$

This can also written in the form

$$N = \left\lfloor \frac{Bd_0}{Bd_0 - 1} B^{n-1} \right\rfloor = \left\lfloor \left(1 + \frac{1}{Bd_0 - 1} \right) B^{n-1} \right\rfloor$$

Using example 1 from above with $B = 10$, $d_0 = 4$ and $n = \mathrm{ord}_{Bd_0 - 1}(B) = \mathrm{ord}_{39}(10) = 6$ it is then

$$N = \left\lfloor \frac{40}{39} \cdot 10^5 \right\rfloor$$

In this case the calculation can be done by performing the division $40/39 = 1.025641025641025641\ldots$ to a sufficient number of decimal digits, then shifting the decimal point five places to the right aka multiplying with 10^5, and finally discarding the digits after the decimal point, resulting in $N = 102564$.

If for the calculation of (9) a number system is internally used with a base of B^2 or B^3 the number $d_0 B^n$ can easily set in the program and as then $Bd_0 - 1$ is lower than the base, the value of N can be calculated with a short division algorithm, see for example [14], page 59 or [12], section 4.3.1, exercise 16.

The following C program uses a base of B^3 for the calculation of (9) as then the output formatting in groups of three digits is very easy. For brevity of the program the divide function overwrites its input array a and returns the result in the same array.

It solves the original puzzle problem for the base 10 of our common decimal system, but cannot be used for other bases. Its advantage is that it does not depend on any external arbitrary precision library.

The C program micropuzzle5.c (only comments removed):

```
#include <stdio.h>
#include <stdlib.h>

int ord(int a, int m)
{
    int r = 1;
    int power = a % m;
    while (power != 1) {
        ++r;
        power = (power * a) % m;
    }
    return r;
}

void divide(int base, int a_high, int a[], int b)
{
    int hi = 0;
    for (int i = a_high; i >= 0; --i) {
        div_t q = div(hi * base + a[i], b);
        a[i] = q.quot;
        hi = q.rem;
    }
}

void print_least_solution(int base, int d0)
{
    int m = base * d0 - 1;
    int n = ord(base, m);
    int multiplier[3] = {1, base, base*base};
    int B = base * base * base;
    div_t digits = div(n, 3);
    int a_high = digits.quot;
    int a[a_high+1];
    for (int i = 0; i < a_high; ++i) {
        a[i] = 0;
    }
    a[a_high] = d0 * multiplier[digits.rem];
    divide(B, a_high, a, m);
    if (a[a_high] == 0)
        --a_high;
    printf("digit %d: ", d0);
    for (int i = a_high; i >= 0; --i) {
        printf(i == a_high ? "%d" :",%03d", a[i]);
    }
    printf("\n");
}

int main(void)
{
    int base = 10;
    for (int digit = 2; digit < base; ++digit) {
        print_least_solution(base, digit);
    }
}
```

4.6 Micropuzzle 7 – More perfect squares

From the book [2]: "Find the smallest perfect square that is also the average of two other perfect squares. In other words, find three perfect squares A, B, and C such that

$$B = (A + C)/2. \tag{10}$$

Oh yes, one other stipulation to curtail all the smart-alecs: A, B, and C may not be equal."

Without loss of generality it can be assumed $A < B < C$.

As the numbers are perfect squares there exist nonnegative integers a, b and c so that $A = a^2$, $B = b^2$, $C = c^2$, and $0 \leq a < b < c$.

Inserting this into (10) and multiplying both sides with 2 gives the condition

$$2b^2 = a^2 + c^2 \tag{11}$$

A primitive solution is one where the numbers a, b and c do not have a common divisor. The general solutions can be obtained by multiplying all three numbers of a primitive solution by the same positive integer d.

From now on it is looked only for primitive solutions, therefore a, b and c have to be numbers without a common prime factor.

If a and c are both even, a^2 and c^2 are multiples of 4, therefore also their sum and by (11) also $2b^2$ is a multiple of 4, hence b^2 is a multiple of 2 and therefore it is an even number. This is only possible with b itself an even number, thus having a factor of 2. As all three numbers then would have a common factor of 2, this cannot be for a primitive solution.

If one of a and c is even and one odd, it is also one of a^2 and c^2 odd and one even, thus $a^2 + c^2$ is an odd number, which is not possible as the left side of (11) is an even number. Therefore this case cannot happen.

Therefore both a and c must be odd numbers for any primitive solution. As sum and difference of two odd numbers are even numbers, both $c + a$ and $c - a$ are even numbers. Hence the numbers

$$n = \frac{c+a}{2}, \qquad m = \frac{c-a}{2} \tag{12}$$

are both positive integers. It is

$$\begin{aligned}
n^2 + m^2 &= \left(\frac{c+a}{2}\right)^2 + \left(\frac{c-a}{2}\right)^2 \\
&= \frac{1}{4}\left((c+a)^2 + (c-a)^2\right) \\
&= \frac{1}{4}\left(c^2 + 2ac + a^2 + c^2 - 2ac + a^2\right) \\
&= \frac{1}{4}\left(2a^2 + 2c^2\right) \\
&= \frac{1}{2}\left(a^2 + c^2\right) \\
&= b^2
\end{aligned}$$

and so the triple (n, m, b) must be a triple of Pythagorean numbers.

Multiplying both sides of the equations in (12) by 2 gives

$$2n = c + a, \qquad 2m = c - a$$

and

$$a = n - m, \qquad c = n + m \tag{13}$$

This way you can take any primitive Pythagorean triple (m, n, b) with $m < n < b$ and generate from it using (13) a primitive solution triple (a, b, c) for (11).

Example: A well-known Pythagorean triple is $(3, 4, 5)$ because $3^2 + 4^2 = 5^2$. Here it is $m = 3$, $n = 4$ and $b = 5$ and by (13) it follows $a = 4 - 3 = 1$ and $c = 4 + 3 = 7$. Therefore it is $A = 1^2 = 1$, $B = 5^2 = 25$, $C = 7^2 = 49$ a solution of the equation (10).

A table with the first primitive solutions ordered by increasing B:

$A = a^2$	$B = b^2$	$C = c^2$	difference
$1 = 1^2$	$25 = 5^2$	$49 = 7^2$	24
$49 = 7^2$	$169 = 13^2$	$289 = 17^2$	120
$49 = 7^2$	$289 = 17^2$	$529 = 23^2$	240
$289 = 17^2$	$625 = 25^2$	$961 = 31^2$	336
$1 = 1^2$	$841 = 29^2$	$1681 = 41^2$	840
$529 = 23^2$	$1369 = 37^2$	$2209 = 47^2$	840
$961 = 31^2$	$1681 = 41^2$	$2401 = 49^2$	720
$289 = 17^2$	$2809 = 53^2$	$5329 = 73^2$	2520
$2401 = 49^2$	$3721 = 61^2$	$5041 = 71^2$	1320
$2209 = 47^2$	$4225 = 65^2$	$6241 = 79^2$	2016
$529 = 23^2$	$4225 = 65^2$	$7921 = 89^2$	3696
$49 = 7^2$	$5329 = 73^2$	$10609 = 103^2$	5280
$5041 = 71^2$	$7225 = 85^2$	$9409 = 97^2$	2184
$1681 = 41^2$	$7225 = 85^2$	$12769 = 113^2$	5544
$1681 = 41^2$	$7921 = 89^2$	$14161 = 119^2$	6240
$49 = 7^2$	$9409 = 97^2$	$18769 = 137^2$	9360

4.7 Micropuzzle 11 – The not-so-perfect square

From the book [2]: "A certain perfect square has the property that, if 5 is added to it, a second perfect square is obtained, and if 5 is subtracted from it, a third perfect square is obtained.

