Assignment 1 Solutions

Experimental Mathematics 2020

Exercise 1. This appears in the second volume (Section 4.5.3, Exercise 39) of Donald Knuth's masterpiece *The Art of Computer Programming*.

There is a brute-force way to do this:

```
def batting_average(p, q):
    return (1000*p/q).round()

def smallest_q(avg, start_at=0, end_before=1000):
    for q in range(start_at, end_before):
        for p in range(q):
            if batting_average(p, q) == avg:
                return p/q

sage: smallest_q(334)
96/287
sage: smallest_q(334, start_at=288)
97/290
```

So the smallest number of times at bat is 287, and the second smallest is 290.

Alternatively, a more refined approach is looking for the fraction

$$.3335 \leqslant \frac{p}{q} < .3345$$

with the smallest possible q.

This can be done with continued fractions as follows:

```
def minimal_denominator(a, b):
    """
    Given positive real numbers a and b, return the fraction a <= p/q < b
    with smallest possible denominator q.
    """
    ca = continued_fraction(a)
    cb = continued_fraction(b)
    ell = min([len(ca), len(cb)])

    j = 0
    while (j <= ell) and (ca[j] == cb[j]):
        j += 1

    if ca[j] < cb[j]:
        c = ca[j] + 1
```

```
    else:
        c = cb[j] + 1

    cc = continued_fraction(list(ca)[:j] + [c])
    return cc.value()

sage: minimal_denominator(.3335, .3345)
96/287
```

To get the second smallest denominator, we need to replace the +1 by +2 in two places:

```
    if ca[j] < cb[j]:
        c = ca[j] + 2
    else:
        c = cb[j] + 2
```

□

Exercise 2. Here's the experiment with default precision:

```
def ex2alpha(j, N):
    res = sum([1/RR(16^k*(8*k+j)) for k in range(N+1)])
    return res

sage: xlst = [ex2alpha(j, 100) for j in range(1, 8)]
sage: xlst
[1.00718447641468,
 0.506476876667430,
 0.339230245245199,
 0.255412811882995,
 0.205002557636424,
 0.171317070666497,
 0.147201934672635]
sage: xlst.append(RR(pi))
sage: intrel(xlst, 10^6)
(-4, 0, 0, 2, 1, 1, 0, 1)
sage: intrel(xlst, 10^8)
(-4, 0, 0, 2, 1, 1, 0, 1)
sage: intrel(xlst, 10^10)
(-4, 0, 0, 2, 1, 1, 0, 1)
```

The result persists if we increase the precision, for instance using `R = RealField(1000)`. The experiment suggests that

$$\pi = \sum_{k=0}^{\infty} \frac{4}{16^k(8k+1)} - \sum_{k=0}^{\infty} \frac{2}{16^k(8k+4)} - \sum_{k=0}^{\infty} \frac{1}{16^k(8k+5)} - \sum_{k=0}^{\infty} \frac{1}{16^k(8k+6)}$$

This is known as the BBP (Bailey–Borwein–Plouffe) formula for π and was discovered in 1995. □

Exercise 3. The idea is of course to use the logarithm function to translate the multiplicative relation into a standard integer relation:

$$m_1 \log(\alpha_1) + m_2 \log(\alpha_2) + \cdots + m_n \log(\alpha_n) = 0$$

We apply this in the setting required in the question:

```
sage: xlst = [RR(log(p)) for p in prime_range(18)]
sage: xlst.append(RR(log(pi)))
sage: xlst.append(RR(log(zeta(14))))
sage: intrel(xlst, 10^6)
(0, 1, 1, -1, 3, 0, -2, -2, 1)
sage: intrel(xlst, 10^8)
(1, 0, -2, 5, 4, -3, -2, -3, 1)
sage: intrel(xlst, 10^9)
(-6, 0, 3, -1, -4, 6, -2, 1, 1)
sage: intrel(xlst, 10^10)
(-6, 0, -4, 1, 1, -4, 3, 7, -2)
sage: intrel(xlst, 10^11)
(-1, 6, 2, 1, 1, 1, 0, -14, 1)
sage: intrel(xlst, 10^12)
(-1, 6, 2, 1, 1, 1, 0, -14, 1)
sage: intrel(xlst, 10^13)
(-1, 6, 2, 1, 1, 1, 0, -14, 1)
sage: intrel(xlst, 10^14)
(-1, 6, 2, 1, 1, 1, 0, -14, 1)
sage: intrel(xlst, 10^15)
(-1, 6, 2, 1, 1, 1, 0, -14, 1)
```

