ERIC BAILEY

ABSTRACT ALGEBRA IN GAP

Contents

Basic System Interaction

Exercise 1

a) *Write a function that takes a positive integer n as input and returns* `true` *if n is perfect and* `false` *if n is not perfect.*

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

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

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

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

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

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` 7a.

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⟩≡ (7a)

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

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

b) *Use your function to find all perfect numbers less than 1000.*

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

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

$\{n \in \mathbb{Z}^+ \mid 1 \leq n \leq 999,\ \texttt{IsPerfect}(n)\}$

Uses `IsPerfect` 7a.

... which results in:

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

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

c) *Notice that all of the numbers you found have a certain form, namely $2^n(2^{n+1}-1)$ for some integer n. Are all numbers of this form perfect?*

No, using GAP we can show not all such numbers are perfect.

6a ⟨*not all such numbers are perfect* 6a⟩≡

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

Uses IsPerfect 7a.

d) *By experimenting in GAP, conjecture a necessary and sufficient condition for $2^n(2^{n+1}-1)$ to be a perfect number.*

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.

6b ⟨*Euclid's IX.36* 6b⟩≡

```
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 7a.

e) *Prove your conjecture is correct.*

Prove it

Code

For IsPerfect, use the following filter, since we only care about integers, or more specifically, positive integers.

6c ⟨*Filter for positive integers* 6c⟩≡ (6d 7a)

```
IsInt and IsPosInt
```

6d ⟨*gap/PerfectNumbers.gd* 6d⟩≡

```
#! @Chapter PerfectNumbers

#! @Section The IsPerfect() Operation

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

Uses IsPerfect 7a.

7a ⟨*gap/PerfectNumbers.gi* 7a⟩≡

```
#! @Chapter PerfectNumbers

#! @Section The IsPerfect() Operation

InstallMethod( IsPerfect,
    "for a positive integer",
    [ ⟨Filter for positive integers 6c⟩ ],
    ⟨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 and 6.

Tests

Describe this

7b ⟨*tst/PerfectNumbers.tst* 7b⟩≡

```
gap> START_TEST("AAIG package: PerfectNumbers.tst");

gap> ⟨Find all perfect numbers less than 1000 5d⟩
⟨All perfect numbers less than 1000 5e⟩

gap> STOP_TEST( "AAIG package: PerfectNumbers.tst", 10000 );
```

To test the package, create a file `tst/testall.g`.

7c ⟨*tst/testall.g* 7c⟩≡

```
⟨Load the package 7d⟩

⟨Call TestDirectory 8a⟩

⟨Force quit GAP 8b⟩
```

First load the package:

7d ⟨*Load the package* 7d⟩≡ (7c)

```
LoadPackage( "AAIG" );
```

Then get the list of directory objects for the `tst` directory of the AAIG package:

7e ⟨*The list of directory objects* 7e⟩≡ (8a)

```
DirectoriesPackageLibrary("AAIG", "tst"),
```

... and call `TestDirectory` on it, with the following options:

7f ⟨*TestDirectory options record* 7f⟩≡ (8a)

```
rec( exitGAP := true,
     testOptions := rec(compareFunction := "uptowhitespace") )
```

8a ⟨*Call TestDirectory* 8a⟩≡ (7c)

```
TestDirectory( ⟨The list of directory objects 7e⟩
               ⟨TestDirectory options record 7f⟩ );
```

Finally, force quit GAP, in case it hasn't exited already:

8b ⟨*Force quit GAP* 8b⟩≡ (7c)

```
FORCE_QUIT_GAP(1);
```

Miscellaneous

9 ⟨*PackageInfo.g* 9⟩≡

```
SetPackageInfo( rec(
    PackageName := "AAIG",
    Subtitle := "Abstract Algebra in GAP",
    Version := "0.0.1",
    Date := "06/10/2017", # NOTE: dd/mm/yyyy
    PackageWWWHome :=
        Concatenation( "https://yurrriq.github.io/",
                       LowercaseString( ~.PackageName ) ),
    SourceRepository := rec(
        Type := "git",
        URL := "https://github.com/yurrriq/abstract-algebra-in-gap"
    ),
    IssueTrackerURL := Concatenation( ~.SourceRepository.URL, "/issues" ),
    SupportEmail := "eric@ericb.me",
    Persons := [
        rec(
          LastName := "Bailey",
          FirstNames := "Eric",
          IsAuthor := true,
          IsMaintainer := true,
          Email := ~.SupportEmail,
          # WWWHome := ...,
          # PostalAddress := ...,
          # Place := ...,
          # Institution := ...
        )
    ],
    Status := "other",
    README_URL := Concatenation( ~.PackageWWWHome, "/README.md" ),
    PackageInfoURL := Concatenation( ~.PackageWWWHome, "/PackageInfo.g" ),
    # TODO: AbstractHTML := ...,
    PackageDoc := rec(
      BookName := "AAIG",
      ArchiveURLSubset := ["docs"],
      HTMLStart := "docs/chap0.html",
      PDFFile := "docs/manual.pdf",
      SixFile := "docs/manual.six",
      LongTitle := "Abstract Algebra in GAP"
    ),
    Dependencies := rec(
      GAP := "4.8.3",
      NeededOtherPackages := [],
      SuggestedOtherPackages := [],
      ExternalConditions := []
    ),
    AvailabilityTest := ReturnTrue,
    TestFile := "tst/testall.g",
    Autoload := false,
    # Keywords := [ ... ],
    # BannerString := ...
  ));
```

10a ⟨*init.g* 10a⟩≡

```
  ReadPackage( "AAIG", "gap/PerfectNumbers.gd" );
```

10b ⟨*makedoc.g* 10b⟩≡

```
  LoadPackage( "AutoDoc" );
  AutoDoc( rec( autodoc := true,
                dir := "docs",
                scaffold := true ) );

  QUIT;
```

10c ⟨*read.g* 10c⟩≡

```
  ReadPackage( "AAIG", "gap/PerfectNumbers.gi" );
```

11 ⟨*default.nix* 11⟩≡

```
with import <nixpkgs> {};


let

  # gap = callPackage ./nix/gap.nix {};

in


stdenv.mkDerivation rec {
  name = "howtogap-${version}";
  version = builtins.readFile ./VERSION;
  src = ./.;

  buildInputs = [
    gap

    # coreutils
    less
    # which
  ];

  buildFlags = [ "GAPROOT=${gap}/share/gap/build-dir" ];

  installPhase = ''
    ${gap}/bin/gap.sh -b makedoc.g
    local pkgdir=$out/share/gap/build-dir/pkg/aiig
    mkdir -p $pkgdir
    cp -R {PackageInfo,init,makedoc,read}.g docs/ gap/ tst/ $pkgdir
  '';
}
```

Chunks

Index

Bibliography