What is the original perfect square?"

It is $3^2 - 2^2 = 9 - 4 = 5$, but there are no other perfect squares within the integers that have the same difference, therefore there does not exist any solution within the integers.

Let $\left(\frac{a}{d}\right)^2$ be this certain perfect square as a fraction, then we need to find a sequence of three fractions

$$\left(\frac{a}{d}\right)^2 - 5, \quad \left(\frac{a}{d}\right)^2, \quad \left(\frac{a}{d}\right)^2 + 5$$

where also the first and the third number are perfect squares.

Multiplying all numbers by d^2 we get

$$a^2 - 5d^2, \quad a^2, \quad a^2 + 5d^2$$

where $a^2 - 5d^2$ and $a^2 + 5d^2$ must be perfect squares within the integers.

As it has to be $a^2 - 5d^2 > 0$ it must be $d < a/\sqrt{5}$.

4.8 Micropuzzle 13 – A natural mistake

The woman is buying 4 items with prices p_1, p_2, p_3, p_4 that must fulfill the two conditions

$$\begin{aligned} p_1 + p_2 + p_3 + p_4 &= 7.11 \\ p_1 \cdot p_2 \cdot p_3 \cdot p_4 &= 7.11 \end{aligned}$$

It will be easier to transform these equations so further calculations will be done only within integers.

The British currency had many changes over time. The puzzles had been stated after the decimalization of the British coinage 1971, but at the time the puzzle had been first published, the halfpenny had still been legal tender. Therefore the variables a, b, c and d give the prices of the four items in halfpennies; a halfpenny had been at that time 1/200 of a pound.

We have to look for solutions in positive integers of the equations

$$\begin{aligned} \frac{a}{200} + \frac{b}{200} + \frac{c}{200} + \frac{d}{200} &= \frac{711}{100} \\ \frac{a}{200} \cdot \frac{b}{200} \cdot \frac{c}{200} \cdot \frac{d}{200} &= \frac{711}{100} \end{aligned}$$

Multiplying the first equation with 200 and the second one with 200^4 gives

$$\begin{aligned} a + b + c + d &= 1422 \\ a \cdot b \cdot c \cdot d &= 1422 \cdot 200^3 = 11\,376\,000\,000 \end{aligned}$$

Without loss of generality we can assume $a \geq b \geq c \geq d$.

I then got the three possible solutions with a small Java program with three nested loops, as the value of the fourth variable is then determined by the first equation. If you want to try to solve the the problem without programming, you could use that the variables need to be divisors of $1422 \cdot 200^3$.

The solutions are

a	b	c	d
625	316	256	225
625	320	240	237
632	300	250	240

and so the possible prices are in pounds

p_1	p_2	p_3	p_4
3.125	1.580	1.280	1.125
3.125	1.600	1.200	1.185
3.160	1.500	1.250	1.200

The last solution is the only one that does not require usage of halfpenny coinage.

4.9 Micropuzzle 14 – Ten-digit perfect squares

The program Micropuzzle14.java is more general than the exercise, allowing to give the number of digits as command line parameter. As the Java type long is used for the calculation, a maximum of 18 digits is possible.

Nice individual results for square numbers with the most fours within 5 to 8 decimal digits:

$$\begin{aligned}212^2 &= 44944\\ 738^2 &= 544644\\ 2538^2 &= 6441444\\ 6888^2 &= 47444544\end{aligned}$$

In the first result both numbers are additionally palindromic numbers.

The result of my program for the puzzle with 10 digits is

```
Record table for 10 digits, range from 1000000000 to 9999999999

 0   8      40000          1600000000
 1   6      33183          1101111489
 2   6      35415          1254222225
 3   6      57735          3333330225
 4   7      66592          4434494464
 5   6      67495          4555575025
 6   6      68313          4666665969
 7   6      88924          7907477776
 8   6      61878          3828886884
 9   6      53937          2909199969
```

The first column gives the digit that is looked for, the second column how often it occurs in the number in the fourth column, the third the number to be squared to get the number in the fourth column.

4.10 Micropuzzle 25 – Fieldcraft

The fastest time is achieved when walking in straight lines across the different types of soil. Let s_1, s_2, s_3 be the distances on each kind of surface.

Let x_1 the horizontal distance to the point where the farmer leaves from bog to ploughed soil and x_2 the horizontal distance to the point where the farmer passes over from ploughed soil to turf.

Then it is by applying the theorem of Pythagoras ($0 \leq x_1, x_2 \leq 600$)

$$\begin{aligned}s_1(x_1, x_2) &= \sqrt{x_1^2 + 100^2}\\ s_2(x_1, x_2) &= \sqrt{(x_2 - x_1)^2 + 200^2}\\ s_3(x_1, x_2) &= \sqrt{(600 - x_2)^2 + 300^2}\end{aligned}$$

Using the equation $v = s/t$, where v is speed, and t is time, in the form $t = s/v$ the total walking time in seconds becomes with $v_1 = 5/2$, $v_2 = 5$ and $v_3 = 10$

$$t(x_1, x_2) = \sum_{k=1}^{3} \frac{s_k(x_1, x_2)}{v_k}$$

If $x_1 > x_2$ it is $t(x_1, x_2) > t(x_2, x_1)$, as by the middle term $s_2(x_1, x_2)$ does not change its value, and the other two terms $s_1(x_1, x_2)$ and $s_3(x_1, x_2)$ decrease with swapping of the variables. Hence, the minimum for the function $t\,(x_1, x_2)$ will be obtained for $x_1 \le x_2$.

Therefore the function

$$t\,(x_1, x_2) = \frac{2}{5}\sqrt{x_1^2 + 100^2} + \frac{1}{5}\sqrt{(x_2 - x_1)^2 + 200^2} + \frac{1}{10}\sqrt{(600 - x_2)^2 + 300^2}$$

has to be minimized with the constraints $0 \le x_1 \le x_2 \le 600$.

The minimum calculated by the Mathematica function NMinimize is

$$\begin{aligned} x_1 &\approx 21.750059753139185663553649151237504007507 6 \\ x_2 &\approx 115.669703582869959229191842239375878852973 \\ t\,(x_1, x_2) &\approx 142.097659044197040067245550187723934324992 \end{aligned}$$

The shortest time rounded to the nearest second is 2 minutes 22 seconds. The high precision is not needed for the puzzle answer, but could be useful as reference value for checking another implementation of this problem.

Practical solution for the farmer

For any practical purpose it would be sufficient if the farmer uses $x_1 = 21$ feet 9 inches and $x_2 = 115$ feet 8 inches, as then the walking time becomes

$$\begin{aligned} t\left(21\frac{3}{4}, 115\frac{2}{3}\right) &= \frac{1}{60}\left(6\sqrt{167569} + 2\sqrt{2921209} + \sqrt{7030129}\right) \\ &\approx 142.0976590477148358443011714551 67 \end{aligned}$$

and this would take only less than 3.52×10^{-9} seconds or 3.52 ns longer than following the path of the optimal solution.