So we appear to have settled on the integer relation

$$-\log(2) + 6\log(3) + 2\log(5) + \log(7) + \log(11) + \log(13) - 14\log(\pi) + \log(\zeta(14)) = 0$$

or, written multiplicatively,

$$\frac{3^6 \cdot 5^2 \cdot 7 \cdot 11 \cdot 13 \cdot \zeta(14)}{2\pi^{14}} = 1$$

This is starting to look very familiar, especially if we rewrite it as

$$\frac{\zeta(14)}{\pi^{14}} = \frac{2}{3^6 \cdot 5^2 \cdot 7 \cdot 11 \cdot 13} = \frac{2}{18243225}$$

Trust, but verify:

```
sage: alpha = zeta(14.0)/RR(pi)^14
```

The ugliness (compared to `zeta(14)/pi^14`) is needed to work around Sage being too clever; we don't want to be using what we are trying to verify.

```
sage: alpha
1.09629739259369e-7
sage: continued_fraction(alpha)
[0; 9121612, 2, 41146564]
```

The huge denominator following the 2 indicates a possible roundoff error (which we could investigate further by increasing the working precision), so we stop short of it:

```
sage: continued_fraction(alpha).convergents()[-2]
2/18243225
sage: continued_fraction(alpha).convergents()[-2].factor()
2 * 3^-6 * 5^-2 * 7^-1 * 11^-1 * 13^-1
```

□

Exercise 4. Here is one possible implementation:

```
IntRel[x_, A_] :=
 Drop[LatticeReduce[
    Transpose[
     Append[IdentityMatrix[Length[x]],
      Table[Round[A*x[[j]]], {j, 1, Length[x]}]]]][[1]], -1]

IntRel[{ArcTan[1], ArcTan[1/5], ArcTan[1/239]}, 10^6]

{1, -4, 1}

FindIntegerNullVector[{ArcTan[1], ArcTan[1/5], ArcTan[1/239]}]

{1, -4, 1}
```

□

Exercise 5. Using the syntax from the documentation of `mpmath`:

```
sage: from mpmath import *
sage: mp.dps = 15; mp.pretty = True
sage: for a in range(2, 1001):
....:     for b in range(a, 1001):
....:         res = pslq([pi, acot(a), acot(b)])
....:         if res is not None and res[0] != 0:
....:             print(a, b, res)
2 3 [1, -4, -4]
2 7 [1, -8, 4]
3 7 [1, -8, -4]
5 239 [1, -16, 4]
139 688 [1, -527, 447]
245 981 [1, -715, -219]
```

But... we should always ask ourselves whether what the computer gives us is accurate. For this question, we could do this by increasing the `mpmath` precision, and/or by using LLL to independently verify the results. Since the entire search with higher precision would take somewhat long, we can decide to verify only the candidate relations found above:

```
sage: mp.dps = 20
sage: for (a, b) in [(2, 3), (2, 7), (3, 7), (5, 239), (139, 688), (245, 981)]:
....:     res = pslq([pi, acot(a), acot(b)])
....:     print(a, b, res)
2 3 [1, -4, -4]
2 7 [1, -8, 4]
3 7 [1, -8, -4]
```

```
5 239 [1, -16, 4]
139 688 None
245 981 None
```

Aha, so the last two were just noise. Further experiments with the precision will indicate that the other four relations seem to be genuine, so we get:

$$\pi = 4\,\text{arccot}(2) + 4\,\text{arccot}(3)$$
$$\pi = 8\,\text{arccot}(2) - 4\,\text{arccot}(7)$$
$$\pi = 8\,\text{arccot}(3) + 4\,\text{arccot}(7)$$
$$\pi = 16\,\text{arccot}(5) - 4\,\text{arccot}(239)$$

(One can prove that these are the only Machin-type formulas of this particular form.) □

Exercise 6. I had a lot of fun reading through what you came up with here.

The aim was open-ended exploration of something that does not really have (as far as I know) a fully satisfactory "nice" and complete answer (something like: 42).

