GAPを利用した有限群論

(全19頁)

脇 克志

2007年9月20日

A. GAPで出来る事 (3–7頁)

B. 有限群の計算 (8–11頁)

C. GAPを使った有限群論の成功と失敗 (12–19頁)

A. GAPで出来る事 (3–7頁)

B. 有限群の計算 (8–11頁)

C. GAPを使った有限群論の成功と失敗 (12–19頁)

A. GAPで出来る事 (3–7頁)

B. 有限群の計算 (8–11頁)

C. GAPを使った有限群論の成功と失敗 (12–19頁)

A1:GAP とは?

G A P

A1:GAP とは?

Groups, A P

A1:GAP とは?

Groups, Algorithms, P

A1:GAP とは?

Groups, Algorithms, Programming

を意味しています。

A1:GAP とは?

Groups, Algorithms, Programming

を意味しています。

GAP は、
計算可能な離散代数、特に**計算可能な群論**のために作られたシステムです。

A1:GAP とは?

Groups, Algorithms, Programming

を意味しています。

GAP は、
計算可能な離散代数、特に**計算可能な群論**のために作られたシステムです。

GAP は、
プログラム言語であり、代数に関連する沢山の**関数**と代数構造の**データベース**を備えていて、いろいろな群、環、体を構成しながら、その代数構造を調べることが出来ます。

A1:GAP とは?

Groups, Algorithms, Programming

を意味しています。

GAP は、
計算可能な離散代数、特に**計算可能な群論**のために作られたシステムです。

GAP は、
プログラム言語であり、代数に関連する沢山の**関数**と代数構造の**データベース**を備えていて、いろいろな群、環、体を構成しながら、その代数構造を調べることが出来ます。

GAP は、
Free なソフトであり、**Knoppix/Math** にも収録されています。

A2:GAP で利用できる基本オブジェクト

- **真偽値 (Booleans)**
- **整数 (Integers)**
- **有理数 (Rational numbers)**
- **円分数 (Cyclotomic numbers)**
  $\sqrt{2} = \xi - \xi^3$ **(**ξ **は、１の原始８乗根)**
- **有限体の元 (Elements of finite fields)**
- **文字 (Characters)**

A2:GAP で利用できる基本オブジェクト

- 真偽値 (Booleans)
- **整数 (Integers)**
- 有理数 (Rational numbers)
- 円分数 (Cyclotomic numbers)
  $\sqrt{2} = \xi - \xi^3$ (ξ は、１の原始８乗根)
- 有限体の元 (Elements of finite fields)
- 文字 (Characters)

A2:GAP で利用できる基本オブジェクト

- **真偽値** (Booleans)
- **整数** (Integers)
- **有理数** (Rational numbers)
- **円分数** (Cyclotomic numbers)
  $\sqrt{2} = \xi - \xi^3$ (ξ **は、１の原始８乗根**)
- **有限体の元** (Elements of finite fields)
- **文字** (Characters)

A2:GAP で利用できる基本オブジェクト

- **真偽値** (Booleans)
- **整数** (Integers)
- **有理数** (Rational numbers)
- **円分数** (Cyclotomic numbers)
  $\sqrt{2} = \xi - \xi^3$ (**ξ は、１の原始８乗根**)
- **有限体の元** (Elements of finite fields)
- **文字** (Characters)

A2:GAP で利用できる基本オブジェクト

- 真偽値 (Booleans)
- 整数 (Integers)
- 有理数 (Rational numbers)
- 円分数 (Cyclotomic numbers)
  $\sqrt{2} = \xi - \xi^3$ (ξ は、１の原始８乗根)
- **有限体の元 (Elements of finite fields)**
- 文字 (Characters)

A2:GAP で利用できる基本オブジェクト

- **真偽値** (Booleans)
- **整数** (Integers)
- **有理数** (Rational numbers)
- **円分数** (Cyclotomic numbers)
  $\sqrt{2} = \xi - \xi^3$ (ξ **は、１の原始８乗根**)
- **有限体の元** (Elements of finite fields)
- **文字** (Characters)

A3:オブジェクトから作る構造型

- リスト (List)，集合 (Sets)，ベクトル (Vectors)，行列 (Matrices)，多項式 (Polynomials)，文字列 (Strings)
- 代数構造 (Domain) ... 群、環、体 etc
- レコード (Records) ... ユーザーが定義できる構造型データ

A3:オブジェクトから作る構造型

- リスト (List)，集合 (Sets)，ベクトル (Vectors)，行列 (Matrices)，多項式 (Polynomials)，文字列 (Strings)
- **代数構造 (Domain) ... 群、環、体 etc**
- レコード (Records) ... ユーザーが定義できる構造型データ