A way to a very high precision solution

As in the formula for the walking time rather large numbers like 200^2 are used, it makes sense to substitute variables by

$$\begin{aligned} x_1 &= 100u \\ x_2 &= 100v \end{aligned}$$

The function for the walking time then becomes

$$\mathrm{time}(u, v) = 40\sqrt{u^2 + 1} + 20\sqrt{(u - v)^2 + 4} + 10\sqrt{(6 - v)^2 + 9} \tag{14}$$

and has to be minimized with the constraints $0 \le u \le v \le 6$.

A necessary condition for a minimum in the interior of this area is, that both partial derivatives with respect to u and to v become zero. Using $(\sqrt{x})' = \frac{1}{2\sqrt{x}}$ and the chain rule the partial derivatives of (14) are

$$\begin{aligned} \frac{\partial\,\mathrm{time}(u, v)}{\partial u} &= \frac{40u}{\sqrt{u^2 + 1}} + \frac{20(u - v)}{\sqrt{(u - v)^2 + 4}} \\ \frac{\partial\,\mathrm{time}(u, v)}{\partial v} &= -\frac{20(u - v)}{\sqrt{(u - v)^2 + 4}} + \frac{10(v - 6)}{\sqrt{(6 - v)^2 + 9}} \end{aligned}$$

Looking for a point (u, v) where the partial derivatives both become zero, it must be

$$\begin{aligned}
\frac{40u}{\sqrt{u^2+1}} &= -\frac{20(u-v)}{\sqrt{(u-v)^2+4}} \\
\frac{20(u-v)}{\sqrt{(u-v)^2+4}} &= \frac{10(v-6)}{\sqrt{(6-v)^2+9}}
\end{aligned}$$

Multiplying both sides in the equations with the respective denominators and dividing by common numerical factors, the equation system now is

$$\begin{aligned}
2u\sqrt{(u-v)^2+4} &= -(u-v)\sqrt{u^2+1} \\
2(u-v)\sqrt{(6-v)^2+9} &= (v-6)\sqrt{(u-v)^2+4}
\end{aligned}$$

To get rid of the square roots, now the square can be taken from both sides of the equations. This is to equivalence transformation. The new equation system can have additional solutions, so each solution needs to be checked, if it also solves the original equations.

$$4u^2\left((u-v)^2+4\right) = (u-v)^2\left(u^2+1\right) \tag{15}$$

$$4(u-v)^2\left((6-v)^2+9\right) = (v-6)^2\left((u-v)^2+4\right) \tag{16}$$

There exist computational expensive algorithms to simplify this system of polynomial equations, like Buchberger's algorithm to obtain a more separated equation system with the same zeros, a Gröbner basis, see for example the book [5], chapters 2 and 3, the book [7], chapter 21 or the book [8], chapters 9 and 10.

Using a computer algebra system for this step, with polynomials in one variable p and q

$$\begin{aligned}
p(x) = {} & 225x^{12} - 5400x^{11} + 55140x^{10} - 270720x^9 \\
& + 494494x^8 + 205872x^7 - 382388x^6 - 49120x^5 \\
& + 88953x^4 + 3880x^3 - 6440x^2 - 96x + 144
\end{aligned} \tag{17}$$

$$\begin{aligned}
q(x) = {} & \frac{1}{54272}\big(-10575x^{11} + 190350x^{10} - 1067880x^9 - 2841840x^8 \\
& + 53193382x^7 - 148893612x^6 - 44541932x^5 + 122986520x^4 \\
& + 10116609x^3 - 26192402x^2 - 416084x + 1049256\big)
\end{aligned} \tag{18}$$

it is first to solve the polynomial equation (19) of degree 12 for u and then v can be calculated by inserting the value of u into (20)

$$p(u) = 0 \tag{19}$$

$$v = q(u) \tag{20}$$

Instead of having to solve a nonlinear system of equations with two variables, now first one polynomial equation in one variable (19) for u has to be solved and then the v can be calculated by just inserting the value of u into (20) and then the time calculated by inserting the values for u and v into (14).

The polynomial $p(u)$ must have by the fundamental theorem of algebra exactly 12 zeros in the complex plane counted with multiplicity, and these can be determined approximately in increasing order of their real parts as

$$\begin{aligned}
u_{1,2} &\approx 0.5613333 \pm 0.0210778i \\
u_3 &= -0.23757\,81643\,89996\,62923\ldots \\
u_4 &= -0.22245\,18523\,57101\,62446\ldots \\
u_5 &= +0.21750\,05975\,31391\,85663\ldots \\
u_6 &= +0.23492\,11422\,05806\,35141\ldots \\
u_{7,8} &\approx +0.5597088 \pm 0.0302218i \\
u_{9,10} &\approx +5.9979936 \pm 0.7814857i \\
u_{11,12} &\approx +6.0074350 \pm 5.4085462i
\end{aligned}$$

where the real zeros are here given with additional decimal places.

Newton's method

The first order Taylor series of the function $f(x)$ about a point x_0 gives under some general conditions the approximation

$$f(x) \approx f(x_0) + (x - x_0)f'(x_0) \tag{21}$$

around the point x_0.

If we look for a zero $f(x) = 0$ of the function with a guess x_0, we can hope to improve it by setting the left size of (21) to 0

$$0 = f(x_0) + (x_1 - x_0)f'(x_0)$$

and solve this for x_1 assuming that $f'(x_0) \neq 0$ resulting in

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)} \tag{22}$$

As in this case Newton's method has quadratic convergence, it approximately doubles with each further iteration step the number of correct decimal places, and therefore is a very efficient way to get a very high precision solution.

Only the real zeros of (19) are of interest and they can be determined numerically with Newton's method. It requires in this case a relatively good starting value as the real zeros are pairwise relatively close together.

Then for each real zero u_k the walking time $\mathrm{time}\,(u_k, q(u_k))$ can easily be calculated by (20) and (14) with the results shown in the following table.

real zero u_k	$\mathrm{time}\,(u_k, q(u_k))$
$u_3 = -0.23757\,81643\,89996\,62923\ldots$	$164.96444\,02811\,96622\,88394\ldots$
$u_4 = -0.22245\,18523\,57101\,62446\ldots$	$145.92211\,25393\,28821\,96824\ldots$
$u_5 = +0.21750\,05975\,31391\,85663\ldots$	$142.09765\,90441\,97040\,06724\ldots$
$u_6 = +0.23492\,11422\,05806\,35141\ldots$	$160.33649\,04362\,10988\,34849\ldots$

The minimal walking time is achieved by u_5 and it is well within the range of $0 \leq u \leq 6$ required by the original puzzle.

Usually you would still have to check also for a possible minimum on the border of the area of definition. But here you can do another argumentation: Imagine the strips of the field with bog, ploughed soil and turf extending to the left and right without any border. Then you have for time(u, v) only interior minima to consider, and all possible such candidates are the real zeros of $p(u) = 0$. Among them gives u_5 the minimal walking time and is within the ranges of the puzzle condition, therefore this gives the global minimum and the solution for the puzzle.

