THE UNIVERSITY OF SYDNEY
SCHOOL OF MATHEMATICS AND STATISTICS

Solutions to An Introduction to MAGMA

MagmaMondays: 9 October 2023 — Semester 2, 2023

Web Page: `https://sites.google.com/view/magma-mondays/`
Lecturer: Don Taylor

1. Suppose that *letters* is a sequence of letters. The following code produces all 'words' made from these letters.

```
[&*[ letters[i^p] : i in [1..n]] : p in Sym(n)] where n is #letters;
```

If you first type

```
letters := ElementToSequence("aact");
```

and then use the code above you will see that some 'words' appear twice.

(a) Write a few lines of code that produce a sequence of words without duplicates.

Solution:

```
[&*[ letters[i^p] : i in [1..n]] : p in Sym(n)] where n is #letters;
SetSeq(Set($1));
```

```
[ ctaa, atac, tcaa, acat, taca, taac, caat, cata, aact, aatc, atca, acta ]
```

(b) Change the code so that it produces a sequence of three letter 'words'.

Solution:

```
k := 3;
wds := SetSeq(Set(&cat[[&*[ letters[SetSeq(A)][i^p] : i in [1..k]] :
       p in Sym(k) ] : A in Subsets({1..#letters}, k)]));
wds;
```

```
[ tca, cta, tac, aat, ata, caa, aca, taa, cat, aac, act, atc ]
```

2. Write a function expression CatNum := **func**< n | . . . > such that CatNum(n) returns the nth Catalan number.

Solution:

```
CatNum := func< n | Binomial(2*n, n) div (n+1) >;
CatNum(100);
```

```
896519947090131496687170070074100632420837521538745909320
```

3. Here is the CatSeq function from the lecture.

```
CatSeq := function(n);
    if n eq 0 then seq := [1];
    elif n eq 1 then seq := [1,1];
    else
        seq := $$(n-1);
        Append(~seq, &+[Integers()| seq[k+1]*seq[n-k] : k in [0..n-1]]);
    end if;
    return seq;
end function;
```

Rewrite CatSeq as a function expression using **select**.

Solution:

CatSeq2 := ***func***< *n* | *n* ***eq*** 0 **select** [1] **else** *n* ***eq*** 1 **select** [1,1] **else**
  Append($$(*n*−1), &+[$$(*n*−1)[*k*+1]∗$$(*n*−1)[*n*−*k*] : *k* ***in*** [0. . *n*−1]]) >;
CatSeq3 := ***func***< *n* | *n* ***eq*** 0 **select** [1] **else** *n* ***eq*** 1 **select** [1,1] **else**
  (Append(*L*, &+[*L*[*k*+1]∗*L*[*n*−*k*] : *k* ***in*** [0. . *n*−1]]) **where** *L* ***is*** $$(*n*−1)) >;

There is considerable difference in the timing.

time CatSeq(8);

```
[ 1, 1, 2, 5, 14, 42, 132, 429, 1430 ]
Time: 0.000
```

time CatSeq2(8);

```
[ 1, 1, 2, 5, 14, 42, 132, 429, 1430 ]
Time: 13.220
```

time CatSeq3(8);

```
[ 1, 1, 2, 5, 14, 42, 132, 429, 1430 ]
Time: 0.000
```

4. A *hyperoval* in a projective plane of even order q is a set of $q+2$ points, no three of which are on a line.

(a) Find an example of a hyperoval in the 21-point projective plane. You can begin with the command

plane, *points*, *lines* := FiniteProjectivePlane(4);

Hint 1. What are *points*.1 and *points*.2? What is *lines*.3?

Hint 2. Exclude(∼*S*, *v*) removes the element *v* from the set *S*. If you want to remove a representative from *S* and assign it to a variable *x*, use ExtractRep(∼*S*, ∼*x*).

Solution: A very direct way to find a hyperoval is to inspect the coordinates of the points:

[*points*.*i* : *i* ***in*** [1. . 21]];

It is clear that no three of the four points

X := [*points*.*i* : *i* ***in*** [1,2,3,21]]; *X*;

```
[ (1 : 0 : 0), (0 : 1 : 0), (0 : 0 : 1), (1 : 1 : 1) ]
```

lie on a line. To extend *X* to a hyperoval you can use Magma to find the points not on any line through a pair of points of *X*. (For neater output let w be a primitive element of the field of 4 elements.)

F<*w*> := GaloisField(4);

Begin by letting *Y* be the set of all points. The object *points* is **not** a Magma set (check its type). So we convert it to a set as follows.

Y := Set(*points*);

Note that Points(*plane*) creates the *indexed* set of points but we don't use this because the intrinsic procedure Exclude requires a set or multi-set.

Now remove the points on lines through pairs of points of X. The line through the points u and v is *lines* ! $[u, v]$.

```
for i := 1 to 3 do for j := i+1 to 4 do
    for p in Set(lines ! [X[i], X[j]]) do Exclude(~Y, p); end for;
end for; end for;
Y;
```

```
{ ( 1 : w : w^2 ), ( 1 : w^2 : w ) }
```

The union of Set(X) with Y is a hyperoval.

(b) Write a function IsHyperoval(P, X) to test whether X is a hyperoval in a projective plane P.

Solution:

```
IsHyperoval := func< P, X | #X eq (Order(P) + 2) and
    forall{ m : m in Lines(P) | #{ x : x in X | x in m } le 2 } >;
```

Test this on the set found in part (a).

```
IsHyperoval( plane, Set(X) join Y );
```

```
true
```

(c) Find all the hyperovals in the 21-point projective plane.

