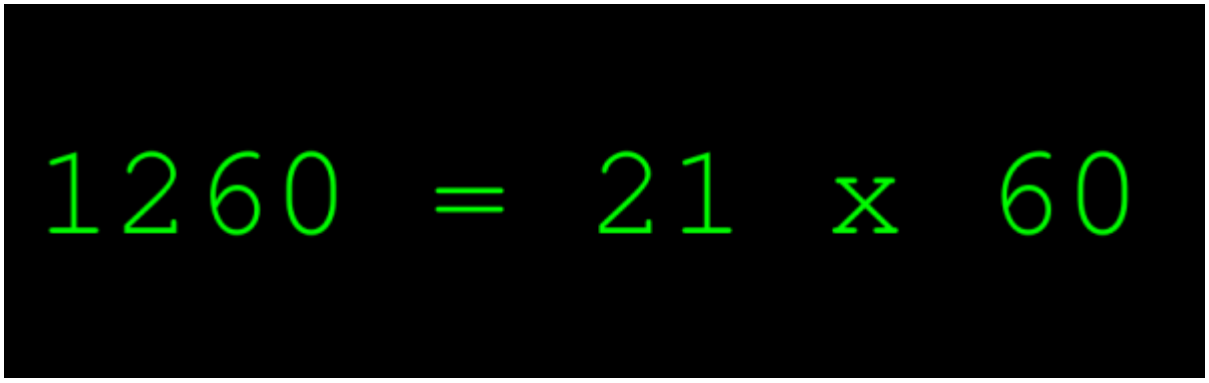

Page under construction, it is draft invisible without knowing url

Vampire Numbers: Traits and Uniqueness

Vampire numbers are numbers that have unique mathematical characteristics. They are even numbers that can be represented as the product of two numbers (called fangs) with the same number of digits, where the digits of the fangs must be a combination of the digits of the vampire number.

Features of Vampire Numbers

1. **Even number of digits**: A vampire number must have an even number of digits.
2. **Fangs**: They should be paired in such a way that both fangs connect to the vampire number.
3. **Lack of two zeros at the end**: They must not end in „00” (e.g. 1000).

Examples

1. **1260 = 21 * 60**
2. **1827 = 21 * 87**
3. **2187 = 27 * 81**

Difficulties in Programming

Writing an efficient program to find vampire numbers is a challenging task for several reasons:

The complexity of the Algorithm

1. **Combinations of numbers**: Find all possible pairs of numbers that satisfy the conditions. Checking that the digits of the two canines make up the vampire number requires the analysis of certain permutations.
2. **Mathematical constraints**: The program must skip those numbers that end in „00”, which further complicates the process.

Performance

1. Computing for a large number of permutations can be time consuming. A typical approach is to search through all possible tuples, which can result in large programme execution times.

Example implementation in python and comparison with the implementation in C

```
def is_vampire_number(n):
    # Check if the number ends with "00"
    if str(n).endswith("00"):
        return False

    # Iterate over possible fangs
    for i in range(10, 100):  # Fangs must be two digits
        if n % i == 0:  # Check if i is a divisor of n
            fang2 = n // i  # Calculate the second fang
            # Check if both have the same number of digits and are
permutations
            if len(str(i)) == len(str(fang2)) and sorted(str(i)) ==
sorted(str(fang2)):
                return True
    return False

# Searching for vampire numbers in a given range
vampire_numbers = []
for num in range(1000, 10000):  # Searching in the four-digit range
    if is_vampire_number(num):
        vampire_numbers.append(num)

print("Vampire numbers:", vampire_numbers)
```

Running the program:

```
$ python3 main.py
Vampire numbers: [1008, 1024, 1089, 1156, 1207, 1225, 1296, 1369, 1444,
1458, 1462, 1521, 1612, 1681, 1729, 1764, 1849, 1855, 1936, 1944, 2025,
2116, 2209, 2268, 2296, 2304, 2401, 2430, 2601, 2668, 2701, 2704, 2809,
2916, 2944, 3025, 3136, 3154, 3249, 3364, 3478, 3481, 3627, 3640, 3721,
3844, 3969, 4032, 4096, 4225, 4275, 4356, 4489, 4606, 4624, 4761, 4930,
5041, 5092, 5184, 5329, 5476, 5605, 5625, 5776, 5848, 5929, 6084, 6241,
6561, 6624, 6724, 6786, 6889, 7056, 7225, 7396, 7569, 7663, 7744, 7921,
8281, 8464, 8649, 8722, 8836, 9025, 9216, 9409, 9604, 9801]
```

Implementation in C

Link: https://github.com/plerros/helsing

```
$ time ./helsing -n 4 --progress
Checking interval: [1000, 9999]
1000, 1133  1/66
1134, 1267  2/66
1268, 1401  3/66
1402, 1535  4/66
1536, 1669  5/66
1670, 1803  6/66
1804, 1937  7/66
1938, 2071  8/66
2072, 2205  9/66
2206, 2339  10/66
2340, 2473  11/66
2474, 2607  12/66
2608, 2741  13/66
2742, 2875  14/66
2876, 3009  15/66
3010, 3143  16/66
3144, 3277  17/66
3278, 3411  18/66
3412, 3545  19/66
3546, 3679  20/66
3680, 3813  21/66
3814, 3947  22/66
3948, 4081  23/66
4082, 4215  24/66
4216, 4349  25/66
4350, 4483  26/66
4484, 4617  27/66
4618, 4751  28/66
4752, 4885  29/66
4886, 5019  30/66
5020, 5153  31/66
5154, 5287  32/66
5288, 5421  33/66
5422, 5555  34/66
5556, 5689  35/66
5690, 5823  36/66
5824, 5957  37/66
5958, 6091  38/66
6092, 6225  39/66
6226, 6359  40/66
6360, 6493  41/66
6494, 6627  42/66
6628, 6761  43/66
6762, 6895  44/66
6896, 7029  45/66
7030, 7163  46/66
7164, 7297  47/66
```

```
7298, 7431  48/66
7432, 7565  49/66
7566, 7699  50/66
7700, 7833  51/66
7834, 7967  52/66
7968, 8101  53/66
8102, 8235  54/66
8236, 8369  55/66
8370, 8503  56/66
8504, 8637  57/66
8638, 8771  58/66
8772, 8905  59/66
8906, 9039  60/66
9040, 9173  61/66
9174, 9307  62/66
9308, 9441  63/66
9442, 9575  64/66
9576, 9709  65/66
9710, 9801  66/66
Found: 7 vampire number(s).

real    0m0,003s
user    0m0,001s
sys 0m0,004s
```

As can be seen in the range of 4-digit numbers, the execution was thousandths of a second.

Summary

Vampire numbers present an intriguing problem in mathematics and programming. The performance of an algorithm searching for these numbers requires an understanding of digit combinations and appropriate management of execution time. Examples and algorithms from the programming forum show the variety of approaches to the problem, making it an attractive challenge for programmers.