Result

I wrote a Java program using the BigDecimal class for arbitrary high precision arithmetic and got the result to more than 100 000 decimal places with a calculation time of a few minutes, including finally calculating v by (20), time(u, v) by (14) and multiplying u and v by 100 to obtain the original variables x_1 and x_2.

The final result is to 500 decimal places

```
x1 =
   21.75005 97531 39185 66355 36491 51237 50400 75075 93530 48433
      41700 49199 83229 92656 44805 39007 61663 76044 27562 11123
      39383 76368 64116 91216 73215 15398 16676 47237 91116 58955
      57926 04327 86281 33813 20920 73226 90713 68957 71354 51111
      53014 86340 27995 69087 04313 47053 45008 33031 36689 94026
      52471 21505 22059 06073 03553 87366 74407 96805 04879 83329
      97848 08439 90978 03131 35986 53776 62331 33141 34205 03106
      60386 10564 69071 01302 21666 57421 69559 28798 03277 33633
      74210 26420 29367 13314 82469 94258 19186 78357 58017 97971
      80485 05974 50221 69555 23311 03540 59030 38353 54250 26532 ...
x2 =
  115.66970 35828 69959 22919 18422 39375 87885 29730 45866 43891
      62112 08259 92595 38034 67372 55689 84659 27638 82382 34801
      27967 71901 77992 93680 42998 11751 76836 88723 28960 18608
      55354 50218 69416 34502 70207 36443 73266 33779 47957 45214
      84849 38834 33978 08978 47300 43989 11380 13218 76393 46410
      01408 22038 37122 64787 78970 48939 81521 34765 56282 40538
      54565 39042 30112 19033 12390 46123 54751 48402 28121 46557
      58860 67998 56642 70289 77550 98609 75475 26725 04335 25978
      14340 32246 52151 99508 97125 38029 20720 10741 80936 65927
      08897 45690 09900 05223 47966 83226 69890 21519 65556 29204 ...
t =
  142.09765 90441 97040 06724 55501 87723 93432 49916 85914 27608
      92313 22512 28634 20433 33097 17835 22650 55791 40103 48644
      75376 86717 81174 27522 81863 20963 13865 29763 77359 76649
      66305 80892 99279 30965 29470 63240 99281 56022 38632 39233
      04147 39585 17753 54868 09328 79640 25672 75727 80305 33842
      56597 09419 61550 32206 05677 71393 30500 53848 30799 21215
      83565 19074 37533 26337 62990 16539 87953 26385 58921 20895
      94598 66678 10892 38098 73338 30317 43550 42925 51534 91065
      60229 22520 12642 30173 67373 92190 15445 98688 92776 05845
      21174 02734 21890 29408 10998 14041 23863 11049 86564 31658 ...
```

4.11 Micropuzzle 34 – An unusual number

From the book [2]:

"A fairly straightforward problem, this time. Readily solvable by the chip-based number-cruncher.

I want you to find a six-digit number which, when multiplied by an integer between 2 and 9 inclusive, gives the original six-digit number with its digits reversed.

Thus, if the original number was 123456, and the chosen integer is 8, then 123456×8 should equal 654321, which, of course, it doesn't. However, it is possible to find more than one solution to this problem, but I'll accept any one that meets the required condition."

I found the only two solutions for the original problem with a program as

$$\begin{aligned} 219978 \times 4 &= 879912 \\ 109989 \times 9 &= 989901 \end{aligned}$$

It is interesting to look at the problem in a more general way by looking for n-digit integers instead of only six-digit numbers. Therefore we search for solutions of

$$d \cdot a = p \tag{23}$$

with d an n-digit integer, $a \in \{2, 3, 4, \dots, 9\}$ and p the number d with its digits reversed.

Remark 1: If $a = 1$ it must be $d = p$, therefore exactly all the n-digit palindromic numbers are solutions. This explains why in the puzzle it is $a \geq 2$.

Remark 2: If $n = 1$ and $a \geq 2$ it is $d \cdot a > d$, and as d and p just have one digit, there exists no solution of the puzzle.

Therefore we can now always assume $n \geq 2$, so the numbers d and p will have at least two digits.

Let us write the decimal number d in decimal notation with its n digits $d_0, d_1, \dots, d_{n-1}$ in the form

$$(d_{n-1}, d_{n-2}, \dots, d_2, d_1, d_0)_{10}$$

where the 10 indicates the base of the decimal system. For example the number 2178 would be written in this way as $(2, 1, 7, 8)_{10}$ with $d_0 = 8$, $d_1 = 7$, $d_2 = 1$ and $d_3 = 2$.

It is then

$$d = \sum_{k=0}^{n-1} d_k 10^k, \quad d_{n-1} \neq 0$$

The puzzle equation (23) takes the form

$$(d_{n-1}, d_{n-2}, \dots, d_1, d_0)_{10} \cdot a = (d_0, d_1, \dots, d_{n-2}, d_{n-1})_{10} \tag{24}$$

as p is just d with reversed decimal digits.

By performing the multiplication for the last digit it is

$$d_{n-1} = d_0 \cdot a \mod 10 \tag{25}$$

and therefore the first digit of d is determined by its last digit and the factor a.

Using (25) it is possible to get more information by looking at the first digits of d and p in equation (24):

- As both d and p must be n-digit numbers it must be $d_{n-1} \neq 0$ therefore some combinations of d_0 and a can be excluded, for example $d_0 = 4$ and $a = 5$ with $d_{n-1} = 4 \cdot 5 \mod 10 = 20 \mod 10 = 0$.
- If $d_{n-1} \cdot a \geq d_0 + 1$ the first digit of the product p would become too large, and no equality in (24) possible.
- If $(d_{n-1} + 1) \cdot a < d_0$ the first digit of the product p would be too small, and no equality in (24) possible.

These conditions can easily be checked by the following Java program for all possible combinations of a and d_0. $d_0 = 0$ is not possible as it is the first digit of the product p and this must have the same number of digits as d.

```
public class Micropuzzle34a
{
    public static void main(String[] args)
    {
        final int B = 10;
        for (int a = 2; a < B; ++a) {
            for (int d0 = 1; d0 < B; ++d0) {
                int dHigh = d0 * a % B;
                String classification = "";
                if (dHigh == 0)
                    classification = "First digit zero";
                else if (dHigh * a >= d0 + 1)
                    classification = "Product too large";
                else if ((dHigh + 1) * a < d0)
                    classification = "Product too small";
                System.out.printf("a =%2d, d0 =%2d:  dH =%2d  %s%n", a, d0, dHigh, classification);
            }
        }
    }
}
```

The program excluded all possible combinations with the exception of just three.

The case $a = 2$ and $d_0 = 6$ with $d_{n-1} = 2$ cannot happen, as even if all remaining digits of d would be nines, the product of $d \cdot a$ would still be too small. The two remaining cases are

$$a = 4, \quad d_0 = 8: \qquad d_{n-1} = 2$$

$$a = 9, \quad d_0 = 9: \qquad d_{n-1} = 1$$

As $28 \cdot 4 \neq 82$ and $19 \cdot 9 \neq 91$, we can conclude that there are no solutions for $n = 2$ possible.

