Appendix B

Solutions to Some Exercises

B.1 Sections 1.1 to 1.7

1. a) $f(20)/f(19) = 4.7326261527$ b) Same answer as in a). Use Listing 1.25 after changing 1E-09 into 1E-08.

2. bisiter-q2.py. This iterative program finds a root of the continuous function f in the interval $[a, b]$, if $f(a) * f(b) < 0$. The boolean variable `faneg` is true if $f(a)$ is negative. The name should remind you of this.

3. a) 0.73908513226 b) 1.7632228322 c) 0.56714328937 d) 4.493409458

4. x_1 = -1.8793852418, $x_2 = 0.34729635529$, $x_3 = 1.5320888860$.

5. b) We conjecture $g^2(n) - 5 * f^2(n) = 4 * (-1)^n$. Proof by induction.

6. a) True. b) True. Proof by induction.

7. a) $f(1/3) = p \cdot f(2/3) = p(p + q \cdot f(1/3))$, $f(1/3) = p^2/(1 - pq)$.

b) $f(2/5) = p \cdot f(4/5) = p^2 + pq$?f(3/5) $= p^2 + p^2q + pq^2$f(1/5) $= p^2 + p^2q + p^2q^2/(2/5)$, f(2/5) $= p^2(1 + q)/(1 - p^2q^2)$.

c) f(1/1984) $= p^{11}/(1 - p^4q)$. For $p = q = 1/2$ we get $f(x) = x$.

d) The longer the period of the binary expansion of x the harder it is to find $f(x)$.

8. joseph2-q8.py

11. We get an unbranching recursion if we compute u values in blocks of two. This reduces the number of operations to $O(\log n)$. (The time complexity is greater because just the number of digits in $u(n)$ is $O(n)$.)

12. By a slight modification of joseph1.py we get Listing 5.25.

13. a)

```
s = 0; n = 0
while n < 10000:
  n += 1
  s += 1/n
```

b)

```
s = 0; n = 10000
while n < 0:
  s += 1/n
  n -= 1
```

c) When two positive floating-point numbers with different exponents are added, the digits of the smaller number which are in places below the least significant digit of the larger number are mostly neglected. At best, they are taken into account in the rounding of the result. Thus, if we form the sum starting with the smallest terms, the error will be smaller than if we start with the largest terms. It would be even better to add groups of small terms and then add the resulting totals instead of adding the terms individually to the total.

14. The computations suggest

$$\text{round}(\frac{n}{2}) + \text{round}(\frac{n}{4}) + \text{round}(\frac{n}{8}) + ... = n. \tag{B.1}$$

This can be proved as follows. The formula is true for $n = 0$. If we increase n by 1, the term $\text{round}(n/2^k)$ in the sum (B.1) is unchanged except when n reaches a value which is an odd multiple of 2^{k-1}, and then it increases by 1. For each value of n there is one such k and therefore the sum of the series is n.

15. a)

```
for x in range(100):
  for y in range(100):
    if abs(x*x - x*y - y*y) == 1:
      print(x, y);
```

We get pairs of successive Fibonacci numbers.

b) We observe that the solutions are

$$(\text{fib}(n+1), \text{fib}(n)) \quad n = 1, 2, ... \tag{B.2}$$

It is easy to verify that

$$\text{if } w = u + v, \text{ then } w^2 - wv - v^2 = -(v^2 - vu - u^2). \tag{B.3}$$

This equation enables us to start from the solution $(1, 1) = (\text{fib}(2), \text{fib}(1))$ and obtain the family of solutions (B.2).

Next we show that if x and y are > 0 and

$$|x^2 - xy - y^2| = 1 \tag{B.4}$$

then (x, y) is in the family (B.2).

If $x = 1$ then Equ. (B.4) implies $y = 1$ so we have the first member of the family (B.2). If $x > 1$, (B.4) implies $y < x$. By (B.3), $(y, x - y)$ is also a solution with positive components. We can continue to descend as long as the x value is > 1, i.e. until we reach $(1, 1)$. We can now reverse the process and get back to the (x, y) we started with, and this amounts to forming the Fibonacci sequence.

16. We give five different solutions, which are successively more sophisticated and efficient. A sixth program will be given later.

equisum1-q16.py
equisum2-q16.py
equisum3-q16.py
equisum4-q16.py
equisum5-q16.py

Listings `equisum1-q16.py` to `equisum4-q16.py` use 1000000, 28000, 2800, 1400 comparisons, respectively. In Listing `equisum2-q16.py` we denote by $p(s)$ the number of triples (a, b, c) with fixed sum s. For each such sum $s = a + b + c$ there are $p(s)$ triples (d, e, f) with the same sum $s = d + e + f$. So $p(s) * p(s)$ is the total number of cases with sums. This is added up for $s = 0$ to 27. In Listing `equisum3-q16.py` we save the inner c-loop by means of the test if $(a + b \le s)$ and $(a + b \ge s - 9)$, which is satisfied for $0 \le c \le 9$. For given s, a, b the variable c is also given by $c = s - a - b$.

In Listing `equisum4-q16.py` we use the bijection $abcdef \leftrightarrow a'b'c'd'e'f'$ between blocks with equisum property, where $a' = 9 - a$, etc. If $s = a + b + c$ then $s' = a' + b' + c' = 27 - s$. So triples (a, b, c) with sums and $27 - s$ are equi-numerous. Here s runs from 0 to 13 and the total count is doubled at the end.

The recursive program Listing `equisum5-q16.py` is based on the recurrences

$$p(s) = p(s-1) + s + 1 \quad \text{for } s \le 9,$$
$$p(s) = 100 - p(s-10) - p(17-s) \quad \text{for } s > 9.$$

A triple (a, b, c) with sum s is determined by its two digits a, b. Also, any pair of digits a, b with

$$s - 9 \le a + b \le s \tag{B.5}$$

will form such a triple with $c = 9 - a - b$. Thus $p(s)$ is the number of solutions of (B.5). To count these, consider the 100 lattice points (a, b) in or on the boundary of the square $\mathrm{S} : 0 \le a, b \le 9$. For $s \le 9$, $p(s)$ is the number of lattice points in or on the boundary of the triangle cut off from S by the line $a + b = s$. For $s > 9$ we can think of the lattice points in the region (B.5) as the lattice points in S from which we remove the lattice point in or on the boundary of the triangle $a + b \le s - 10$ and also the lattice points in or on the boundary of the triangle $a + b \ge s + 1$. The last triangle is congruent to the triangle $a + b \le 17 - s$.

B.2 Section 2.1

4. a) We want to find $x = 7^{9999} \bmod 1000$. We first find the smallest power of 7 which is 1 (mod 1000).

```
p = 1; i = 0
while True:
  i += 1;
  p = (7*p) % 1000
  if p == 1:
    break
```

We get $i = 400$. Since $9999 = 400 * 24 + 399$ we have $7^{9999} \equiv 7^{399} \pmod{1000}$.

```
p = 1
for i in range(399):
  p = (7*p) % 1000;
```

Now $p = 143$, which is the solution x.

