Kleine Einfache Gruppen

Patrick Dabbert, Stephan Hilb und Martin Rösner

14. Februar 2013

Die Aufgabenstellung

Aufgabe 1:

Fertigen Sie eine Liste der endlichen einfachen nicht-abelschen Gruppen bis Ordnung 10.000 an. Sie dürfen hierbei verwenden, dass diese in folgender Liste enthalten sind: $A_n, PSL(2,q), PSL(3,3), PSU(3,3)$ *und* M_{11}*. Verwenden sie, dass die Ordnung eines endlichen Körpers eine Primzahlpotenz ist. Welche der von ihnen gefundenen Gruppen sind minimal einfach?*

Definition: Einfache Gruppen

Eine Gruppe heißt einfach wenn sie nur 1 und sich selbst als Normalteiler besitzt.

Definition Minimal Einfache Gruppen

Eine einfache Gruppe G heißt minimal einfach, wenn alle echten Untergruppen von G auflösbar sind.

Lösungsschritte

1. *gegebene Gruppen in Liste eintragen*
2. *Überprüfen ob Gruppen Einfach, Abelsch und Ordnung kleiner 10000.*
3. *Gruppen auf Isomorphie prüfen*
4. *Untergruppen auf Auflösbarkeit prüfen (hierbei reicht es die die Konjugationsklassen der Untergruppen zu betrachten)* $\Rightarrow$ *Minimal einfach*

Implementierung in GAP (Liste01.g)

```
1   #liste01.g
2   Liste01 := function ()
3   local Liste, Liste01, Liste02;
4   Liste := [];
5   #Schleife ueber die Ordnung
6   for n in [1..10] do
7       #Alternierende Gruppe erstellen
8       Add(Liste, AlternatingGroup(n));
9   od;
10  for n in [2..30] do
11      # ist q primzahlpotenz? (Koerpergrade sind Primzahlpotenzen)
12      if IsPrimePowerInt(n) then
13          Add(Liste, PSL(2,n));
14      fi;
15  od;
16  Add(Liste, PSL(3,3));
17  Add(Liste, PSU(3,3));
18  Add(Liste, MathieuGroup(11));
19
20  #Eigenschaften fuer einfache Gruppen ueberpruefen.
21  Liste := Filtered(Liste, g -> (IsSimpleGroup(g) and (not IsAbelian(g)) and Order(g)<= 10000))
```

Implementierung in GAP (Liste01.g)

```
1   #liste01.g
2   #Isomorphe Gruppen ausschliessen
3   k := Size(Liste);
4   u := IsomorphismGroups(AlternatingGroup(3),AlternatingGroup(4));
5   Liste01 := [];
6   for i in [1..k] do
7       p := 0;
8       for j in [i+1..k] do
9           if not u = IsomorphismGroups(Liste[i], Liste[j]) then
10              p := 1;
11          fi;
12      od;
13      if p = 0 then
14          Add(Liste01, Liste[i]);
15      fi;
16  od;
17  Liste02 := List(Liste01, l -> StructureDescription(l));
18
19  return Liste01;
20
21  end;
```

Implementierung in GAP (gapfile01.g)

```
1   Read("liste01.g");
2   Liste01 := Liste01();
3   Liste02 := List(Liste01, l -> StructureDescription(l));
4   Print("Liste aller endlichen, einfachen, nicht-abelschen Gruppen bis Ordnung 10000: \n");
5   Print(Liste02, "\n");
6
7
8   Liste03:=[];
9   # minimal einfach? (alle untergruppen aufloesbar):
10  for G in Liste01 do
11      minimal := 1;
12      for U in List(ConjugacyClassesSubgroups(G), Representative) do
13          if not IsSolvableGroup(U) and not U = G then
14              minimal := 0;
15              break;
16          fi;
17      od;
18      if minimal = 1 then
19          Add(Liste03, G);
20      fi;
21  od;
22  Print("\nMinimale einfache Gruppen \n");
23  Liste04 := List(Liste03, l -> StructureDescription(l));
24  Print(Liste04, "\n");
```

Ergebnis

einfach ["A7", "A5", "PSL(3,2)", "PSL(2,8)", "A6", "PSL(2,11)", "PSL(2,13)", "PSL(2,16)","PSL(2,17)", "PSL(2,19)", "PSL(2,23)","PSL(2,25)", "PSL(2,27)", "PSL(3,3)", "PSU(3,3)","M11"]

minimal einfach ["A5", "PSL(3,2)", "PSL(2,8)", "PSL(2,13)", "PSL(2,17)","PSL(2,23)", "PSL(2,27)", "PSL(3,3)"]

Anmerkung: PSL(3,2)$\cong$ PSL(2,7)

Die Aufgabenstellung

Aufgabe 2:

Zeigen Sie, dass A_8 *und* $PSL(3,4)$ *nicht isomorph sind.*

Lösungsschritte

Überprüfe für A_8 und $PSL(3,4)$ für welche Ordnungen Elemente vorkommen.