For the case of $n = 3$, 3-digit numbers it has to be checked if there are solutions of

$$(2, d_1, 8)_{10} \cdot 4 = (8, d_1, 2)_{10}$$

or

$$(1, d_1, 9)_{10} \cdot 9 = (9, d_1, 1)_{10}$$

possible. In the first case only $208 \cdot 4 = 832$, $218 \cdot 4 = 872$ and $228 \cdot 4 = 912$ has to be calculated, as then the product is already too large. In the second case it is $109 \cdot 9 = 981$ and $119 \cdot 9 = 1071$ is already too large. Therefore there are no solutions for $n = 3$.

Now the same idea as in the previous program can be repeated with varying the factor a and the last two digits d_1 and d_0 instead of only a and d_0. This allows to calculate the last two digits of the product and hence the first two digits of the number. Then the same conditions can be checked, this time with the first two and the last two digits of the numbers d and p. The following Java program performs this task:

```
public class Micropuzzle34b
{
    public static void main(String[] args)
    {
        final int B = 10;
        for (int a = 2; a < B; ++a) {
            for (int d0 = 1; d0 < B; ++d0) {
                int dH0   = (d0 * a) % B;
                int carry = (d0 * a) / B;
                for (int d1 = 0; d1 < B; ++d1) {
                    int dH1 = (d1 * a + carry) % B;
                    int dHigh = B * dH0 + dH1;
                    int dLow = B * d0 + d1;
                    String classification = "";
                    if (dH0 == 0)
                        classification = "First digit zero";
                    else if (dHigh * a >= dLow + 1)
                        classification = "Product too large";
                    else if ((dHigh + 1) * a < dLow)
                        classification = "Product too small";
                    System.out.printf("a = %d, dLow = %02d:  dHigh = %02d  %s%n",
                           a, dLow, dHigh, classification);
                }
            }
        }
    }
}
```

The result is that all possible solutions must be within one of the two cases

$$a = 4, \quad d = (2, 1, \ldots, 7, 8)_{10}$$

$$a = 9, \quad d = (1, 0, \ldots, 8, 9)_{10}$$

The following program searches therefore for solutions only with these two cases varying the remaining $n - 4$ digits in the middle of the number d with $n \geq 4$ overall decimal digits.

```
public class Micropuzzle34c
{
    public static long power(long base, int exponent)
    {
        long result = 1;
        while (exponent > 0) {
            if ((exponent & 1) != 0)
                result *= base;
            exponent >>= 1;
            base *= base;
        }
        return result;
    }
```

```
    public static long number(final int[] digits)
    {
        final int base = 10;
        if (digits == null)
            return 0;
        long n = 0;
        for (int i = digits.length-1; i >= 0; --i) {
            n *= base;
            n += digits[i];
        }
        return n;
    }

    public static void search(final int nDigits)
    {
        if (nDigits <= 3)
             return;
        search(nDigits, 4, 21, 78);
        search(nDigits, 9, 10, 89);
    }

    private static void search(final int nDigits, long a, long dHigh, long dLow)
    {
        final long nHigh = power(10, nDigits-4);
        int[] digits = new int[nDigits];
        for (long dMiddle = 0; dMiddle < nHigh; ++dMiddle) {
            long d = 100 * (nHigh * dHigh + dMiddle) + dLow;
            long p = d;
            for (int i = 0; i < nDigits; ++i) {
                digits[nDigits-1-i] = (int)(p % 10);
                p /= 10;
            }
            p = number(digits);
            if (d * a == p) {
                System.out.println("Solution: " + d + " * " + a + " = " + p);
            }
        }
    }

    public static void main(String[] args)
    {
        if (args.length == 0) {
            search(6);  // the original problem as stated in the book
        } else {
            for (int i = 0; i < args.length; ++i) {
                int nDigits = Integer.parseInt(args[i]);
                if (nDigits >= 2 && nDigits <= 18) {
                    search(nDigits);
                }
            }
        }
    }
}
```

The first ones of these solutions are

$$\begin{aligned}
2178 \times 4 &= 8712 \\
1089 \times 9 &= 9801 \\
21978 \times 4 &= 87912 \\
10989 \times 9 &= 98901 \\
219978 \times 4 &= 879912 \\
109989 \times 9 &= 989901 \\
2199978 \times 4 &= 8799912 \\
1099989 \times 9 &= 9899901 \\
21782178 \times 4 &= 87128712 \\
21999978 \times 4 &= 87999912 \\
10891089 \times 9 &= 98019801 \\
10999989 \times 9 &= 98999901 \\
217802178 \times 4 &= 871208712 \\
219999978 \times 4 &= 879999912 \\
108901089 \times 9 &= 980109801 \\
109999989 \times 9 &= 989999901
\end{aligned}$$

4.12 Micropuzzle 35 – Tadpoles, terrapins, tortoises, and turtles

To only use integers the prices are in pence for the calculation. Then using a, b, c and d for the numbers of each kind these two equations arise:

$$\begin{aligned}
59a + 199b + 287c + 344d &= 10000 &(26)\\
a + b + c + d &= 100 &(27)
\end{aligned}$$

Multiplying (27) with 59 and then subtracting this from the equation (26) results in

$$(59 - 59)a + (199 - 59)b + (287 - 59)c + (344 - 59)d = 10000 - 59 \cdot 100$$

and simplified

$$140b + 228c + 285d = 4100 \tag{28}$$

The simple solution program Micropuzzle35.java varies the values of b and c in the expression given by the left side of (28) in the appropriate range. If then d would be an integer, d is calculated from (28) and finally $a = 100 - b - c - d$ can be calculated and a solution is found.

Also the solution can be found without writing a computer program. One good systematic approach for such problems is demonstrated in the solution of the book [2].

Another way to solve it is the following:

Looking into equation (28) you can see that all the numbers 140, 228 and 4100 are divisible by 4, thus are also $140b$ and $228c$ divisible by 4. Therefore $285d$ must be divisible by 4 too, and therefore $4|d$, so it exists a nonnegative integer d' with

$$d = 4d'$$

and putting this into (28) it becomes

$$140b + 228c + 285 \cdot 4d' = 4100$$

and dividing both sides by 4 we get

$$35b + 57c + 285d' = 1025 \tag{29}$$

As now the numbers 35, 285 and 1025 all contain the common factor of 5, also $57c$ most be a multiple of 5, and therefore it exists a nonnegative integer c' so that

$$c = 5c'$$

and the equation (29) becomes

$$35b + 57 \cdot 5c' + 285d' = 1025$$

and dividing both sides by 5 it simplifies to

$$7b + 57c' + 57d' = 205 \tag{30}$$

As it is $7b = 205 - 57(c' + d')$ we can now check

$$\begin{array}{rcl} 205 - 0 \cdot 57 & = & 7 \cdot 29 + 2 \\ 205 - 1 \cdot 57 & = & 7 \cdot 21 + 1 \\ 205 - 2 \cdot 57 & = & 7 \cdot 13 \\ 205 - 3 \cdot 57 & = & 7 \cdot 4 + 6 \\ 205 - 4 \cdot 57 & < & 0 \end{array}$$

Therefore it must be for any possible solution

$$b = 13$$

and by equation (30) it follows

$$c' + d' = 2.$$

So the cases $c' = 0, d' = 2$, $c' = 1, d' = 1$ and $c' = 2, d' = 0$ have to be checked, and each gives indeed a solution of the problem, with only the middle one having all four animals present.