A completely different solution is based on Listing 2.4. We want to find $x = 7^{9999} \bmod 1000$, or $7x = 7^{10000} \bmod 1000$. Now $7^{10000} = 7^{400*25} = 1 \pmod{1000}$. Thus $7x \equiv 1 \pmod{1000}$ or $7x + 1000y = 1$. Listing 2.4 gives $x = 143$, $y = -1$, and so $x = 143$. b) and c) may require Listing 2.5.

6. a)

```
sum = 0
for i in range(11):
  sum += i
```

b)

```
def sum(n):
  if n == 1:
    return 1
  else:
    return sum(n-1)+n
```

7. a)

```
def bin(n):
  if n > 1:
    bin(n // 2)
  print(n % 2, end='')
```

b)

```
while n > 0:
  print(n % 2, end='')
  n = n // 2
```

In b) the digits are printed in reverse order. In a) the digits are printed in correct order. But if we switch the bin() and print() procedures we get the bits in reverse order.

binary-q8.py

8.

```
d = 0
while n > 0:
  d += n % 10
  n = n // 10
```

9. digdig-q10.py

$f(n)$ is 9 if n is a multiple of 9; otherwise it is n mod 9.

10. rev6-q11.py

11. subs6-q12.py

12. subs6-q13.py

13. q(n) = fib(n + 1)/fib(n) approaches $q = 1.6180339887... = (\sqrt{5}+1)/2$. Show by induction that $q(n) - q = (-1)^n/\text{fib}(n)/q^n$.

fibRatio-q15.py

14. divBy-q16.py

15. a) The sequence completes a cycle if two successive terms repeat. This must happen because of the pigeonhole principle.

b) Addition mod m is invertible. Hence the sequence can be extended uniquely into the past. But in a loop with a tail the end of the tail has two predecessors.

c) FibPeriodMODm-q17.py

16. a) $L(p^n) = L(p) * p^{n-1}$ for every prime p.

b) $L(p_1^{a_1} p_2^{a_2} \cdots p_r^{a_r}) = \text{lcm}(L(p_1^{a_1}), L(p_2^{a_2}), \cdots, L(p_r^{a_r}))$. In particular,

$$L(10^n) = \text{lcm}(3 * 2^{n-1}, 2^2 * 5^n) = \begin{cases} 12 * 5^n & \text{for } n \le 3 \\ 15 * 10^{n-1} & \text{for } n > 3 \end{cases}$$

17. a) $L|p-1$. b) $L|2(p+1)$. c) $L = 20$.

19. Zeta(3) $= 1 + 1/2^3 + \cdots + 1/n^3 + 1/2(n+1/2)^2 = 1.2020569032...$ from $n = 200$ on.

20. Let A_d be the event gcd(x, y, z) = d. A_d is equivalent to the occurrence of four independent events

$$d|x, \quad d|y, \quad d|z, \quad \gcd(x/d, y/d, z/d) = 1.$$

Thus

$$P(A_e) = \frac{1}{d} * \frac{1}{d} * \frac{1}{d} * p_3 = p_3/d^3, \text{ where } p_3 = P(\gcd(x, y, z) = 1).$$

Since $A_1 \cup A_2 \cup A_3 \cup ...$ will occur with probability 1, we have

$$p_3 \sum_{d \ge 1} 1/d^k = 1, \text{ or } p_3 = 1/\sum_{d \ge 1} 1/d^3.$$

21. Let A_d be the event $\gcd(x_1, \cdots, x_k) = d$. That is

$$d|x_1, \cdots, d|x_n, \quad \gcd(x_1/d, ..., x_n/d) = 1.$$

Then $P(A_d) = p_k/d^k$, where $P(A_1) = 1$. Thus

$$p_k \sum_{d\geq 1} 1/d^k = 1 \rightarrow p_k = 1/\sum_{d\geq 1} 1/d^k.$$

22. a) will be proved after c).

First we show that the process we gave for generating the Morse-Thue sequence is equivalent to the following:

Denote by S the operation, applicable to a sequence of 0's and 1's, which replaces each 0 by 01 and each 1 by 10. Observe that 10 is the complement of 01 so that if we apply S to a pair of complementary sequences we get a pair of complementary sequences. We claim the initial segment of 2^n digits of the Morse-Thue sequence is $S^n(0)$. Indeed this is clear for $n = 0$ and $n = 1$. Now $S^{n+1}(0) = S^n(01)$, which is $S^n(0)$ followed by $S^n(1)$. Since the S-operator takes complements into complements, what we have now is the rule of formation of the Morse-Thue sequence originally given.

Observe that the operation S can be applied to the entire infinite Morse-Thue sequence and it leaves the sequence unchanged. Suppose now that $x(n) = 0$. In the Morse-Thue sequence $x(n)$ is preceded by n digits. If we apply S to the sequence we replace $x(n) = 0$ by 01, and then digits preceding it by $2n$ digits. So in this case $x(2n) = x(n)$ and $x(2n + 1) = 1 - x(2n)$. The case $x(n) = 1$ follows similarly. This proves c); the assertion b) is just another way of phrasing the first part of c).

To prove a), we note that the two equations c) together imply that $x(n)$ and $x(2n + 1)$ are always different. If the sequence were ultimately periodic and $x(n)$ were in the periodic part of the sequence, we could conclude that $n + 1$ is not a multiple of the period. The same would be true of $n + 2$, $n + 3$, ... but this is impossible.

c) MorseThueC-q24.py

d) is just a way of expressing the definition of the sequence.

MorseThueD-q24.py

e) is true because the sequence of $n+1$ digit binary numbers is obtained from the sequence of all numbers with up to n digits by putting a 1 and possibly some 0's in front of them.

MorseThueE-q24.py

23. b) One can argue the same way as for the Morse-Thue sequence above.

Keane-q25.py

24. You save about 12% in running time.

25. pascalTriangle-q27.py

Use $n = 15$ for input.

c) Replace the 5th line from below by

```
x[i] = (a+b) % 2
print(x[i])
a = b
b = x[i+1]
```

26. a)

```
for i in range(1,801):
  if (i-1)%3 == 0:
    if (i-1)%4 == 0:
      if (i-1)%5 == 0:
        if i%7 == 0:
          print(i)
```

A better approach is this: $i - 1$ is a multiple of 3, 4, 5, i.e. of 60. So we must check the terms $60i + 1$ for divisibility by 7.

```
for i in range(1,50):
  if (60*i + 1)%7 == 0:
    print(60*i+1)
```

b) One gets $n = 301 + 420t$. For $t = 0$ we get $n = 301$ eggs of weight ≈ 17 kg. For $t = 1$ we get $n = 721$ eggs of weight ≈ 40 kg, which can be carried by a strong man on his back, but not by an old woman in her basket.

c) One generalization: by taking them a, b, c at a time she got remainders r, s, t, but by taking them d at a time, it came out even:

```
for i in range(1,max+1):
  if (i-r)%a == 0:
    if (i-s)%b == 0:
      if (i-t)%c == 0:
        if i%d == 0:
          print(i)
```

If the smallest non-negative solution is s then all solutions are $x = s + t * \mathrm{lcm}(a, b, c)$ (Chinese remainder theorem.)