Implementierung in GAP (gapfile02.g)

```
1   G := AlternatingGroup(8);
2   H := PSL(3,4);
3   Print("G = ", G, " ", Order(G), "\nH = ", H, " ", Order(G), "\n");
4   Print("Sind G und H isomorph? \n", IsomorphismGroups(G, H), "\n");
5   Liste01 := []; Liste02 := [];
6   for x in G do
7       if not Order(x) in Liste01 then
8           Add(Liste01, Order(x));
9       fi;
10  od;
11  for x in H do
12      if not Order(x) in Liste02 then
13          Add(Liste02, Order(x));
14      fi;
15  od;
16  Sort(Liste01); Sort(Liste02);
17  Print("In G gibt es Elemente der Ordnung: ", Liste01, "\n","In H gibt es Elemente der Ordnung
18  : ", Liste02,  "\n");
```

Ergebnis

Ausgabe:

```
1  G = AlternatingGroup( [ 1 .. 8 ] ) 20160
2  H = Group( [ ( 3, 4, 5)( 7, 9, 8)(10,14,18)(11,17,20)(12,15,21)(13,16,19),
3    ( 1, 2, 6, 7,11, 3,10)( 4,14, 8,15,16,20,13)( 5,18, 9,19,21,17,12) ] ) 20160
4  Sind G und H isomorph?
5  fail
6  In G gibt es Elemente der Ordnung: [ 1, 2, 3, 4, 5, 6, 7, 15 ]
7  In H gibt es Elemente der Ordnung: [ 1, 2, 3, 4, 5, 7 ]
```

Es gibt also in A_8 Elemente von Ordnung 6 und 15, aber nicht in PSL(3,4). $\Rightarrow A_8 \ncong PSL(3,4)$

Die Aufgabenstellung

Aufgabe 3:
(In Zusammenarbeit mit Partitionen-Gruppe)
Bestimmen Sie, welche der von ihnen bestimmten Gruppen eine nicht-triviale Partition von Untergruppen besitzt.

Lösungsschritte

1. *Benutze erste Liste aus Aufgabe 1*
2. *Überprüfe für jede Gruppe ob nicht Triviale Partitionen vorliegen*
3. *Wenn ja, speichere die Gruppe in eine Liste*

Implementierung in GAP (gapfile03.g)

```
1   Read("liste01.g");
2   Read("HasNonTrivialPartition.txt");
3   Read("FindAllPartitions.txt");
4   Liste01 := Liste01();
5
6   Liste03 := [];
7   Print("Liste aller endlichen, einfachen, nicht-abelschen Gruppen bis
8    Ordnung 10000 mit allen Partitionen: \n");
9   for i in [1..Length(Liste01)] do
10      G:=Liste01[i];
11      if  HasNonTrivialPartition(G) then
12          Add(Liste03, G);
13          Print(StructureDescription(G), " hat nicht triviale Partitionen \n");
14      fi;
15  od;
```

Ergebnis

Programm rechnet noch, Zwischenergebnisse ohne A_7 bis PSL(3,3):
A5, PSL(3,2), PSL(2,8), A6 PSL(2,11), PSL(2,13), PSL(2,16), PSL(2,17), PSL(2,19), PSL(2,23), PSL(2,25), PSL(2,27) (Alle bisher geprüften)

Die Aufgabenstellung

Aufgabe 4:

(In Zusammenarbeit mit Sylowgruppen-Gruppe)
Bestimmen Sie, welche der von Ihnen gefundenen Gruppen N-Gruppen sind. Geben Sie außerdem für $P \in Syl_2(G)$ *und* $P \in Syl_3(G)$ *die Isomorphietypen von* P, $Z(P)$, $C_G(P)$ *und* $N_G(P)$ *an, wobei* G *eine Gruppe aus ihrer Liste ist.*

Lösungsschritte

1. Benutze erste Liste aus Teil eins.
2. Überprüfe die Gruppen darauf ob sie N Gruppen sind
3. wenn ja: Gib Isomorphietyp , Z(P), $C_G(P)$ und $N_G(P)$ an.

Implementierung in GAP (gapfile04.g)

```
1   Read("liste01.g");
2   Read("IsNGroup.g");
3   Read("StructureCentralizerSylowSubgroup.g");
4   Read("StructureCentreSylowSubgroup.g");
5   Read("StructureNormalizerSylowSubgroup.g");
6   Read("StructureSylowSubgroup.g");
7   Liste01 := Liste01();
8   Liste03 := [];
9   Print("Liste aller N-Gruppen davon: \n");
10  for G in Liste01 do
11      if IsNGroup(G) then
12          Print("==== ", StructureDescription(G), " ====\n");
13          Print("Syl2G-isomorphietyp: ", StructureSylowSubgroup(2, G), "\n");
14          Print("Syl3G-isomorphietyp: ", StructureSylowSubgroup(3, G), "\n");
15          Print("Zentrum: ", StructureCentreSylowSubgroup(2, G), "\n");
16          Print("Zentralisator ", StructureCentralizerSylowSubgroup(2, G), "\n");
17          Print("Normalisator ", StructureNormalizerSylowSubgroup(2, G), "\n\n");
18          Add(Liste03, G);
19      fi;
20  od;
21  Liste04 := List(Liste03, l -> StructureDescription(l));
22  Print(Liste04, "\n");
```

Ergebnis

Alle getesteten Gruppen sind N-Gruppen. (Genaueres siehe Tabelle)

einfach	["A7", "A5", "PSL(3,2)", "PSL(2,8)", "A6", "PSL(2,11)", "PSL(2,13)", "PSL(2,16)","PSL(2,17)", "PSL(2,19)", "PSL(2,23)","PSL(2,25)", "PSL(2,27)", "PSL(3,3)", "PSU(3,3)","M11"]
minimal einfach	["A5", "PSL(3,2)", "PSL(2,8)", "PSL(2,13)", "PSL(2,17)","PSL(2,23)", "PSL(2,27)", "PSL(3,3)"]
N-Gruppen	["A7", "A5", "PSL(3,2)", "PSL(2,8)", "A6", "PSL(2,11)", "PSL(2,13)", "PSL(2,16)","PSL(2,17)", "PSL(2,19)", "PSL(2,23)","PSL(2,25)", "PSL(2,27)", "PSL(3,3)", "PSU(3,3)","M11"]