

# A Generalised Quadrangle

## Solutions

John Bamberg, Maska Law, Alice C. Niemeyer

```
vs := VectorSpace(GF(2), IdentityMat(4, GF(2)));
# The points are the non-zero vectors
points := Filtered( vs, x -> x <> Zero(vs) );
grp := SP(4, 2);

# 0 and 1 in GF(2)
o := Zero(GF(2));
i := One(GF(2));

# The form
f := [[o, o, o, i], [o, o, i, o], [o, i, o, o], [i, o, o, o]];
Display(f);

# The first vector in line l1
u := [i, o, o, o];

1. # find a point which has inner product 0 with u
v := First(points, x -> (u*f)*x = o);

# Generate the subspace of vs with basis u and x
l1sub := Subspace(vs, [u, v]);
# the line l1 consists of the non-zero vectors in l1sub
l1 := Difference(l1sub, [Zero(vs)]);
gap> Display(l1);
```

```
. . 1 .
1 . . .
1 . 1 .
```

```

2. # All lines are the images of l1 under Sp(4,2) acting on sets
   lines := Orbit(grp, l1, OnSets);

3. Length(lines);
   15

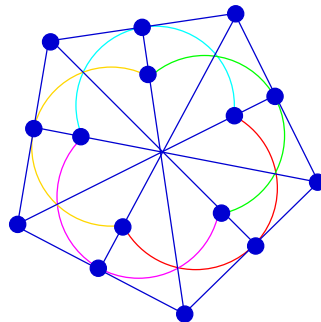
4. List( lines, i-> Length(i) );
   [ 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3 ]
   Hence each line has 3 points.

5. # Turn the group into a permutation group
   grpOnLines := Action(grp, lines, OnSets);

   # Is the group transitive on lines?
   IsTransitive(grpOnLines);
   true.

```

6. The picture:



```

7. # Find first line which has trivial intersection with l1
   lt := First( lines, i-> Length(Intersection( l1, i )) = 0 );
   Display(lt);

```

```

. 1 . 1
1 . 1 1
1 1 1 .

```

```

8. l2 := First( lines, i -> ((Length(Intersection( l1, i )) <>
   0 and

```

```
Length( Intersection( lt, i ) ) <> 0 )));
gap> Display(l2);
```

```
  . 1 . 1
  1 . 1 .
  1 1 1 1
```

9. yes.

10. no.

11. # The action is on the non-zero vectors by right multiplication

```
grpOnPoints:=Action(grp, points, OnRight);
Group([ (1,15,8,11)(2,5,6,12)(3,10,14,7)(4,9),
(1,4,8,2)(3,5,12,10)(6,9)(7,13,14,11) ])
AppendTo('mygroup.g', grpOnLines );
```

12. # Action of grp on flags

```
# flg is a flag and g a group element
# note that flg[1] is the point and
# flg[2] a line containing the point
OnFlags := function(flg,g)
return [flg[1]^g, OnSets(flg[2],g)];
end;
```

13. # Generate all flags

```
flags := Union(List( lines, l -> List( l, p -> [p, l] ) ));
```

```
# Generate the action of grp on the flags as permutation group
```

```
grpOnFlags:=Action(grp, flags, OnFlags);
```

14. IsTransitive(grpOnFlags);

```
true
```