28. a) Let $m = a + b$. Then $b \equiv -a \pmod{m}$. In every step $a \bmod m$ is replaced by $2a \bmod m$. Hence the process will terminate if for some i, $2^i a$ is a multiple of m. This will be the case if and only if a is divisible by the largest odd divisor of m.

b) If a and b have a common divisor, divide it out. So we can assume with out loss of generality that $\gcd(a, m) = 1$. We get pure periodicity if for some n, $2^n a = a \pmod{m}$. This means $(2^n - 1)a$ is divisible by m. This implies $2^n \equiv 1 \pmod{m}$. So m divides $2^n - 1$ and it must be odd. Conversely, if m is odd, $2^n \equiv 1 \pmod{m}$ for some n and the least such n is the length of the cycle.

29. $f(0) = 0$, $f(n) = n \bmod 2 + f(n \mathbin{//} 2)$ for $n > 0$

30. $n = 5186$

31. involution-q33.py

B.3 Sections 2.2 to 2.4

3. pyttrip3.py

7. $c_4 = \pi/2$

8. $c_{2k} = \pi^k/k!, c_{2k+1} = 2(2\pi)^k/(1 \times 3 \times \cdots \times (2k+1))$. The volume of the unit ball approaches 0 as k goes to infinity. The proportion goes to 0 even faster.

10. $\pi = 3.1415927054$ and $\pi = 3.1415926535$. All digits of the second number are correct.

B.4 Section 2.5

1. prison-q1.py

2. The 'obvious' program for the U-sequence is

unique-q2.py

Comment: `can` is the candidate currently being tested. If it can be represented in one way ($l = 1$) then it is stored in an array a[1..n] of successful candidates.

3. An 'obvious' program is:

unique1-q3.py

Use 50 or 100 for n.

4. a) firstFib-q4.py

For b) and c) one gets the same frequencies.

5. $q^2 \nmid n$ with probability $1 - 1/q^2$ for any prime q. The probability that a 'random' integer is square free is

$$p = \prod_{\text{all primes}} (1 - 1/q^2) = 6/\pi^2.$$

6. Look up the solution of Ex. 22 of Sections 2.6 to 2.12 below. There you will find an efficient program for the Frobenius Problem.

7. c) collatz-q7.py

This is one of many versions using a procedure to get from n to its successor $f(n)$.

11. cycles-q11.py

B.5 Section 2.5.4

2. We show first that the sequence of numbers whose ternary representations contain no 2's contains no three numbers a, b, c such that $a + b = 2c$. The digits of $2c$ are all 0's and 2's. Since the digits in a and in c are all 0's and 1's, the equation could hold only if the 1's in a were in the same place as the 1's in b, which would mean $a = b$.

Next we show that every number which is not in the sequence forms an arithmetic progression with two smaller numbers which are in the sequence. Let c be a number which has one or more 2's in its ternary representation. Let b be obtained by replacing these 2's by 1's, and let a be obtained by replacing them by 0's. Then a, b, c form an arithmetic progression. Since the only numbers excluded from our sequence are the ones which form an arithmetic progression with two smaller numbers in the sequence, our sequence is the one given by the greedy algorithm.

B.6 Sections 2.6 to 2.12

1. rotation-q1.py

2. The recursive program is clear. Here is the iterative program:
bIter-q2.py

6. a) 2498 b) 1229587

7. 63992

8. 4562

9. $a(200, 8) = 104561$ with D = {1, 2, 4, 10, 20, 40, 100, 200}. Scale by 2, so that coins become integers.

10. weights-q10.py

Here $m = a_1 + \cdots + a_s$ and c[n] is the number of solutions of $a_1x_1 + + a_sx_s = n$. The numbers c[0],..., c[m] are stored in the array c[0..m].

11. pans-q11.py

This straightforward program is valid for $s = 4$ and $m = a[1] + a[2] + a[3] + a[4]$.

It is easy to see how to generalize and complete the program. Unfortunately its time complexity is $O(3^s)$. So it is only feasible for small s, say $s = 12$. The following program has complexity which is proportional to $n * s$, i.e. $O(n * s)$.

rep3-q11.py

12. c) TwoThreeFive-q12.py

17. revit-q17.py

18. a) fn-q18.py

b) In the interval 1..1988 the relation $i = f(i)$ is satisfied 92 times.

g) fn-18g.py

21. $CBA = (A^R B^R C^R)^R$. Write the corresponding program. It uses the procedure reverse four times.

22. The following algorithm can be used in the exploration of the Frobenius Problem:

frobenius-q22.py

Let $b(k,i)$ be the number of ways of representing k as a sum of numbers from among $a(1),...,a(i)$. The same summand may occur several times. Representations which differ only in the order of the terms are considered the same. Let $b(0,i) = 1$ and $b(k,0) = 0$ for $k > 0$. (This expresses the useful convention that the empty sum has value 0). A representation of k as a sum of numbers from among $a(1),...,a(i)$ is one of two kinds: either it has no summand $a(i)$ or it is obtained from a representation of $k - a(i)$ by adding $a(i)$ to it. This gives the recursion formula

$$b(k,i) = \begin{cases} b(k,i-1) & \text{if } k < a(i) \\ b(k,i-1) + b(k-a(i),i) & \text{if } k \geq a(i). \end{cases} \tag{B.6}$$

It is easy to write a program which computes the lists $b[k][1], b[k][2], ...$ by means of Equ. (B.6). Observe that once $b[k][i]$ has been computed, $b[k][i-1]$ is no longer needed so instead of a two-parameter list b[k][i] we can use a 1-parameter list b[0..n] and recompute it for successive values of i. In this way we get the algorithm above.

23. Let $p(k,i)$ be the number of ordered i-tuples of decimal digits with sum k. We have $p(0,i) = 1$ for $i = 0,1,...$ and $p(k,0) = 0$ for $k = 1,2,....$ It will simplify our formulas to define $p(k,i) = 0$ for $k = -1,-2,...,-9$ and all i. A recursive formula for $p(k,i)$ $(k,i > 0)$ can be obtained as follows: the i-tuples with sum k and last digit 1, ..., 9 are obtained from the i-tuples with sum $k-1$ and last digit 0, ..., 8 by adding one to the last digit. The number of the latter i-tuples is $p(k-1,i) - p((k-1)-9,i-1)$. The number of i-tuples with sum k and last digit 0 is $p(k,i-1)$. Combining these facts we get

$$p(k,i) = p(k-1,i) + p(k,i-1) - p(k-10,i-1), \quad k,i > 0. \tag{B.7}$$

We know the number of ordered triples with sum k is the same as the number of ordered triples with sum $27-k$. Hence the number of 6-tuples we want is the same as the number of 6-tuples such that the sum of all 6 digits is 27; we get one kind from the other by replacing each of the last 3 digits d_i by $9-d_i$. Thus $p(27,6)$ is an answer to our question.

equisum6-q23.py

B.7 Sections 2.13 to 2.14

8. chain-q8.py

By means of `chain-q8.py` we get the pairs of primes in Table B.1.

We can speed up `chain-q8.py`. Change the loop as follows:

```
while (not a) or (not b):
  p += 2
  a = prime(p, 3)
  if a:
    b = prime(p+p+1, 3)
```

max	p	2p+1
1000	1013	2027
10000	10061	20123
100000	100043	200087
1000000	1000151	2000303
10000000	10000079	20000159
100000000	100000223	200000447

Table B.1. Outputs for various max values

In the program each time both p and $2p+1$ are tested for primality. In the modified program, $2p+1$ is only tested if p is prime, and we must declare the boolean variables a and b.

