Two Logicians

Sunaina Pai

Problem

Two perfect logicians, S and P, are told that integers x and y have been chosen such that $1 < x < y$ and $x + y < 100$. S is given the value $x + y$ and P is given the value xy. They then have the following conversation.

P: I cannot determine the two numbers.
S: I knew that.
P: Now I can determine them.
S: So can I.

Given that the above statements are true, what are the two numbers? (Computer assistance allowed.)

Source: http://www.qbyte.org/puzzles/puzzle01.html#p3

Solution

This problem is solved using a computer program written in Python. The program and its output, respectively, are available at the following URLs:

- https://github.com/sunainapai/lab/blob/master/math/nick/003.py
- https://github.com/sunainapai/lab/blob/master/math/nick/003.txt

The program and the output are also included in the next two sections.

The output shows that the solution is

$$x = 4$$
$$y = 13$$

Program Code

```python
#!/usr/bin/env python3

def factors(p):
    for x in range(2, int(p ** 0.5) + 1):
        if p % x == 0:
            y = p // x
            if x != y and x + y < 100:
                yield x, y


def partitions(s):
    for x in range(2, s // 2 + 1):
        y = s - x
        if x != y and x + y < 100:
            yield x, y


# P: I cannot determine the two numbers.
#
# P's statement implies that P has a product p that can definitely be
# factorized in two or more ways. If p can be factorized in exactly one
# way, then P can easily determine the two factors x and y.
def candidate_product_1(p):
    return len(list(factors(p))) > 1


# S: I knew that.
#
# S's statement implies that S has a sum s such that when we partition s
# into all possible pairs, the product of the two numbers in each pair
# is a candidate for product p that P has.
def candidate_sum_1(s, candidate_products):
    return all(x * y in candidate_products for x, y in partitions(s))


# P: Now I can determine them.
#
# P's statement implies that P has a product p such that when we
# factorize p into all possible pairs, there is exactly one pair whose
# sum is a candidate for sum s that S has.
#
# At this stage, P has already determined the factors x and y of p. But
# we, as observers, still do not know x and y.
def candidate_product_2(p, candidate_sums):
    factorizations = [(x, y) for x, y in factors(p)
                      if x + y in candidate_sums]
    return len(factorizations) == 1


# S: So can I.
#
```

```
# S's statement implies that S has a sum s such that when we partition s
# into all possible pairs, there is exactly one pair whose product is a
# candidate for product p that P has.
#
# At this stage, S has determined the summands x and y of s. But we need
# to run a few more calculations to arrive at x and y.
def candidate_sum_2(s, candidate_products):
    partitionings = [(x, y) for x, y in partitions(s)
                     if x * y in candidate_products]
    return len(partitionings) == 1


# Now we can filter the list of candidate products p further by
# exploiting the constraint that when we factorize a candidate product p
# into all possible pairs, there is exactly one pair whose sum is s.
def candidate_product_3(p, s):
    factorizations = [(x, y) for x, y in factors(p) if x + y == s]
    if len(factorizations) == 1:
        x, y = factorizations[0]
        return x, y
    else:
        return None


def main():
    cp1 = [p for p in range(2 * 3, 49 * 50 + 1) if candidate_product_1(p)]
    print('cp1:', cp1)
    print()

    cs1 = [s for s in range(2 + 3, 49 + 50 + 1) if candidate_sum_1(s, cp1)]
    print('cs1:', cs1)
    print()

    cp2 = [p for p in cp1 if candidate_product_2(p, cs1)]
    print('cp2:', cp2)
    print()

    cs2 = [s for s in cs1 if candidate_sum_2(s, cp2)]
    print('cs2:', cs2)
    print()

    # The problem is set such that we have only one candidate sum in the end.
    assert len(cs2) == 1
    s = cs2[0]

    # Check if each p in cp2 is a candidate product.
    cxy = [candidate_product_3(p, s) for p in cp2]

    # Remove all None values from the list of (x, y) pairs.
    cxy = [xy for xy in cxy if xy]

    # The problem is set such that we have only one candidate product
    # with factors x and y such that x + y = s.
```

```
    assert len(cxy) == 1
    x, y = cxy[0]
    print('x:', x)
    print('y:', y)


if __name__ == '__main__':
    main()
```