Therefore we have these three solutions of the problem

a	b	c	d
79	13	0	8
78	13	5	4
77	13	10	0

4.13 Micropuzzle 42 – The numerate marathon runner

From the book [2]: “Runners in a marathon race are assigned consecutive numbers starting at 1.

One of the entrants with a mathematical bent noticed that the sum of the numbers less than his own number was equal to the sum of the numbers greater.

If there were more than 100 runners but less than 1000, what number was he and how many runners were there in the race?"

A preliminary remark: There is a well-known formula for the sum of the first n positive integers (see for example [16] on page 21)

$$1 + 2 + 3 + \ldots + (n-1) + n = \frac{n(n+1)}{2} \tag{31}$$

For example it is in case of $n = 5$

$$1 + 2 + 3 + 4 + 5 = \frac{5 \cdot (5+1)}{2} = \frac{30}{2} = 15$$

Let us assume that there were altogether n runners in the race and the one mathematical interested runner had himself the number m, then m and n are positive integers with $1 \leq m \leq n$.

Then we have

- runners with numbers $1, 2, 3, \ldots, (m-2), (m-1)$ before our runner
- our mathematical runner itself with number m
- runners with numbers $(m+1), (m+2), \ldots, (n-1), n$ after our runner

We get the sum of numbers less than his own number by putting $m-1$ for n into the formula (31):

$$\text{sum before} = \frac{(m-1)m}{2} \tag{32}$$

We get the sum of numbers greater than our runner by subtracting from the total sum of numbers $1, 2, \ldots, n$ the sum of the numbers $1, 2, \ldots, m$ up to and including our runner:

$$\text{sum after} = \frac{n(n+1)}{2} - \frac{m(m+1)}{2} \tag{33}$$

As by the problem the sums before our runner (32) and after our runner (33) must be equal, we get the equation

$$\frac{(m-1)m}{2} = \frac{n(n+1)}{2} - \frac{m(m+1)}{2}$$

Adding $\frac{m(m+1)}{2}$ to both sides gives

$$\frac{(m-1)m + (m+1)m}{2} = \frac{n(n+1)}{2}$$

and simplifies the equation to

$$m^2 = \frac{n(n+1)}{2} \tag{34}$$

This already allows a solution of the given puzzle by a small program in the spirit of the book. We can try out all possible values of n, then calculate $n(n+1)/2$, which is always an integer, as either n or $n+1$ must be an even number. This must be equal to a square number, so we calculate the square root and then round its value and check if this potential value of m multiplied itself becomes $n(n+1)/2$.

```
public class Micropuzzle42
{
    public static void main(String[] args)
    {
        for (long n = 101; n < 1000; ++n) {
            long potentialSquare = n * (n+1) / 2;
            long m = Math.round(Math.sqrt(potentialSquare));
            if (m * m == potentialSquare) {
                System.out.println("Runner " + m + " out of total " + n + " runners.");
            }
        }
    }
}
```

This gives the solution for the problem with runner 204 out of total 288 runners. You can also run the program for all n from for example 1 to 100000 by adjusting the values in the *for* statement. But it does not allow a more general statement about any solutions of (34) within the positive integers.

The numerator of the right side of (34) is $n(n+1) = n^2 + n$. By trying to simplify this you could begin with

$$\left(n + \frac{1}{2}\right)^2 = n^2 + n + \frac{1}{4}$$

and

$$(2n+1)^2 = 4n^2 + 4n + 1 \tag{35}$$

Then you can go on multiplying both sides of equation (34) with 8 to obtain

$$8m^2 = 4n(n+1)$$

and further using (35)

$$2 \cdot (2m)^2 = (2n+1)^2 - 1$$

and

$$(2n+1)^2 - 2 \cdot (2m)^2 = 1 \tag{36}$$

Setting $x = 2n+1$, $y = 2m$ and $d = 2$ we can see that any solution of the problem must fulfill a Pell's equation $x^2 - dy^2 = 1$ for the integers x and y, here in the special case with $d = 2$. See for example [9], section 10.4 or [11], section 14.5.

It follows from the theory of Pell's equations that (34) has an infinite number of solutions that can be rather easily calculated, see program and table below.

```
public class Micropuzzle42
{
  private static void printSolutions(int solutions)
  {
    final double sqrt2 = Math.sqrt(2.0);
    int n = 2*solutions-1;
    long[] P = new long[n+1];
    long[] Q = new long[n+1];
    P[0] = 1;
    Q[0] = 1;
    P[1] = 3;
    Q[1] = 2;
    for (int k = 2; k <= n; ++k) {
```

```
      P[k] = 2 * P[k-1] + P[k-2];
      Q[k] = 2 * Q[k-1] + Q[k-2];
    }

    System.out.println(" No.     numerator    denominator         quotient          quotient-sqrt(2)"
    for (int k = 0; k <= n; ++k) {
      double p = P[k];
      double q = Q[k];
      System.out.printf("%3d %14d %14d  %19.15f %19.15f%n", k, P[k], Q[k], p/q, p/q - sqrt2);
    }
    System.out.println();
    System.out.println("  i    k         runner  total runners");
    for (int i = 1; i <= solutions; ++i) {
      int k = 2*i-1;
      long runner = Q[k] / 2;
      long total  = (P[k]-1) / 2;
      System.out.printf("%3d %4d %14d %14d%n", i, k, runner, total);
    }
  }

  public static void main(String[] args)
  {
    printSolutions(16);
  }
}
```

The first solutions are given in the following table with m the number of our runner and n the total number of runners. The first row with $m = n = 1$ is the special case of our runner doing a solo race.

m	n
1	1
6	8
35	49
204	288
1 189	1 681
6 930	9 800
40 391	57 121
235 416	332 928
1 372 105	1 940 449
7 997 214	11 309 768
46 611 179	65 918 161
271 669 860	384 199 200
1 583 407 981	2 239 277 041
9 228 778 026	13 051 463 048
53 789 260 175	76 069 501 249
313 506 783 024	443 365 544 448

The $T_n = \frac{1}{2}n(n+1)$ with n taken from the above solutions are exactly the triangular numbers that are also squares, compare page 204 in [4].

The values of m in the first column are just the sequence M4217 and the values of n in the second column are the sequence M4536 in [15].

4.14 Micropuzzle 45 – Palindromic cycles

Puzzle from the book [2]: "That little sounds like a bike with a saddle at both ends.

If any two-digit number is reversed and added to itself, and the process repeated over and over, eventually a palindromic number will result (i. e. one which reads the same forwards as backwards).

Thus, consider the number 19. When reversed it gives 91. Then $19 + 91 = 110$.

And repeating $110 + 011 = 121$, which is palindromic after only two operations.

Which two-digit number requires the most number of operations before a palindromic sum is reached? And how many are required?

Clearly there will be two answers – since one will be the reverse of the other. Either one is acceptable."