9. The Euler polynomial $f(x) = x^2+x+41$ is especially rich in primes. From 0 to 2398 there are 1199 primes, i.e. 50%. From 0 to 4000 there are 1860 primes, i.e. 46.5%.

EulerPolynomial-q9.py

16. $n = 4^k(8q+7)$.

17. a) $8q+7$ is not the sum of 3 squares.

b) If $4^k(8q+7)$ is a sum of 3 squares then so is $4^{k-1}(8q+7)$

19. taxi-q19.py

20. $6578 = 1^4 + 2^4 + 9^4 = 3^4 + 7^4 + 8^4$

21. sieveout-q21.py

```
> python  sieveout-q21.py
n=? 100
3 5 7 8 9 11 13 14 15 17 19 20 21 23 24 25 26 27 29 31 32
33 34 35 37 38 39 41 43 44 45 47 48 49 50 51 53 54 55 56
57 59 61 62 63 64 65 67 68 69 71 73 74 75 76 77 79 80 8
1 83 84 85 86 87 89 90 91 92 93 94 95 97 98 99
```

Transform the term $z+y+xy$ so that you can see which integers are contained in this list. The output is for $n = 100$.

B.8 Section 2.16

4. coinStack-q4.py

B.9 Chapters 1 to 2

1. skip-q1.py

This program lists the numbers skipped by our function. For input 1000 it gives the squares up to 961. Show that exactly the squares are skipped!

2. The function skips the triangular numbers, (numbers of the form $n(n+1)/2$).

3. The function skips the numbers $(kn-k+1)n+\lfloor(k-1)/4\rfloor$.

4. a) The function skips the numbers $\lfloor(n+k-1)n)/k\rfloor$.

b) Combine the results in 3. and 4 a).

6. stirling-q6.py

7. sum3Cubes-q7.py

Here x, y, z are non-negative integers. To see that numbers of the form $9n\pm4$ are not representable, consider $x^3+y^3+z^3$ modulo 9.

10. Gauss has shown that all positive integers can be represented as a sum of at most three triangular numbers. The following program checks this up to $n = 32767$.

triangle-q10.py

11. In the program mode we find the mode of the increasing sequence a[i] = $\lfloor\sqrt{i}\rfloor$:

mode-q11.py

12. commons-q12.py

13. c) A closed formula for g is $g(n) = \lfloor(n+1)t\rfloor$ with $t = (\sqrt{5}-1)/2$.

gfn-q13.py

14. c) The points $(n, h(n))$ are well approximated by a straight line $y = tx$ through the origin. By substituting $h(n) \approx tn$ into the functional equation for h we get $tn \approx n - t^3(n-1)$ or $t^3+t \approx 1+t^3/n$. For $n \to \infty$ we see that t must satisfy $t^3+t-1=0$. bis.py gives t = 6.8232780381E-1. So $h(n)$ = round(nt) and $h(n)$ = trunc$((n+1)t)$ are possible candidates. Up to $n = 16000$, $h(n)$ = round$(n*t)$ is correct in about 85% of all cases and deviates from the correct value of $h(n)$ by +1 or -1 in the remaining cases. trunc$((n+1)*t)$ deviates from $h(n)$ in over 20% of all cases by +1 or -1. Can you find an exact formula for $h(n)$?

16. sextuple-q16.py

17. eulerReal-q17.py

x	1	1	1	3	1	5	7	3	17	11	23	45	91	89	93	271	85
y	3	1	5	1	11	9	13	31	5	57	67	47	87	275	449	101	999
n	4	3	5	6	7	8	9	10	11	12	13	14	16	17	18	19	20

Table B.2. Output from eulerReal-q17

By running `eulerReal.py` we get the data in Table B.2 that any x is one half of the sum or difference of the two preceding x, y. From the current x, y you get the next y by computing $(7x+y)/2$ or $|7x-y|/2$. If the next x is a sum then the next y is a difference, and vice versa. That is we have the two transformations S,

T:

$$S : (x, y) \to ((x + y)/2, |7x - y|/2), \quad T : (x, y) \to (|x - y|/2, (7x + y)/2)$$

Prove this by induction! You will have to prove in addition that one of the two new pairs consists of odd elements.

21. MinComElem-q21.py

24. floyd-q24.py

29. a) 381654729 b) 3816547290

30. bellTriangle-q30.py

For $k = 0, 1, ..n$, let $B_{n,k}$ be the number of partitions of {1, ..., n} such that the other elements of the set to which n belongs are all $\leq$ k. We have $B_{n,0} = B_{n-1}$ and $B_{n,n-1} = B_n$.

The partitions counted by $B_{n,k+1}$ are of two kinds. First, those in which the element $k + 1$ is not in the same set as n. There are $B_{n,k}$ such partitions. Second, those in which $k + 1$ and n are in the same set. Removing the element $k + 1$ from all of these, we see that the number of these partitions is $B_{n-1,k}$. Thus $B_{n,k+1} = B_{n,k} + B_{n-1,k}$.

32. c) Both are correct.

33. olympiad-q33.py

35. Each number occurs twice except 1, which occurs four times, and the other powers of 2, which occur three times.

B.10 Section 3.10

1. The following generates a random permutation: randPerm-q1.py

B.11 Section 3.11

1. Since the period is $p - 1$, all of $1, 2, ..., p - 1$ occur as remainders if we divide a positive integer which is $< p$ by p. The digits we obtain from these remainders are the integer parts of $10/p, 20/p, ..., 10(p-1)/p$. The number of times we get the digit d is equal to the number of multiples of 10 in the interval $dp \leq x < (d+1)p$. That number is at least $\lfloor p/10 \rfloor$ and at most $\lceil p/10 \rceil$. The proof for pairs, triples etc. is similar.

B.12 Section 3.12

2. The pure periodicity follows from the fact that the stream of digits can be uniquely extended into the past.

3. a) Consider the sequence mod 2: 11011 11011 11011 .... The numbers 1,2,3,4 taken mod 2 are 1,0,1,0. This pattern does not occur.

b) This is 1,0,0,1 mod 2, and so it cannot occur.

c) Extend the sequence one step into the past and you get 519831138....

d) This block will occur again since the sequence is purely periodic.

4. $x(n) = (x(n-3)+x(n-5)) \bmod 2$ and $y(n) = (4y(n-4)+y(n-5)) \bmod 5$ have periods 2^5-1 and 5^5-1. The period of the resulting sequence is the least common multiple of these two numbers. Since they are prime to each other it is their product 31 * 3124 = 96844.

5. The combined sequence has period $(2^{17}-1)(5^7-1) = 10239790804$.

B.13 Section 3.15

1. Suppose that after the first transformation the largest value occurring among the x_i is M and that a smaller positive value also occurs. If only one such value occurs then after the next step there will be two such values next to each other. Let $x_i, x_{i+i}, ..., x_j$ be an interval of maximal length such that it begins and ends with a positive entry < M and none of the entries in between is = M. Then if we apply the transformation once more, the interval $x'_{i-1}, x'_i, ..., x'_j$ will have positive entries < M at both ends and no entry M inside. Thus the length of the longest such interval increases by at least 1 at each step and after at most $n-1$ steps no entry M is left. We see that after at most $n(M-1)$ steps we shall reach a set such that all the positive x_i have the same value.

