```python

def run(primes: list) -> dict:
  """
  Returns a dict of all Germain Prime sequences identified in the given list
  """

  # dict for storing results
  sequences = dict()

  # building set of primes of O(1) checking
  primeSet = set(primes)

  # iterating through all primes in given list
  for prime in primes:

    # list for storing the current sequence achieved
    seq = list()

    # assigning the first prime to check as the current prime in the given list of primes
    gt = prime

    # checking that gt is Germain, and if so, adding to sequence and updating gt
    while (gt * 2) + 1 in primeSet:
      seq.append(gt)

      gt = (gt * 2) + 1

  # if the seq variable is not empty, meaning at least one Germain prime was identified, it gets added
 to results
  if seq:
    sequences[prime] = {"sequence": seq, "length": len(seq)}

  # returning the results
  return sequences


```

Algorithm to identify sequences of Germain primes.

Let p be a prime number.

p is a Germain Prime if $2p + 1$ is also prime.

Our objective was to identify exceptionally long sequences of Germain primes. To do so, we examined all primes up to 10^9. On this interval, we found 14,156,112 Germain primes. The two longest sequences were found starting at 19099919 and 52554569 and had seven primes each.