ERIC BAILEY

ABSTRACT ALGEBRA IN GAP

Contents

Basic System Interaction

Exercise 1

a `IsPerfect` is a function that takes a positive integer `n` and returns `true` if `n` is perfect and `false` otherwise.

We could define a function to compute the aliquot sum of a positive integer n:

5a ⟨*Compute the aliquot sum of a positive integer* 5a⟩≡

```
AliquotSum := n -> Sum(DivisorsInt(n)) - n;
```

Defines:
`AliquotSum`, used in chunk 5b.

$$s(n) \equiv \sigma(n) - n$$

Then, using that definition, we could write a function to determine whether a positive integer n is perfect:

5b ⟨*Determine whether a positive integer is perfect* 5b⟩≡

```
IsPerfect := n -> n = AliquotSum(n);
```

Uses `AliquotSum` 5a and `IsPerfect` 6d.

Conveniently, GAP ships with `Sigma`, which we can use instead.

$$\sigma(n) = \sum_{d|n} d$$

5c ⟨*Determine whether a positive integer is perfect, using Sigma* 5c⟩≡ (6d)

```
n -> Sigma(n) = 2*n
```

$$\texttt{IsPerfect}(n) := \sigma(n) = 2n$$

b To find all perfect numbers less than 1000, run the following:

5d ⟨*Find all perfect numbers less than 1000* 5d⟩≡ (6d 7b)

```
Filtered([1..999], IsPerfect);
```

Uses `IsPerfect` 6d.

$$\{n \in \mathbb{Z}^+ \mid 1 \leq n < 1000,\ \texttt{IsPerfect}(n)\}$$

... which results in:

5e ⟨*All perfect numbers less than 1000* 5e⟩≡ (6d 7b)

```
[ 6, 28, 496 ]
```

c Not all numbers of the form $2^n(2^{n+1} - 1)$, for some positive integer n, are perfect.

5f ⟨*Not all perfect* 5f⟩≡

```
gap> ForAll( PositiveIntegers,
>            n -> IsPerfect(2^n * (2^(n+1) - 1)) );
false
```

Uses `IsPerfect` 6d.

d In Euclid's formation rule (IX.36), he proved $\frac{q(q+1)}{2}$ is an even perfect number where q is a prime of the form $2^p - 1$ for prime p, a.k.a. a Mersenne prime.

6a ⟨*Euclid's IX.36* 6a⟩≡

```
gap> MersennePrimes := Filtered( List( Primes{[1..50]},
                                       p -> 2^p - 1 ),
                                 IsPrime );
[ 3, 7, 31, 127, 8191, 131071, 524287, 2147483647,
  2305843009213693951, 618970019642690137449562111,
  162259276829213363391578010288127,
  170141183460469231731687303715884105727 ]
gap> ForAll( MersennePrimes, q -> IsPerfect(q * (q + 1) / 2) );
true
```

Uses IsPerfect 6d.

e TODO: Prove it.

Code

6b ⟨*Filter for positive integers* 6b⟩≡ (6)

```
IsInt and IsPosInt
```

6c ⟨*lib/PerfectNumbers.gd* 6c⟩≡

```
#! @Chapter PerfectNumbers

#! @Section The IsPerfect() Operation

#! @Description
#!  Determine whether a positive <A>int</A> is perfect.
#! @Arguments int
DeclareOperation( "IsPerfect",
    [ ⟨Filter for positive integers 6b⟩ ] );
```

Uses IsPerfect 6d.

6d ⟨*lib/PerfectNumbers.gi* 6d⟩≡

```
#! @Chapter PerfectNumbers

#! @Section The IsPerfect() Operation

InstallMethod( IsPerfect,
    "for a positive integer",
    [ ⟨Filter for positive integers 6b⟩ ],
    ⟨Determine whether a positive integer is perfect, using Sigma 5c⟩ );

#! @BeginExample
⟨Find all perfect numbers less than 1000 5d⟩
#! ⟨All perfect numbers less than 1000 5e⟩
#! @EndExample
```

Defines:
IsPerfect, used in chunks 5–7.

Tests

7a ⟨*<Run the tests* 7a⟩≡

```
gap> Test("tst/PerfectNumbers.tst");
```

7b ⟨*tst/PerfectNumbers.tst* 7b⟩≡

```
gap> START_TEST("AAIG package: PerfectNumbers.tst");
gap> LoadPackage("AAIG", false);
#I  method installed for IsPerfect matches more than one declaration
true
gap> ⟨Find all perfect numbers less than 1000 5d⟩
⟨All perfect numbers less than 1000 5e⟩
gap> STOP_TEST( "PerfectNumbers.tst", 10000 );
```

Uses IsPerfect 6d.

Chunks

Index

Bibliography