3. If the polygonal set x consists of 0's and 1's, then Tx also consists of 0's and 1's, and the number of 1's is even. Also, if n is odd then T has an inverse on the set X_0 of x with an even number of 1's. It follows that for odd n all $x \in X_0$ are on cycles.

4. Applying the transformation T twice to a cycle with an even number of entries is the same as applying it once to the cycle consisting of the even numbered entries and once to the cycle consisting of the odd numbered entries. Hence $2c(n)$ is a T-period for any $2n$-component cycle. To see that in some cases the T-period is not shorter than $2c(n)$, use that if the even numbered entries in the original cycle are all 0, the odd numbered entries change only in every second step.

8. a) The following program plays Bulgarian solitaire, starting with any number of stacks of any sizes. It stops when, arranged in increasing order, the i-th stack contains i or $i+1$ cards, and it prints out the stacks in increasing order.

bulsolit-q8.py

B.14 Chapters 1 to 3

3. We get the infinite binary word as follows: $w_1 = 0, w_2 = 001, w_3 = w_2w_2w_1$. By induction we get $w_{k+1} = w_kw_kw_{k-1}$. Let a_k and b_k be the number of zeros and ones in w_k, Then $a_{k+1} = 2a_k + a_{k-1}$, $b_{k+1} = a_k$, $t_k = a_k/a_{k-1}$, $t_{k+1} = a_{k+i}/a_k =$

$2 + 1/t_k$, For $k \to \infty$ we get $t = 2 + 1/t$, or $t = \sqrt{t} + 1$. The ratio a_k/b_k tends to an irrational limit. Thus the sequence is not periodic. If it were periodic, t_k would tend to the rational ratio of zeros to ones in one period. For the infinite binary word we have zeros/ones = $(\sqrt{2} + 1)/1$, zeros/(zeros+ones) = $(\sqrt{2}+1)/(2+\sqrt{2}) = 1/\sqrt{2}$, ones/(zeros+ones) = $1/(2 + \sqrt{2})$. So every $(2 + \sqrt{2})$th digit is a 1. The n-th 1 should have place number $\approx (2 + \sqrt{2})n$. We find empirically that the greatest integer part of the last expression is the place number of the n-th 1. This is tested by the following program. It first constructs n bits of the word. For input u it then finds the place number i of the u-th 1 and compares with f(u) = trunc($(2 + \sqrt{2})$ * u).

BinWord-q3.py

We also give a proof that always i = trunc($(2 + \sqrt{2})n$). Let S be the operator which performs the following substitutions on binary sequences: it replaces each 0 by 001 and each 1 by 0.

Lemma 1. For any sequence D = $d_1...d_n$ of binary digits the proportions of 0's among the digits of D and S(D) are on opposite sides of $\frac{1}{\sqrt{2}}$.

Proof. Let the number of 0's in D be z. Let n' be the number of digits in S(D) and let z' be the number of 0's among them. Then

$$n' = n + 2z, \quad z' = n + z. \tag{B.8}$$

It is easy to check that the ratio $\frac{s}{n}$ would remain unchanged by S if it had the value $\frac{1}{\sqrt{2}}$. Since this is irrational the ratio can not have this value for a finite sequence, but it is instructive to examine how the difference $z - \frac{n}{\sqrt{2}}$ is changed by the substitution S. We find

$$z' - \frac{n'}{\sqrt{2}} = -(\sqrt{2} - 1)(z - \frac{n}{\sqrt{2}}). \tag{B.9}$$

So S not only changes the sign of the difference $z - \frac{n}{\sqrt{2}}$ but also decreases its magnitude by the factor 0.414.... Interestingly, this is the reciprocal of the limit of the factors by which S increases the lengths of sequences as the proportion of 0's approaches the stable value. In the case of the initial segments 0, S(0), S(S(0)), ... of our sequence, the ratios $\frac{n}{z}$ are the continued fraction approximants of $\sqrt{2}$, as one can deduce from the recursion formulas.

Lemma 2. Let $I_n = a_1...a_n$ be an initial segment of our sequence. Let z_n be the number of 0's in the segment. Then $z_n > \frac{n}{\sqrt{2}}$ if $a_n = 0$ and $z_n < \frac{n}{\sqrt{2}}$ if $a_n = 1$.

Roughly speaking, this says that the ratio of digits hovers so close to the invariant ratio that the last digit can tip it in its own direction.

Proof. One can verify the claim for small values of n. Assume $n > 12$ and that Lemma 2 has been proved for $I_1, ..., I_{n-1}$.

If $a_n = 1$ then $I_n = S(I_m)$ for some $m < n$, and the last digit of I_m is 0. Then Lemma 2 follows from the induction hypothesis and Lemma 1. If I_n terminates with a 0, there are two possibilities:

a) $I_n = \mathrm{S}(I_m)$ where I_m ends with 1. Here the induction hypothesis and Lemma 1 again give the required result.

b) The final 0 of I_n is the first or the second 0 of a 001 which comes from substituting for a 0 in a precursor initial segment. For definiteness let it be the first 0. Then $I_n = (\mathrm{S}(I_m))0 = \mathrm{S}(I_n 1)$. Think of the sequence $I_m 1$ as the initial segment up to and including the first 1 beyond a_m, with the 0's beyond a_m deleted. Then we see that the induction hypothesis implies that the proportion of 0's in $I_m 1$ is $< \frac{1}{\sqrt{2}}$ and hence by Lemma 1 the proportion in I_n is $> \frac{1}{\sqrt{2}}$. The case when the final 0 of I_n is the second 0 of a 001 can be dealt with similarly.

We are ready to prove our formula for the number of digits n up to and including the k-th 1. Lemma 2 tells us $k > (1 - \frac{1}{\sqrt{2}})n$. Also, $a_{n+1} = 0$ since every 1 in the sequence is preceded by at least two 0's. Applying Lemma 2 to I_{n+1} we get $k < (1 - \frac{1}{\sqrt{2}})(n+1)$. We can write these two inequalities as $n < k(2+\sqrt{2}) < n+1$, q.e.d.

For a lattice point interpretation of this problem, see the section on continued fractions in Felix Klein's *Elementary Mathematics from an Advanced Standpoint* [**Kle12**].

7. equisum-q7.py

9. $n = 2053$.

10. die100-q10.py

B.15 Sections 4.5 to 4.6

1. Represent a nonincreasing sequence of at most m numbers, each of which is $\leq n$, by columns of unit squares with bases on the x-axis. If we look at such a figure sideways, it becomes a representation of a nonincreasing sequence of at most n numbers, each of which is $\leq m$.

2. TwoSamp-q2.py

5. TwoSamit-q6.py

B.16 Section 4.13

1. TwoCompleteSets-q1.py

4. CoverMultCase-q4.py

5. `CoverHypgCase.py` is obtained from CoverMultCase-q4.py by setting up a list of occupied squares and rejecting a random choice if the square is already occupied.