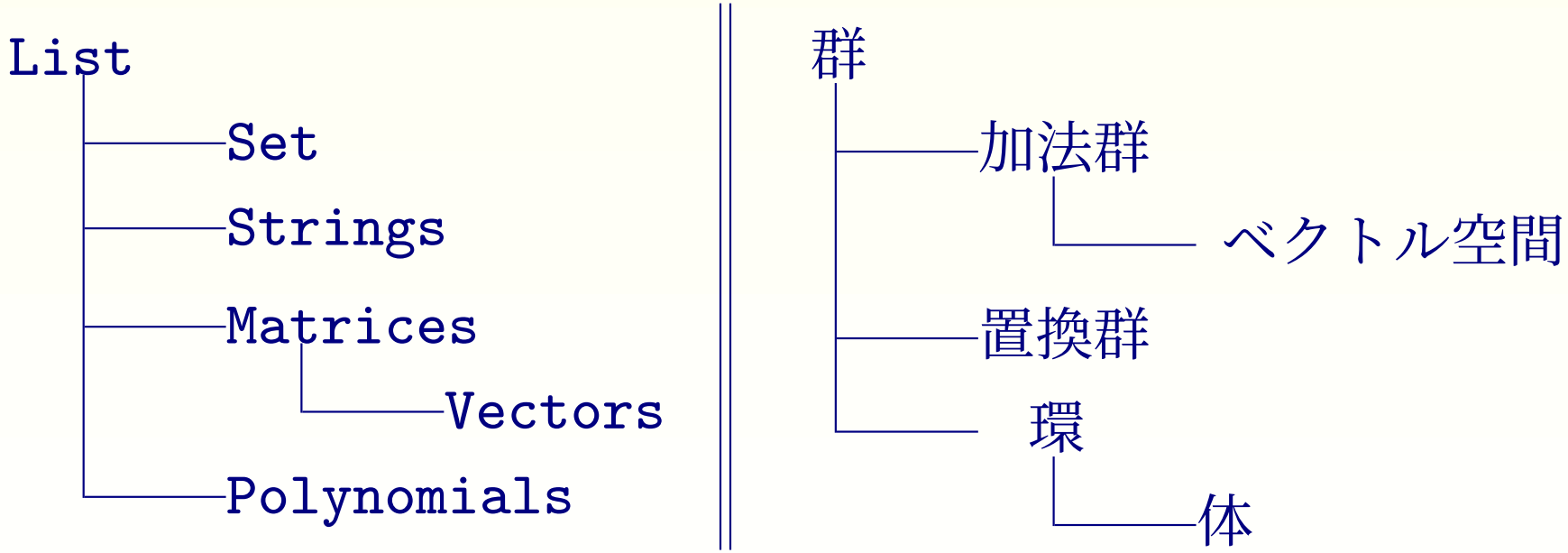
A3:オブジェクトから作る構造型

- リスト (List)，集合 (Sets)，ベクトル (Vectors)，行列 (Matrices)，多項式 (Polynomials)，文字列 (Strings)
- 代数構造 (Domain) ... 群、環、体 etc
- **レコード (Records) ... ユーザーが定義できる構造型データ**

A3:オブジェクトから作る構造型

- リスト (List)，集合 (Sets)，ベクトル (Vectors)，行列 (Matrices)，多項式 (Polynomials)，文字列 (Strings)
- 代数構造 (Domain) ... 群、環、体 etc
- **レコード (Records) ... ユーザーが定義できる構造型データ**

構造は、階層化されている



A4:GAPで作れる代数構造

- **有限群、自由群**
- **ベクトル空間、符号、グラフ**
- **群環、有限体上の行列環、多項式環、整数環**
- **有理数体、円分体、有限体**
- **各種の準同形写像、表現**

A4:GAPで作れる代数構造

- 有限群、自由群
- **ベクトル空間、符号、グラフ**
- 群環、有限体上の行列環、多項式環、整数環
- 有理数体、円分体、有限体
- 各種の準同形写像、表現

A4:GAPで作れる代数構造

- 有限群、自由群
- ベクトル空間、符号、グラフ
- **群環、有限体上の行列環、多項式環、整数環**
- 有理数体、円分体、有限体
- 各種の準同形写像、表現

A4:GAPで作れる代数構造

- 有限群、自由群
- ベクトル空間、符号、グラフ
- 群環、有限体上の行列環、多項式環、整数環
- **有理数体、円分体、有限体**
- 各種の準同形写像、表現