Here are the culprits:

```
def ex6alpha(n):
    if n == 1:
        return 1 + sqrt(2)
    return 1 + sqrt(ex6alpha(n-1))

sage: ex6alpha(1)
sqrt(2) + 1
sage: ex6alpha(2)
sqrt(sqrt(2) + 1) + 1
sage: ex6alpha(3)
sqrt(sqrt(sqrt(2) + 1) + 1) + 1
sage: ex6alpha(4)
sqrt(sqrt(sqrt(sqrt(2) + 1) + 1) + 1) + 1
```

Now we can go the LLL way (which will get us the first few polynomials, if we are careful). We could also just ask Sage for the minimal polynomial:

```
sage: ex6alpha(1).minpoly()
x^2 - 2*x - 1
sage: ex6alpha(2).minpoly()
x^4 - 4*x^3 + 4*x^2 - 2
sage: ex6alpha(3).minpoly()
x^8 - 8*x^7 + 24*x^6 - 32*x^5 + 14*x^4 + 8*x^3 - 8*x^2 - 1
sage: ex6alpha(4).minpoly()
x^16 - 16*x^15 + 112*x^14 - 448*x^13 + 1116*x^12 - 1744*x^11 + 1552*x^10 - 384*x^9
 - 700*x^8 + 736*x^7 - 160*x^6 - 128*x^5 + 64*x^4 - 2
sage: ex6alpha(5).minpoly()
x^32 - 32*x^31 + 480*x^30 - 4480*x^29 + 29112*x^28 - 139552*x^27 + 509600*x^26
 - 1441024*x^25 + 3166616*x^24 - 5345344*x^23 + 6668992*x^22 - 5473536*x^21
 + 1494624*x^20 + 3005056*x^19 - 4820608*x^18 + 3037184*x^17 + 17422*x^16
 - 1528032*x^15 + 1062432*x^14 - 104576*x^13 - 254648*x^12 + 138656*x^11
 - 7200*x^10 - 15616*x^9 + 5496*x^8 - 1472*x^7 + 320*x^6 + 256*x^5 - 128*x^4 - 1
sage: ex6alpha(6).minpoly()
```

```
x^64 - 64*x^63 + 1984*x^62 - 39680*x^61 + 575344*x^60 - 6443072*x^59
 + 57968448*x^58 - 430309888*x^57 + 2685669232*x^56 - 14288028800*x^55
 + 65452677504*x^54 - 260075751936*x^53 + 900910582592*x^52 - 2728832570624*x^51
 + 7234443234560*x^50 - 16764539801600*x^49 + 33812992871516*x^48
 - 58848371601728*x^47 + 86960795528384*x^46 - 105685514369792*x^45
 + 98051282625712*x^44 - 53067489947712*x^43 - 21007808658112*x^42
 + 92582655379968*x^41 - 121819299884272*x^40 + 89311309437312*x^39
 - 15328304678016*x^38 - 51609241549312*x^37 + 71545318517632*x^36
 - 43166742440448*x^35 - 679835956736*x^34 + 25508347363328*x^33
 - 22322048910012*x^32 + 6094442977152*x^31 + 5264750043008*x^30
 - 6305324475904*x^29 + 2290553846240*x^28 + 697575188352*x^27
 - 1114394206592*x^26 + 421011123200*x^25 + 57590707552*x^24 - 120433993984*x^23
 + 45135577856*x^22 - 67318784*x^21 - 6876739200*x^20 + 3366873600*x^19
 - 731986432*x^18 - 165322752*x^17 + 207753280*x^16 - 65356800*x^15
 - 467968*x^14 + 5869568*x^13 - 2058240*x^12 + 540672*x^11 - 16384*x^10
 - 65536*x^9 + 16384*x^8 - 2
```

I'll stop there, as this is enough to notice a few things:

- the degree of p_n is 2^n;
- the leading coefficient of p_n is 1 (i.e. α_n is an algebraic integer);
- the constant coefficient of p_n is -1 if n is odd and -2 if n is even;
- the coefficient of x^{2^n-1} in p_n is -2^n;
- there's also a nice closed form formula for the following coefficient, I invite you to look it up on OEIS;
- at the other end of the polynomial, there a sparsity that can be quantified (a lot of coefficients are zero in the terms of small degree);
- other interesting things that some of you observed and I hadn't thought of;
- possibly most crucial: the coefficients are getting big and complicated; it's a mess!

Considering the last point, one remaining hope for a simple description of the polynomials is to find a recursive formula for them. It turns out there are two of them:

$$p_n(x) = p_{n-1}\left((x-1)^2\right)$$
$$p_n = p_{n-1}^2 + 2p_{n-1} - 1$$

This is kind of cool. We can use this to generate a few more of the polynomials (but not many of them, obviously, since the mere size of the polynomials is exponential in n):

```
sage: R.<x> = ZZ[]
def pol(n):
    if n == 1:
        return x^2 - 2*x - 1
    return pol(n-1).subs({x: (x-1)^2})
```

□