B.17 Section 4.14

tripleBirthday-q2.py

Ten repetitions of this program gave 88.77 for the expected waiting time.

B.18 Chapters 3 to 4

1. craps-q1.py

2. CoinGame-q2.py

3. $E(D_n^2) = n$.

4. $E(D_n^2) = n$.

5. $E(D_n^2) = n$.

randomWalk-q5.py

6. About 0.495.

lotto-q6.py

7. $E(D_n^2) = n$.

8. randDirWalk-q8.py performs a random walk of n steps m times and finds the mean of the squares of the distances from the origin at the end of each walk. Choose $m = 1000$ and $n = 10, 20, 50, 100$ and you will realize that the expectation of the square of the distance is n. Our interpretation of random choice of a point on a unit sphere S is this: it falls into a subset M of S with probability area(M)/4π. We use the theorem of Archimedes that a spherical zone of height h on a sphere of radius 1 has area $2\pi h$. We choose h at random from (-1, 1). The geographical longitude a of the point is chosen by means of a = 2π*random(). If we set $r = \sqrt{1 - h * h}$, then $x = x + r * \cos(a), y = y + r * \sin(a), z = z + h$.

We find that $E(D_n^2) = n$.

10. PokerTest-q10.py

For $n = 10000$ we get 3069 5063 1046 679 95 47 1.

15. a) palindrome-q15.py generates m n-words from the alphabet {0, 1, 2} and counts those which start with a palindrome. The proportion of such words is an estimate for P_n. Table B.3 can be used for checking the program:

16. totalTourCube-q16.py

The exact value is 1996/95 = 21.0105.

B.19 Chapters 1 to 4

2. The formula is $f(a, b, c) = 3abc/(a + b + c)$. Lennart Råade of Sweden put this problem to a number of people in the last 20 years, but nobody could solve it. In 1992, I (Engel) ran a version of ThreeTowerGame-q2.py, and its output suggested the formula which I then proved. (See the end of Appendix A.)

n	Pn
2	1/3
3	5/9
4	$17/27 = 0.63$
5	$55/81 = 0.679$
6	$169/243 = 0.6955$
7	$517/729 = 0.70919$
8	$1561/2187 = 0.71376$
9	$4709/6561 = 0.7177$
10	$14153/19683 = 0.71905$
11	0.7203
12	0.72072
14	0.72125
16	0.72142
18	0.721481581

Table B.3. Output from palindrome-q15.py.

B.20 Section 5.4

2. We assume the random sets are chosen by flipping a fair coin to decide for each element whether it should be in the set. After the first guess, G is in a set containing r other numbers which were picked at random from the set of $n-1$ numbers other than G, etc. The guessing ends when $r = 0$.

RandGuess-q2.py

B.21 Section 5.5.5

1. knights-q1.py

4. The following program tries to find a knight's path visiting every square of an $(m-2)$ x $(n-2)$ chessboard, using Warnsdorf's method. Warnsdroff-q4.py

The list b[i][j] will contain the ordinal numbers of the squares in the path. The list is bordered with two dummy rows on all 4 sides. This enables us to list the squares accessible by a knight jump from a given square (of the actual chessboard) in the same way, whereas otherwise we would have to do it differently for squares near or on the edge. We initialize b[i][j] to be 0 on the chessboard proper and 1 on the border rows. This will cause the algorithm not to put squares of the border rows into the knight's path.

At each stage the algorithm starts from the current position (i, j). It checks all squares reachable by changing the coordinates by at most 2. If s, t are the amounts added to the coordinates, the condition for reachability by knight jump has the simple form $|st| = 2$. For each square reachable from (i, j), we check how

many squares reachable from it are still free. We select as the next square the one with coordinates `imin`, `jmin`, which is a free square reachable from (i, j) and has fewer free neighbors than any of the squares previously tried, and not more than any tried after it. In the program, c becomes the number of free neighbors of a square, and `min` becomes the least value of c among the neighbors of (i, j).

The construction terminates once we have selected a free square with no free neighbors. If this happens before $m \cdot n$ squares have been covered, the algorithm has failed.

Note that if the current square has several free neighbors with `min` free neighbors of their own, choosing the first one our algorithm finds is arbitrary; one could make these choices at random. Then a second run may succeed if the first one fails.

A related approach tries to find a closed knight's tour by using Warnsdorf's selection criterion, and backtracking when it gets stuck.

B.22 Section 5.6

3. savinqBoxes-q3.py

By simulation we find $p(n, k) = k/n$. Here is one of the proofs from the report on the competition in the Hungarian high school periodical *Matematikai Lapok*.

We ignore k for the time being. We consider the following procedure for opening all boxes. Break box 1. Open as many boxes with keys as you can. The last of those has key 1 in it. Now break the lowest numbered box which is still locked, open all the boxes you can, etc.

Let us call the order in which the keys appear in this procedure the *retrieval permutation* P' corresponding to the *hiding permutation* P. Given a retrieval permutation P', we can reconstruct the hiding permutation P which produces it. Hence every permutation of keys is a possible retrieval permutation. If all permutations are equally likely to occur as hiding permutations then they are also equally likely to occur as retrieval permutations.

We note that the key in the last place in a retrieval permutation is the key of the last box which was broken. Let the number of that key be l. When we broke box l, all boxes with smaller numbers were already open, so to open all boxes we needed to break one whose number was $\geq l$.

Let us return now to the opening stage which consists of breaking the first k boxes at the start and not breaking any more. We can open all boxes if and only if the last entry l of the retrieval permutation is one of $1, 2, ..., k$. The probability of this is k/n.

6. The following does the computations for a), b), c) and f).

GoldPerm2-q6.py

B.23 Section 5.8

2. c) Let

$$x(n) = \lfloor nt \rfloor, \quad y(n) = n + \lfloor nt \rfloor = \lfloor n(1+t) \rfloor = \left\lfloor nt^2 \right\rfloor$$

We need to show only that the sequences $x(n)$ and $y(n)$ together contain every positive integer exactly once. This is an instance of S. Beatty's theorem. One way of proving it is as follows. For a given positive integer n, the number of terms $< n$ in the sequence $t, 2t, 3t, ...$ is $\lfloor n/t \rfloor$. The number of terms $< n$ in $t^2, 2t^2, 3t^2, ...$ is $\lfloor n/t^2 \rfloor$. Since t is irrational, n/t and n/t^2 are both irrational, but $n/t + n/t^2 = n(1+t)/t^2 = n$. Thus

$$\lfloor n/t \rfloor + \left\lfloor n/t^2 \right\rfloor = n - 1.$$

This is the number of terms in the two sequences taken together. By taking $n = 1, 2, 3, ...$ we see that there is exactly one multiple of either t or t^2 in each of the intervals $(1, 2), (2, 3), ...$ Hence every positive integer is in exactly one of the sequences $\lfloor nt \rfloor, \lfloor nt^2 \rfloor$.

4. L consists of all multiples of $k + 1$ including 0.

5. L consists of all multiples of 3 including 0.

6. L consists of all multiples of 4 including 0.

12. L consists of all multiples of 6 including 0.

B.24 Section 6.2

13, 14. These computations are much easier in languages which support integer arithmetic with arbitrarily large integers.