A4:GAPで作れる代数構造

- 有限群、自由群
- ベクトル空間、符号、グラフ
- 群環、有限体上の行列環、多項式環、整数環
- 有理数体、円分体、有限体
- **各種の準同形写像、表現**

A5:プログラム言語としてのGAP

- `if, for, while` などの基本的な命令を装備。
- 再帰的な関数定義が可能。
- 基本的に対話型と関数を定義してプログラムを実行することが可能。
- 構造化言語として、オブジェクト定義できる。
- 入出力ストリームを使った外部のバイナリープログラムとの連携

A5:プログラム言語としてのGAP

- `if, for, while` などの基本的な命令を装備。
- **再帰的な関数定義が可能。**
- 基本的に対話型と関数を定義してプログラムを実行することが可能。
- 構造化言語として、オブジェクト定義できる。
- 入出力ストリームを使った外部のバイナリープログラムとの連携

A5:プログラム言語としてのGAP

- `if, for, while` などの基本的な命令を装備。
- 再帰的な関数定義が可能。
- **基本的に対話型と関数を定義してプログラムを実行することが可能。**
- 構造化言語として、オブジェクト定義できる。
- 入出力ストリームを使った外部のバイナリープログラムとの連携

A5:プログラム言語としてのGAP

- `if, for, while` などの基本的な命令を装備。
- 再帰的な関数定義が可能。
- 基本的に対話型と関数を定義してプログラムを実行することが可能。
- **構造化言語として、オブジェクト定義できる。**
- 入出力ストリームを使った外部のバイナリープログラムとの連携

A5:プログラム言語としてのGAP

- `if, for, while` などの基本的な命令を装備。
- 再帰的な関数定義が可能。
- 基本的に対話型と関数を定義してプログラムを実行することが可能。
- 構造化言語として、オブジェクト定義できる。
- 入出力ストリームを使った外部のバイナリープログラムとの連携

B1:GAPで出来る有限群の計算

- **各種の有限群の構成（置換群、有限体上の行列群、生成元と関係式で定義された群、加群）**
- **部分群、極大部分群、正規部分群、シロー p-部分群、交換子群、固定部分群、中心化群、正規化群**
- **直積、半直積、中心拡大、自己同型群**
- **共役類、既約指標、有限体上のモジュラー表現**
- **可移群や完全群そして位数の小さい群を中心に集められた有限群のデータベース**

B1:GAPで出来る有限群の計算

- 各種の有限群の構成（置換群、有限体上の行列群、生成元と関係式で定義された群、加群）
- **部分群、極大部分群、正規部分群、シロー p-部分群、交換子群、固定部分群、中心化群、正規化群**
- 直積、半直積、中心拡大、自己同型群
- 共役類、既約指標、有限体上のモジュラー表現
- 可移群や完全群そして位数の小さい群を中心に集められた有限群のデータベース

B1:GAPで出来る有限群の計算

- 各種の有限群の構成（置換群、有限体上の行列群、生成元と関係式で定義された群、加群）
- 部分群、極大部分群、正規部分群、シロー p-部分群、交換子群、固定部分群、中心化群、正規化群
- **直積、半直積、中心拡大、自己同型群**
- 共役類、既約指標、有限体上のモジュラー表現
- 可移群や完全群そして位数の小さい群を中心に集められた有限群のデータベース

B1:GAPで出来る有限群の計算

- 各種の有限群の構成（置換群、有限体上の行列群、生成元と関係式で定義された群、加群）
- 部分群、極大部分群、正規部分群、シロー p-部分群、交換子群、固定部分群、中心化群、正規化群
- 直積、半直積、中心拡大、自己同型群
- **共役類、既約指標、有限体上のモジュラー表現**
- 可移群や完全群そして位数の小さい群を中心に集められた有限群のデータベース

B1:GAPで出来る有限群の計算

- 各種の有限群の構成（置換群、有限体上の行列群、生成元と関係式で定義された群、加群）
- 部分群、極大部分群、正規部分群、シロー p-部分群、交換子群、固定部分群、中心化群、正規化群
- 直積、半直積、中心拡大、自己同型群
- 共役類、既約指標、有限体上のモジュラー表現
- **可移群や完全群そして位数の小さい群を中心に集められた有限群のデータベース**

B2:有限群の構成 (御指名方式)

```
gap> G:=SymmetricGroup(6);;
gap> H:=MathieuGroup(11);;
gap> S:=SuzukiGroup(32);;
gap> M:=GL(4,3);
gap> N:=SL(2,2);
```