The case from the puzzle: 2-digit numbers

The problem can be rather well analyzed for two-digit numbers:

Let a and b be the digits of this number, then the number is given by $10a + b$ with $a \in \{1, 2, \ldots, 9\}$ and $b \in \{0, 2, \ldots, 9\}$, in the example of 19 it is $a = 1$ and $b = 9$.

If and only if the two digits are identical, this means $a = b$, the number is already a palindrome, and we are already finished with 0 operations.

The reversed number of $10a + b$ is $10b + a$, and the sum of the original two-digit number and its reverse then becomes

$$(10a + b) + (10b + a) = 10a + b + 10b + a = 11a + 11b = 11(a + b)$$

showing that the result of the first step only depends on the sum $a + b$ of the two digits a and b of the original two-digit number.

If it is now $a + b \leq 9$, then after this one step both digits become $a + b$ and we are finished after just one step. For example it is $34 + 43 = 77$.

The digit sum of 18 can only be achieved by both $a = 9$ and $b = 9$, but then $a = b$ and the number 99 is already palindromic.

Therefore the cases with $a \neq b$ and digit sum $10 \leq a + b \leq 17$ remain. They are treated by the following table. The lengthy but easy calculation in case of $a + b = 17$ is only shown partially.

$a+b$	steps after the initial one	steps
10	$110 \to 121$	2
11	121	1
12	$132 \to 363$	2
13	$143 \to 484$	2
14	$154 \to 605 \to 1111$	3
15	$165 \to 726 \to 1353 \to 4884$	4
16	$176 \to 847 \to 1595 \to 7546 \to 14603 \to 44044$	6
17	$187 \to 968 \to 1837 \to \ldots \to 8813200023188$	24

Only the two-digit numbers 89 and 98 have a sum of digits of 17, therefore these are the solutions of the puzzle with the maximum number of 24 steps.

The general case

The actual puzzle solution for only two digit numbers could have been done within the range of the long Java data type. But the interesting cases are beyond the two-digit numbers of the puzzle.

Here some facts in a short overview

- The highest number of iterations for two-digit numbers is 24 and is reached for both the numbers 89 and 98.
- There are numbers where the iteration does not end in a palindromic number even after a large number of iterations. The first such number is 196, and it is still an open problem in mathematics if this iteration ever ends in a palindromic number or if it goes to infinity.
- The number of 24 iterations to a palindromic number is – the exceptional cases excluded – first surpassed only for the number 10 548 with 30 iterations ending in 17 858 768 886 785 871.
- For numbers up to 1 000 000 the record number of iterations is 64 and achieved for the first time for the number 150 296 ending in the palindromic number 682 049 569 465 550 121 055 564 965 940 286.

Therefore my solution program Micropuzzle45.java performs the iterations only until a given limit of 1000 and lists these exceptional cases at the end of its output. This limit is set by the internal constant `ITERATION_LIMIT`.

```
import java.math.BigInteger;
import java.util.SortedSet;
import java.util.TreeSet;

public class Micropuzzle45
{
    public static BigInteger reverseDigits(BigInteger n)
    {
        StringBuilder sb = new StringBuilder(n.toString());
        return new BigInteger(sb.reverse().toString());
    }

    public static void main(String[] args)
    {
        int nMax;
        if (args.length == 0) {
            nMax = 99;  // highest two-digit number, value from puzzle
        } else {
            nMax = Integer.parseInt(args[0]);
        }
        int maxCount = -1;  // not yet found any maximum
        int maxNumber = 0;
        final int ITERATION_LIMIT = 1000;
        SortedSet<Integer> exceptionalCases = new TreeSet<>();
        loop: for (int n = 10; n <= nMax; ++n) {
            BigInteger number = BigInteger.valueOf(n);
            BigInteger reverseNumber = reverseDigits(number);
            int count = 0;
            while (number.compareTo(reverseNumber) != 0) {
```

```
                ++count;
                if (count > ITERATION_LIMIT) {
                    exceptionalCases.add(n);
                    continue loop;
                }
                number = number.add(reverseNumber);
                reverseNumber = reverseDigits(number);
            }
            if (count > maxCount) {
                maxCount = count;
                maxNumber = n;
            }
            System.out.printf("%8d %7d   %d%n", n, count, number);
        }
        if (maxCount >= 0) {
            System.out.println();
            System.out.println("Maximum of " + maxCount +
                    " iterations reached for number " + maxNumber);
        }
        if (!exceptionalCases.isEmpty()) {
            final int NUMBERS_PER_LINE = 8;
            System.out.println();
            System.out.println("No palindrome after " + ITERATION_LIMIT + " iterations:");
            int outputCount = 0;
            for (Integer n : exceptionalCases) {
                System.out.printf(" %8d", n);
                ++outputCount;
                if (outputCount % NUMBERS_PER_LINE == 0)
                    System.out.println();
            }
            if (outputCount % NUMBERS_PER_LINE != 0)
                System.out.println();
        }
    }
}
```

The program can be compiled with

```
javac Micropuzzle45.java
```

on the command line. It can then be run by

```
java Micropuzzle45
```

as the default case to solve the puzzle or with an integer argument to set another upper limit for the search, for example

```
java Micropuzzle45 12000
```

would run it for all integers up to 12000.

The reversal of the number is done by first using toString method to transfer the BigInteger to a string, reverse the string, and transfer it back to a BigInteger number. Avoiding the string conversions could save processing time.

Many of the long iteration sequences eventually run into the same numbers. By storing some of these numbers and checking for already investigated sequences it could be saved time.

See also the book by Martin Gardner [6], chapter 3 for some additional information.

4.15 Micropuzzle 54 – The ladder and the wall

Let the vertical distance from floor to top of the ladder be x feet.
Let the horizontal distance from wall to bottom of ladder be y feet.

Then by theorem of Pythagoras

$$x^2 + y^2 = 15^2 \tag{37}$$

and by the two similar triangles above and right of the box (all angles are equal)

$$\frac{x-3}{3} = \frac{3}{y-3} \tag{38}$$

The two equations (37) and (38) together are a non-linear equation system of two equations with two real variables x and y as unknowns. By the geometry of the problem it must be $3 < x < 15$ and $3 < y < 15$.