B.25 Section 7.6

1. (a) The transition probabilities for this Markov chain with three states are as follows:

```
               POOR SATISFACTORY PREFERRED
POOR            0.6    0.4          0
SATIISFACTORY   0.1    0.6         0.3
PREFERRED        0     0.2         0.8
```

so that the transition probability matrix is

```
    0.6  0.4   0
P = 0.1  0.6  0.3
     0   0.2  0.8
```

(b) We will find the limiting fraction of drivers in each of these categories from the components of the stationary distribution vector π, which satisfies the

following equation:

$$\pi = \pi P.$$

The former is equivalent to the following system of linear equations:

$$\begin{aligned} \pi_1 &= 0.6\pi_1 + 0.1\pi_2 \\ \pi_2 &= 0.4\pi_1 + 0.6\pi_2 + 0.2\pi_3 \\ \pi_3 &= 0.3\pi_2 + 0.8\pi_3 \\ 1 &= \pi(1) + \pi_2 + \pi_3. \end{aligned}$$

This has the following solution: $\pi = (1/11, 4/11, 6/11)$.

Thus, the limiting fraction of drivers in the POOR category is 1/11, in the SATISFACTORY category 4/11, and in the PREFERRED category 6/11. By the way, the proportions of the drivers in each category in 15 years approximate these numbers with two significant digits (you can check it, calculating P^{20} and looking at its rows).

Use `steadyTests.py` and `powersTest.py` on `cars.txt`.

3. The transition matrix is

```
       GO   A   B   C
   GO  1/6  1/3 1/3 1/6
P=  A  1/6  1/6 1/3 1/3
    B  1/3  1/6 1/6 1/3
    C  1/3  1/3 1/6 1/6
```

$\pi = (1/4, 1/4, 1/4, 1/4)$ using `steadyTests.py` on `monopoly.txt`.

From this we see that $\pi \cdot f = 0$ where

```
    15
f= -30
    -5
    20
```

4. We use a Markov chain model with 3 states, H, M, and E, where the state reflects the difficulty of the most recent exam. We are given the transition probabilities:

```
    H     M     E
H   0    0.5   0.5
M  0.25  0.5   0.25
E  0.25  0.25  0.5
```

It is easy to see that our Markov chain has a single recurrent class, which is aperiodic. The balance equations take the form

$$\begin{aligned} \pi_1 &= 0.25\pi_2 + 0.25\pi_3 \\ \pi_2 &= 0.5\pi_1 + 0.5\pi_2 + 0.25\pi_3 \\ \pi_3 &= 0.5\pi_1 + 0.25\pi_2 + 0.5\pi_3 \end{aligned}$$

and solving these with the constraint $\sum_i \pi_i = 1$ gives

$$\pi_1 = \frac{1}{5} \quad \pi_2 = \pi_3 = \frac{2}{5}$$

or use steadyTests.py on exams.txt.

5. Snakes and Ladders.

Run absorbSnakes.py.

Expected time to absorption from square 5 is 8.36

6. It's easy to see that the Markov chain described by the transition diagram in Fig. B.1 captures exactly what we are looking for.

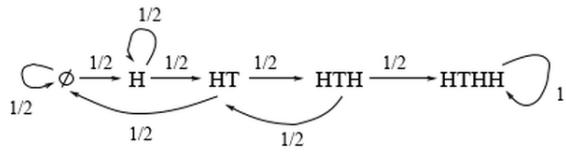

Figure B.1. Reaching HTHH

Rename the states 0, H, HT, HTH, HTHH as 0, 1, 2, 3, 4 respectively. and let v_i be the average number of steps required for the state 4 to be reached if the starting state is i. Writing first-step (backwards) equations we have

$$\begin{aligned} v_0 &= 1 + 0.5v_0 + 0.5v_1 \\ v_1 &= 1 + 0.5v_1 + 0.5v_2 \\ v_2 &= 1 + 0.5v_0 + 0.5v_3 \\ v_3 &= 1 + 0.5v_3 + 0.5v_4 \end{aligned}$$

Also. obviously, $v_4 = 0$. Solving, we find

$$v_3 = 8, \quad v_2 = 14, \quad v_1 = 16, \quad v_0 = 18.$$

So the answer is: "it takes on average 18 coin tosses to see the pattern HTHH for the first time".

Or run absorbFair.py. with fair.txt. and look at t vector

7. Say that a score $x - y$ means that the server has x points and the other player y. If the current score is $3 - 3$ the next score is either $4 - 3$ or $3 - 4$. In either case, the game must continue until one of the players is ahead by 2 points. So let us say that i represents the difference $x - y$. We model the situation by the Markov chain in Fig. B.2.

Let v_1 be the probability that the server wins, i.e. that state 2 is reached before state -2. First-step equations yield:

$$v_i = 0.6v_{i+1} + 0.4v_{i-1}, \quad -1 \le x \le 1$$

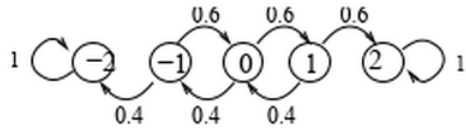

Figure B.2. Playing Tennis

In other words,

$$\begin{aligned} v_{-1} &= 0.6v_0 + 0.4v_{-2} \\ v_0 &= 0.6v_1 + 0.4v_{-1} \\ v_1 &= 0.6v_2 + 0.4v_0 \end{aligned}$$

Of course,

$$v_{-2} = 0, \quad v_2 = 1$$

Solving, we find

$$v_0 = \frac{0.6^2}{1 - 2 \times 0.6 \times 0.4} \approx 0.69, \quad v_1 \approx 0.88, \quad v_{-1} \approx 0.42$$

Or run absorbTests.py on tennis.txt. and look at second column of B

8. (a)

```
        1    2    0    3
   1    0   2/3  1/3   0
P = 2   2/3  0    0    1/3
   0    0    0    1    0
   3    0    0    0    1
```

(b)

```
        1     2
N = 1 9/5    6/5
   2 6/5    9/5

        0     3
B = 1 3/5   2/5
   2 2/5   3/5

t = 1  3
   2  3
```

Obtained by running absorbTests.py on spinner.txt.

(c) The game will last on the average 3 moves.

(d) If Mary deals, the probability that John wins the game is 3/5.

9. The problem can be solved by writing first step equations for the Markov chain representing the number of red cards remaining.

The transition probabilities are:

$$p(3,2) = 3/6, \quad p(2,1) = 2/5, \quad p(1,0) = 1/4,$$
$$p(i,i) = 1 - p(i,i-1), \quad i = 3,2,1,$$
$$p(0,0) = 1$$

Let v_i be the average number of steps till the process ends if the initial state is i. Then

$$\begin{aligned} v_3 &= 1 + 3/6v_3 + 3/6v_2 \\ v_2 &= 1 + 3/5v_2 + 2/5v_1 \\ v_1 &= 1 + 3/4v_1 \end{aligned}$$

Alternatively run absorbReds.py on reds.txt to get $v_3 = 8.5$.