Solution: Use Magma to create all 54 264 sets of 6 points then use your function IsHyperoval to select just those that are hyperovals.

```
plane, points, lines := FiniteProjectivePlane(4);
```

Let P be the indexed set of points.

```
P := Points(plane);
hyperovals := { H : h in Subsets({1..21},6) | IsHyperoval(plane,H)
        where H is P[SetSeq(h)] };
#hyperovals;
```

```
168
```

(d) Find the orbits of the groups PGL(3, 4) and PSL(3, 4) on the set of hyperovals.

Solution: Using the set *hyperovals* just constructed we can find a representative and print the length of its orbits.

```
h1 := Rep(hyperovals);
G := PGL(3,4);
S := PSL(3,4);
#(h1^G), #(h1^S);
```

```
168 56
```

Thus $\mathrm{PGL}(3,4)$ acts transitively on hyperovals and since $\mathrm{PSL}(3,4)$ is a normal subgroup of index 3, it has 3 orbits of length 56.

```
O1 := h1^S;
exists(h2){ h : h in hyperovals | h notin O1};
```

```
true
```

```
O2 := h2^S;
exists(h3){ h : h in hyperovals | h notin O1 and h notin O2};
```

```
true
```

```
O3 := h3^S;
```

```
hyperovals eq O1 join O2 join O3;
true
```

5. The points and lines of the 21-point plane can be identified with the 1- and 2-dimensional subspaces of a vector space of dimension 3 over the field of 4 elements. In this representation an example of a hyperoval is the set of singular points of a quadratic form together with its radical. You can use the following code to construct the form and the quadratic space.

```
P<x,y,z> := PolynomialRing(GaloisField(4),3);
f := x*y + z^2;
V := QuadraticSpace(f);
```

Find 6 *vectors* that represent the points of the hyperoval. Check that they do indeed form a hyperoval. (Hint. Radical(V) is the radical of V and QuadraticNorm(v) is the value of the quadratic form at the vector v.)

Solution: First find the subspaces.

```
ss := { sub<V | v > : v in V | v ne 0 and QuadraticNorm(v) eq 0 };
ss join:= {Radical(V)};
```

Next choose representative vectors.

```
H := { W.1 : W in ss };
```

6. Let G be a group. Write a function that returns exactly one representative of $\{x, x^{-1}\}$ for all $x \in G$. Test your function on the cyclic groups of orders 2,3,4, and 5 and the dihedral groups of orders 6, 8, 10 and 12.

Solution:

```
f := func< G | [ Rep(X) : X in { {x,x^-1} : x in G } ] >;
for n := 2 to 5 do n, f(CyclicGroup(n)); end for;
for n := 3 to 6 do 2*n, f(DihedralGroup(n)); end for;
```

7. A non-empty subset S of a group G is *product-free* if $ab \notin S$ for all $a, b \in S$.

Using the functions

```
prodfree := func< S | forall{<a,b> : a,b in S | a*b notin S } >;
checkmax1 := function(G)
   for a in G do
      if a eq One(G) then continue; end if;
      found := true;
      for b in G do
         if b eq One(G) or b eq a then continue; end if;
         if prodfree({a,b}) then found := false; continue; end if;
      end for;
      if found then return true, a; end if;
   end for;
   return false, _;
end function;
```

defined in the lecture find the groups in the Small Groups Database that contain a *maximal* product-free set of size 1.

Solution:

```
SGD := SmallGroupDatabase();
time for n := 2 to 63 do
    for j := 1 to NumberOfSmallGroups(SGD, n) do
        G := SmallGroup(SGD, n, j);
            found, witness := checkmax_1(G);
            if found then print n, j, witness; end if;
        end for;
    end for;
```

This takes approximately 2.17 seconds on my machine.

8. Write a function *checkmax*$_2$ that can be used to find the groups in the Small Groups Database that contain a *maximal* product-free set of size 2.

Make a conjecture about the classification of all finite group with a maximal product-free set of size 2.

Solution: The following function checks if G contains elements a and b such that $\{ a, b \}$ is product-free and maximal with respect to inclusion. It uses the function *prodfree* defined in the previous question.

```
checkmax_2 := function(G)
    ss := Setseq(Set(G));
    n := #ss;
    for i -> a in ss do    // dual iteration
        if a eq One(G) then continue; end if;
        for j := i+1 to n do
            b := ss[j];
            if b eq One(G) then continue; end if;
            S := { a, b };
            if prodfree(S) then
                found := true;
                for x in G do
                    if x eq One(G) or x in S then continue; end if;
                    if prodfree({a,b,x}) then found := false; continue; end if;
                end for;
                if found then return true, a, b; end if;
            end if;
        end for;
    end for;
    return false, _, _;
end function;

time for n := 2 to 100 do
    d := NumberOfSmallGroups(SGD, n);
    for j := 1 to d do
        G := SmallGroup(SGD, n, j);
        found, a, b := checkmax_2(G);
        if found then print n, j, a, b; end if;
    end for;
end for;
```

This takes almost half an hour of CPU time on my machine. The program finds 11 groups with a maximal product-free set of size 2. The largest order is 16. It can be proved that there are no other groups.

The groups that contain a maximal product-free set of size 3 are known. The largest order is 24. It is unknown which groups have a maximal product-free set of size great then 3. It is conjectured that if a group has a maximal product-free set of size k, its order is at most $3(k+1)^2$.