Multiplying both sides of (38) with both denominators gives

$$(x-3)(y-3) = 3 \cdot 3$$

$$xy - 3x - 3y + 9 = 9$$

$$xy = 3(x+y) \tag{39}$$

By the binomial theorem

$$(x+y)^2 = x^2 + 2xy + y^2$$

it is

$$x^2 + y^2 = (x+y)^2 - 2xy$$

Inserting this into (37) gives together with (39) an equation system

$$(x+y)^2 - 2xy = 15^2 \tag{40}$$

$$xy = 3(x+y) \tag{41}$$

where the variables x and y only appear in the forms $x+y$ and xy. This suggests the substitutions

$$u = x + y, \qquad v = xy \tag{42}$$

resulting in the equation system for u and v

$$u^2 - 2v = 15^2 \tag{43}$$

$$v = 3u \tag{44}$$

Inserting the expression for v from the second equation into the first one obtains the quadratic equation

$$u^2 - 6u - 15^2 = 0$$

with the two solutions

$$u_{1,2} = 3 \pm \sqrt{3^2 + 15^2} = 3 \pm 3\sqrt{1^2 + 5^2} = 3\left(1 \pm \sqrt{26}\right)$$

Because $u = x + y > 0$ only the positive solution matters, so it must be

$$u = 3\left(1 + \sqrt{26}\right)$$

and by (44)

$$v = 3u = 9\left(1+\sqrt{26}\right)$$

Using as shortcut

$$R = 1+\sqrt{26}$$

and inserting the values of u and v into (42) the equations for x and y become

$$\begin{aligned} x+y &= 3R \\ xy &= 9R \end{aligned}$$

By the second equation it is $y = 9R/x$, inserting this into the first equation obtains another quadratic equation, now for x

$$x + \frac{9R}{x} = 3R$$

or

$$x^2 - 3Rx + 9R = 0$$

with the solution

$$\begin{aligned} x_{1,2} &= \frac{3}{2}R \pm \sqrt{\frac{9}{4}R^2 - 9R} \\ &= \frac{3}{2}\left(R \pm \sqrt{R^2 - 4R}\right) \\ &= \frac{3}{2}\left(R \pm \sqrt{R(R-4)}\right) \end{aligned}$$

This gives the two possible solutions

$$\begin{aligned} x_1 &\approx 14.515503482069 \\ x_2 &\approx 3.781555058710 \end{aligned}$$

The distance between the top of the wall and the top of the ladder then becomes $15 - x_{1,2}$ resulting in the two solutions of 0.4844965 feet or 11.2184449 feet.

4.16 Micropuzzle 55 – More ladders and walls

Let a be the height at which the longer ladder of 30m rests against the right wall and b the height where the 25m ladder leans at the other wall, d the distance between the two walls we look for and for convenience let $h = \frac{504}{100}$ the height of the meeting point of the two ladders over the ground.

By the theorem of Pythagoras we have

$$\begin{aligned} a^2 + d^2 &= 30^2 && (45) \\ b^2 + d^2 &= 25^2 && (46) \end{aligned}$$

This gives $d^2 = 30^2 - a^2$ and $d^2 = 25^2 - b^2$, hence

$$30^2 - a^2 = 25^2 - b^2$$

or by adding $a^2 - 25^2$ on both sides

$$a^2 - b^2 = 30^2 - 25^2 \qquad (47)$$

Now let d_1 and d_2 be the distances from the left wall to the meeting point of the two ladders and the distance from there to the right wall, $d_1 + d_2 = d$.

The two right triangles with legs d_1, h and legs d, a are similar, thus

$$\frac{d_1}{d} = \frac{h}{a} \tag{48}$$

and also by the same argument for the other ladder

$$\frac{d_2}{d} = \frac{h}{b} \tag{49}$$

Adding both sides of (48) and (49) gives

$$\frac{d_1 + d_2}{d} = \frac{h}{a} + \frac{h}{b}$$

or by using $d_1 + d_2 = d$, dividing by h and exchanging sides it is

$$\frac{1}{a} + \frac{1}{b} = \frac{1}{h}$$

Resolving for b gives

$$b = \frac{ha}{a - h} \tag{50}$$

Inserting (50) into equation (47) then gives

$$a^2 - \frac{(ha)^2}{(a-h)^2} = 275$$

and by the steps

$$\begin{aligned} a^2(a-h)^2 - h^2a^2 &= 275(a-h)^2 \\ a^2(a^2 - 2ah + h^2) - h^2a^2 &= 275(a^2 - 2ah + h^2) \\ a^4 - 2ha^3 + h^2a^2 - h^2a^2 &= 275(a^2 - 2ah + h^2) \end{aligned}$$

the quartic equation for a (as h is a known constant value)

$$a^4 - 2ha^3 - 275a^2 + 550ha - 275h^2 = 0 \tag{51}$$

Inserting the value of $h = \frac{504}{100}$ into this equation yields

$$a^4 - \frac{252}{25}a^3 - 275a^2 + 2772a - \frac{174636}{25} = 0 \tag{52}$$

A quartic equation is still solvable in a closed form, see for example [16], section 2.2 about polynomials, [1], section 3.8 or [13], section 1.11(iii), but this is not a really practical way.

Alternatively you could look into the equations (45) and (46) and search for integer solutions. After some guessing you might deduce from $3^2 + 4^2 = 5^2$ that $(6\cdot 3)^2 + (6\cdot 4)^2 = (6\cdot 5)^2$ and therefore $18^2 + 24^2 = 30^2$ and inserting 18 into (52) you see that you found a solution, then by extracting the linear factor $(a - 18)$

$$\frac{1}{25}(a - 18)(25a^3 + 198a^2 - 3311a + 9702) = 0$$

Or the equation (52) can be solved with computer help, either numerical or symbolic, resulting in the only positive real solution $a = 18$, a negative real and two conjugate complex solutions.

Only the positive real solution geometrically makes sense, and then

$$d = \sqrt{30^2 - 18^2} = \sqrt{900 - 324} = \sqrt{576} = 24$$

The walls are 24m apart.

References

[1] M. Abramowitz, I. A. Stegun, editors, *Handbook of Mathematical Functions.* United States Government Printing Office, 1964.

[2] J. J. Clessa, *Math and Logic Puzzles for PC Enthusiasts.* Dover, 1996.

[3] Henri Cohen, *A Course in Computational Algebraic Number Theory.* Springer, 1996.

[4] John H. Conway, Richard K. Guy, *The Book of Numbers.* Copernicus, 1995.

[5] David Cox, John Little, Donal O'Shea, *Ideals, Varieties and Algorithms*, 3rd edition. Springer, 2007.

[6] Martin Gardner, *The Colossal Book of Mathematics.* W. W. Norton & Company, 2001.

[7] Joachim von zur Gathen, Jürgen Gerhard, *Modern Computer Algebra*, 2nd edition. Cambridge University Press, 2003.

[8] Keith O. Geddes, Stephen R. Czapor, George Labahn, *Algorithms for Computer Algebra.* Kluwer Acacemic Publishers, 1992.

[9] Peter Giblin, *Primes and Programming – An Introduction to Number Theory with Computing.* Cambridge University Press, 1993.

[10] Ronald L. Graham, Donald E. Knuth, Oren Patashnik, *Concrete Mathematics: A Foundation for Computer Science*, 2nd edition. Addison-Wesley, 1994.

[11] G. H. Hardy, E. M. Wright, *An Introduction to the Theory of Numbers*, 5th edition. Oxford University Press, 1979.

[12] Donald E. Knuth, *The Art of Computer Programming v. 2. Seminumerical Algorithms*, 3rd edition. Addison-Wesley, 1997.

[13] Frank W. J. Olver et al, *NIST Handbook of Mathematical Formulas.* Cambridge University Press, 2010.

[14] Victor Shoup, *A Computational Introduction to Number Theory and Algebra*, 2nd edition. Cambridge University Press, 2009.

[15] N. J. A. Sloane, Simon Plouffe, *The Encyclopedia of Integer Sequences*, Academic Press, 1995.

[16] D. Zwillinger, Editor-in-Chief, *CRC Standard Mathematical Tables and Formulae*, 30th edition. CRC Press, 1996.