B.26 Chapters 1 to 7

1. Let the number of people who have heard the joke be `inf` and the number of people who have told it to someone who heard it `em`. (`inf` stands for infected, to indicate similarity to models of epidemics, and `em` stands for embarrassed, for having told a familiar joke.)

The next person to be told could be one who heard it already, other than the person telling it, or someone new. We model this by taking an integer at random from 2..n. If the number is $\leq$ `inf` we add 1 to `em`, otherwise we add 1 to `inf`. When `em` = `inf` the process stops and we record how many people have not heard the joke. The experiment is repeated m times. 11 repetitions of Joke-q1.py gave the median output 0.2035 and mean output 0.2032. That is, about 20% of the population does not hear the joke.

If n is large and we set `inf/n` = x and `em/n` = y then the two quantities grow approximately in accordance with the differential equation $dy/dx = z/(1-z)$, $y(0) = 0$, until y becomes equal to x, which happens when $2x = \ln(1-x)$. Then $1-x$, the proportion of people who did not hear the joke, is 0.2031878699799799...

2. You will get all numbers of the form $2^m 3^n - 1$.

5. The following program checks Watson's claim with interpretation a):

checkit-q5.py

We have 33/70=0.47142857.... On the other hand, five runs of with max=10000 resulted in 0.4564, 0.4549, 0.4637, 0.4600, 0.4564, which is close to, but consistently below 0.47. So this does not seem to be the right interpretation.

By trying interpretation b) we get 0.472. This seems to be the correct interpretation. One can get the exact value by algebraic computation. One can show that the required probability is the same as for the following simpler problem: let S be a sphere with north pole N. Pick two points A, B inside S at random. Find the probability that $\bigtriangleup NAB$ is acute. If one takes the north pole as the origin of polar coordinates, evaluating the triple integrals representing the probability of

getting an obtuse angle at the north pole, or at another vertex, becomes a laborious exercise in third-term calculus.

Restricting the random points to a cube or a sphere is arbitrary. The following argument seems to reduce the completely unrestricted problem to the simplified problem above.

Let A be that one of the three points chosen which is farthest from the origin 0. Then all we know about B and C is that they randomly chosen points inside the sphere about O with radius OA. Thus the probability that $\bigtriangleup ABC$ is acute is the same as the probability that a triangle formed by the North Pole of a sphere and two points chosen at random inside it is acute.

The weakness of this argument is that it tacitly uses the notion of selecting a point at random from an infinite space, without giving preference to any part of the space. Assuming that this can be done leads to contradictions. For example, take the two dimensional case of our problem. Let A be the first point we pick, B the second, C the third. Consider the probability P that the angle at C is $> 45°$. This will happen if C is in the union of the interiors of the two circles through A and B with radii $AB/\sqrt{2}$. We can place infinitely many disjoint copies of this region in the plane. The probability that C will be in any one of these ought to be P also. The sum of these probabilities must be < 1, hence P can not be positive, it must be 0. Thus the probability that the angle at C is $> 45°$ is 0. But there is nothing special about the last point C in our problem. The three points could be selected by three people simultaneously, and they could tell us in random order. So the probability that a random triangle has any angle $> 45°$ is 0, which is absurd.

6. The following program first computes the list f[1..n]. Then for input q, it prints out $f(q)$.

selfDescrSeq-q6.py

We get $c * 10000^d \approx 356$, $c * 5000^d \approx 232$. Hence $2^d \approx 356/232 = 89/58$, $d \approx \ln(89/58)/\ln(2) = 0.61775...$, $c \approx 356/10000^d = 1.23035....$ So $f(n) \sim 1.23035n^{0.61775}$. Check $f(n) = \lfloor cn^d + 0.5 \rfloor$ with the correct value of $f(n)$. Can you do better?

8. a) mergeSortComp-q8.py

b) $\text{com(n)} = n\lceil \log_2 n \rceil - 2^{\lceil \log_2 n \rceil} + 1$

11. ThreePlayers-q11.py

Simulation did not help with four players.

12. intertwinedFunctions-q12.py

If you plot the functions $F(n)$ and $M(n)$ you will observe that both are strongly clustering about the same straight line through the origin: $F(n) \sim tn, M(n) \sim tn$. For t we get $tn = n - t^2(n-1)$, or, for large n, $t^2 + t = 1$ with the solution $t = (\sqrt{5}-1)/2$. The program finds $F(i), M(i), \lfloor it + 0.5 \rfloor, \lfloor (i+1)t \rfloor$ from k to n. The agreement is exact with a few exceptions, when the difference between any pair is at most 1.

14. a) $g(a,b,c) = ab + bc + ca$;
piles-q14a.py
b) $g(a,b,c,d) = ab + ac + ad + bc + bd + cd$;
c) $g(x_1, ..., x_n) = \sum_{i<k} x_i x_k$
15. isPoly-q15.py
19. trans-q19.py
20. The point (x, y) can be reached from $(1, 1)$ if and only if gcd(x,y) $= 2^n$, $n = 0, 1, 2, ...$
22. alg-q22.py
24. armstrong-q24.py
26. twinperm-q26.py
26. b) p[10000] $= 0.36787024156 \approx e^{-1} = 0.36787944117....$ This is also the probability of getting a random permutation of {1,..., n} without a fixed point. See Ex. 3 of this section. But the convergence to the asymptotic value e^{-1} is slower.
27. For $z = 26861$ we have $\pi_1(x) = 1473$, $\pi_3(x) = 1472$
28. c) (1) is exact for $n \geq 1 + s(b-1)$, where $2s$ is the word length and b is the base.
32. pentagon-q32.py picks the negative terms to be inverted in random order.

We find that the algorithm always stops and the number of steps is $20n - 10$ no matter how we pick the next operation, although the sets of numbers which come up do depend on that.

The key to the proof is to find an integer-valued, non-negative function f$(x_1, x_2, x_3, x_4, x_5)$ of the vertex labels whose value decreases when the given operation is performed. All but one of the students who solved the problem found the same function,

$$f(x_1, x_2, x_3, x_4, x_5) = \sum_{i=1}^{5} (x_i - x_{i+2})^2, \quad \text{where } x_6 = x_1,\ x_7 = x_2.$$

If, say, x_4 is flipped, $f_{\text{new}} - f_{\text{old}} = 2sx_4 < 0$. A decreasing sequence of non-negative integers must be finite, so the algorithm must stop.

Bernhard Chazelle (Princeton) gave an argument which shows why the number of steps is determined by the initial values. He considers the infinite multiset S of all sums $s(i,j) = x_i + \cdots + x_j$ with $1 \leq i \leq 5$ and $i \leq j$, where $x_6 = x_1$, etc. A multiset is a set which can have equal elements. In this multiset one element changes from negative to positive and all the others are either unchanged or switched with others when we perform an operation. For instance, if x_4 is changed from negative to positive, and the new values are denoted by primes, $s'(4,4) = -s(4,4)$ changes from negative to positive whereas $s'(4,5) = s(5,5)$, $s'(5,5) = s(4,5)$, $s'(4,6) = s(5,6)$, $s'(5,6) = s(4,6)$, etc. There are only finitely

many negative elements in S, since $s > 0$. The number of steps until the finish is the number of negative elements of S. The x_i need not be integers for this argument to be valid.