Program Output

```
cp1: [12, 18, 20, 24, 28, 30, 32, 36, 40, 42, 44, 45, 48, 50, 52, 54, 56, 60, 63, 64,
    66, 68, 70, 72, 75, 76, 78, 80, 84, 88, 90, 92, 96, 98, 99, 100, 102, 104, 105, 108,
     110, 112, 114, 116, 117, 120, 124, 126, 128, 130, 132, 135, 136, 138, 140, 144,
    147, 148, 150, 152, 153, 154, 156, 160, 162, 164, 165, 168, 170, 171, 172, 174, 175,
     176, 180, 182, 184, 186, 188, 189, 190, 192, 195, 196, 198, 200, 204, 207, 208,
    210, 216, 220, 222, 224, 225, 228, 230, 231, 232, 234, 238, 240, 243, 245, 246, 248,
     250, 252, 255, 256, 258, 260, 261, 264, 266, 270, 272, 273, 275, 276, 279, 280,
    282, 285, 286, 288, 290, 294, 296, 297, 300, 304, 306, 308, 310, 312, 315, 320, 322,
     324, 325, 328, 330, 336, 340, 342, 344, 345, 348, 350, 351, 352, 357, 360, 364,
    368, 370, 372, 374, 375, 376, 378, 380, 384, 385, 390, 392, 396, 399, 400, 405, 406,
     408, 410, 414, 416, 418, 420, 425, 429, 430, 432, 434, 435, 440, 441, 442, 444,
    448, 450, 455, 456, 459, 460, 462, 464, 465, 468, 470, 476, 480, 483, 486, 490, 492,
     494, 495, 496, 500, 504, 506, 510, 512, 513, 516, 518, 520, 522, 525, 528, 532,
    539, 540, 544, 546, 550, 552, 558, 560, 561, 567, 570, 572, 574, 576, 580, 585, 588,
     592, 594, 595, 598, 600, 602, 608, 609, 612, 616, 620, 621, 624, 627, 630, 637,
    638, 640, 644, 646, 648, 650, 656, 660, 663, 666, 672, 675, 680, 682, 684, 688, 690,
     693, 696, 700, 702, 704, 714, 715, 720, 726, 728, 735, 736, 738, 740, 741, 744,
    748, 750, 754, 756, 759, 760, 765, 768, 770, 774, 780, 782, 783, 784, 792, 798, 800,
     806, 810, 812, 814, 816, 819, 820, 825, 828, 832, 836, 840, 850, 855, 858, 860,
    864, 868, 870, 874, 880, 882, 884, 888, 891, 896, 897, 900, 902, 910, 912, 918, 920,
     924, 928, 930, 935, 936, 945, 946, 950, 952, 957, 960, 962, 966, 968, 969, 972,
    975, 980, 984, 986, 988, 990, 992, 1000, 1008, 1012, 1014, 1020, 1026, 1032, 1035,
    1036, 1040, 1044, 1050, 1053, 1054, 1056, 1064, 1066, 1071, 1078, 1080, 1088, 1092,
    1100, 1102, 1104, 1105, 1110, 1116, 1118, 1120, 1122, 1125, 1134, 1140, 1144, 1148,
    1150, 1152, 1155, 1160, 1170, 1173, 1176, 1178, 1184, 1188, 1190, 1196, 1197, 1200,
    1215, 1216, 1218, 1224, 1230, 1232, 1240, 1242, 1248, 1254, 1258, 1260, 1275, 1276,
    1280, 1288, 1292, 1296, 1300, 1302, 1311, 1312, 1320, 1323, 1326, 1330, 1332, 1334,
    1344, 1350, 1360, 1364, 1365, 1368, 1377, 1380, 1386, 1392, 1394, 1400, 1404, 1406,
    1408, 1425, 1426, 1428, 1430, 1440, 1449, 1450, 1452, 1456, 1458, 1470, 1472, 1480,
    1482, 1485, 1488, 1496, 1500, 1508, 1512, 1518, 1520, 1530, 1536, 1540, 1550, 1554,
    1560, 1564, 1566, 1568, 1575, 1584, 1596, 1600, 1610, 1612, 1617, 1620, 1624, 1628,
    1632, 1638, 1650, 1656, 1664, 1672, 1674, 1680, 1700, 1702, 1710, 1716, 1725, 1728,
    1736, 1740, 1748, 1750, 1755, 1760, 1764, 1768, 1776, 1782, 1792, 1794, 1798, 1800,
    1820, 1824, 1836, 1848, 1850, 1856, 1860, 1872, 1890, 1904, 1914, 1920, 1932, 1938,
    1944, 1950, 1960, 1972, 1980, 1984, 2016, 2030, 2040, 2046, 2052, 2070, 2080, 2100,
    2108, 2112, 2142, 2145, 2160, 2184, 2200, 2205, 2240, 2268, 2280, 2340, 2352]

cs1: [11, 17, 23, 27, 29, 35, 37, 41, 47, 53]

cp2: [18, 24, 28, 50, 52, 54, 76, 92, 96, 100, 110, 112, 114, 124, 130, 138, 140, 148,
    152, 154, 160, 162, 168, 170, 172, 174, 176, 182, 186, 190, 198, 204, 208, 216, 232,
     234, 238, 240, 246, 250, 252, 270, 276, 280, 282, 288, 294, 304, 306, 310, 336,
```

```
    340, 348, 360, 364, 370, 378, 390, 400, 408, 414, 418, 430, 442, 480, 492, 496, 510,
     520, 522, 532, 540, 550, 552, 570, 592, 612, 630, 646, 660, 672, 682, 690, 696,
    700, 702]

cs2: [17]

x: 4
y: 13
```