Advent of Code

December 2025

1	**2**	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31				

```
    *
   /.\
  /o..\
  /..o\
 /.o..o\
 /...o.\
/..o....\
^^^[_]^^^
```

Answers

Part 1

Executed on the example input.
- Expected answer: **1227775554**
- Computed answer: **1227775554**

This seems correct

Executed on the evaluation input.
- Computed answer: **40398804950**

Part 2

Executed on the example input.
- Expected answer: **4174379265**
- Computed answer: **4174379265**

This seems correct

Executed on the evaluation input.
- Computed answer: **65794984339**

Logs

Parsing

Part 1

```
(
  ("11", "22"),
  ("95", "115"),
  ("998", "1012"),
  ("1188511880", "1188511890"),
  ("222220", "222224"),
  ("1698522", "1698528"),
  ("446443", "446449"),
  ("38593856", "38593862"),
  ("565653", "565659"),
  ("824824821", "824824827"),
  ("2121212118", "2121212124"),
)
```

Part 2

(Same)

Part 1

```
Range: 11 - 22
Range: 95 - 115
lo decides the length
Real range: 95 - 99
Range: 998 - 1012
hi decides the length
Real range: 1000 - 1012
Range: 1188511880 - 1188511890
Range: 222220 - 222224
Range: 1698522 - 1698528
Odd length is safe
Range: 446443 - 446449
Range: 38593856 - 38593862
Range: 565653 - 565659
Range: 824824821 - 824824827
Odd length is safe
Range: 2121212118 - 2121212124
Invalid: (
  11,
  22,
  99,
  1010,
  1188511885,
  222222,
  446446,
  38593859,
)
```

Part 2

```
Range: 11 - 22
try dividing in 2 repetitions of 1 each
found 11
found 22
Range: 95 - 99
try dividing in 2 repetitions of 1 each
found 99
Range: 100 - 115
try dividing in 3 repetitions of 1 each
found 111
Range: 998 - 999
try dividing in 3 repetitions of 1 each
found 999
Range: 1000 - 1012
try dividing in 2 repetitions of 2 each
found 1010
try dividing in 4 repetitions of 1 each
Range: 1188511880 - 1188511890
try dividing in 2 repetitions of 5 each
found 1188511885
try dividing in 5 repetitions of 2 each
try dividing in 10 repetitions of 1 each
Range: 222220 - 222224
try dividing in 2 repetitions of 3 each
found 222222
try dividing in 3 repetitions of 2 each
found 222222
try dividing in 6 repetitions of 1 each
found 222222
```

```
Range: 1698522 - 1698528
try dividing in 7 repetitions of 1 each
Range: 446443 - 446449
try dividing in 2 repetitions of 3 each
found 446446
try dividing in 3 repetitions of 2 each
try dividing in 6 repetitions of 1 each
Range: 38593856 - 38593862
try dividing in 2 repetitions of 4 each
found 38593859
try dividing in 4 repetitions of 2 each
try dividing in 8 repetitions of 1 each
Range: 565653 - 565659
try dividing in 2 repetitions of 3 each
try dividing in 3 repetitions of 2 each
found 565656
try dividing in 6 repetitions of 1 each
Range: 824824821 - 824824827
try dividing in 3 repetitions of 3 each
found 824824824
try dividing in 9 repetitions of 1 each
Range: 2121212118 - 2121212124
try dividing in 2 repetitions of 5 each
try dividing in 5 repetitions of 2 each
found 2121212121
try dividing in 10 repetitions of 1 each
Invalid: (
  (11, 11, 22),
  (22, 11, 22),
  (99, 95, 99),
  (111, 100, 115),
  (999, 998, 999),
  (1010, 1000, 1012),
  (1188511885, 1188511880, 1188511890),
  (222222, 222220, 222224),
  (446446, 446443, 446449),
  (38593859, 38593856, 38593862),
  (565656, 565653, 565659),
  (824824824, 824824821, 824824827),
  (2121212121, 2121212118, 2121212124),
)
```

Source code

Preliminaries

```
#import "/template/aot.typ"

#show: aot.format

#aot.parser(input => {
  input.trim("\n").split(",").map(l => l.split("-"))
})
```

Part 1

```
#aot.solve(input => {
  let invalids = ()
  for (lo, hi) in input {
    if lo.len() == hi.len() {
      if calc.rem(lo.len(), 2) == 1 {
        continue
      }
    } else if lo.len() + 1 == hi.len() {
      if calc.rem(lo.len(), 2) == 1 {
         lo = "1" + "0" * (hi.len() - 1)
      } else {
         hi = "9" * lo.len()
      }
    } else {
    }

    let half = int(lo.slice(0, int(lo.len() / 2)))
    let hi = int(hi)
    let lo = int(lo)
    while true {
      let num = int(str(half) + str(half))
      if num > hi { break }
      if num >= lo {
        invalids.push(num)
      }
      half += 1
    }
  }
  aot.answer(invalids.sum())
})
```

Part 2

```
#aot.solve(input => {
  let actual-ranges = ()
  for (lo, hi) in input {
    if lo.len() == hi.len() {
      actual-ranges.push((lo, hi))
    } else if lo.len() + 1 == hi.len() {
      actual-ranges.push((lo, "9" * lo.len()))
      actual-ranges.push(("1" + "0" * (hi.len() - 1), hi))
    }
  }

  let invalids = ()
  for (lo, hi) in actual-ranges {
    let divs = aot.utils.divisors(lo.len())
    for div in divs.filter(d => d != 1) {
      let len = int(lo.len() / div)
      let piece = int(lo.slice(0, len))
      let hi = int(hi)
      let lo = int(lo)
      while true {
        let num = int(str(piece) * div)
        if num > hi { break }
        if num >= lo {
          assert(lo <= num and num <= hi)
          assert(num == int(str(num).slice(0, len) * div))
          invalids.push((num, lo, hi))
        }
        piece += 1
      }
    }
  }
  let invalids = invalids.dedup()
  aot.answer(invalids.map(t => t.at(0)).sum())
})
```