この方法で構成できるのは、巡回群、可換群、２面体群、Extraspecial 群 (格別群)、対称群、交代群、マシュー群、鈴木群、リー群、(射影) 一般線形群、(射影) 特殊線形群、(射影) 一般ユニタリー群、(射影) 特殊ユニタリー群、(射影) シンプレクティック群、一般直交群、特殊直交群

B3:有限群の構成 (条件選抜方式)

```
gap> AllTransitiveGroups(NrMovedPoints,[10..15],
>                        Size,          [1..100],
>                        IsAbelian,     false    );
gap> AllSmallGroups(     Size,          [1..100],
>                        IsAbelian,     false,
>                        IsNilpotentGroup,true);
```

B3:有限群の構成 (条件選抜方式)

```
gap> AllTransitiveGroups(NrMovedPoints,[10..15],
>                        Size,          [1..100],
>                        IsAbelian,     false    );
gap> AllSmallGroups(     Size,          [1..100],
>                        IsAbelian,     false,
>                        IsNilpotentGroup,true);
```

１つ目の命令では、**次数**が 10 から 15 で、

B3:有限群の構成 (条件選抜方式)

```
gap> AllTransitiveGroups(NrMovedPoints,[10..15],
>                        Size,          [1..100],
>                        IsAbelian,     false    );
gap> AllSmallGroups(     Size,          [1..100],
>                        IsAbelian,     false,
>                        IsNilpotentGroup,true);
```

1つ目の命令では、**次数**が10から15で、**位数**が1から100の

B3:有限群の構成 (条件選抜方式)

```
gap> AllTransitiveGroups(NrMovedPoints,[10..15],
>                        Size,          [1..100],
>                        IsAbelian,     false    );
gap> AllSmallGroups(     Size,          [1..100],
>                        IsAbelian,     false,
>                        IsNilpotentGroup,true);
```

1 つ目の命令では、**次数**が 10 から 15 で、**位数**が 1 から 100 の**可換でない**可移な群 97 個が得られます。

B3:有限群の構成 (条件選抜方式)

```
gap> AllTransitiveGroups(NrMovedPoints,[10..15],
>                        Size,          [1..100],
>                        IsAbelian,     false    );
gap> AllSmallGroups(     Size,          [1..100],
>                        IsAbelian,     false,
>                        IsNilpotentGroup,true);
```

１つ目の命令では、**次数**が 10 から 15 で、**位数**が 1 から 100 の**可換でない**可移な群 97 個が得られます。２つ目の命令では、**位数**が 100 以下の

B3:有限群の構成 (条件選抜方式)

```
gap> AllTransitiveGroups(NrMovedPoints,[10..15],
>                        Size,          [1..100],
>                        IsAbelian,     false    );
gap> AllSmallGroups(     Size,          [1..100],
>                        IsAbelian,     false,
>                        IsNilpotentGroup,true);
```

１つ目の命令では、**次数**が 10 から 15 で、**位数**が 1 から 100 の**可換でない**可移な群 97 個が得られます。２つ目の命令では、**位数**が 100 以下の**非可換**な

B3:有限群の構成 (条件選抜方式)

```
gap> AllTransitiveGroups(NrMovedPoints,[10..15],
>                        Size,          [1..100],
>                        IsAbelian,     false    );
gap> AllSmallGroups(     Size,          [1..100],
>                        IsAbelian,     false,
>                        IsNilpotentGroup,true);
```

１つ目の命令では、**次数**が 10 から 15 で、**位数**が 1 から 100 の**可換でない**可移な群 97 個が得られます。２つ目の命令では、**位数**が 100 以下の**非可換**な**巾零群**が 399 個が求まります。

B3:有限群の構成 (条件選抜方式)

```
gap> AllTransitiveGroups(NrMovedPoints,[10..15],
>                        Size,          [1..100],
>                        IsAbelian,     false    );
gap> AllSmallGroups(     Size,          [1..100],
>                        IsAbelian,     false,
>                        IsNilpotentGroup,true);
```

１つ目の命令では、**次数**が 10 から 15 で、**位数**が 1 から 100 の**可換でない**可移な群 97 個が得られます。２つ目の命令では、**位数**が 100 以下の**非可換**な**巾零群**が 399 個が求まります。 AllSmallGroups では、位数が 1024 でない 2000 以下のすべての有限群、位数が 50,000 以下でその素因数分解で各素数の巾が２以下のすべての有限群、位数が p^n $(n \leqq 6)$ のすべての p-群などが含まれています。

B4:有限群の構成 (部分群選択)

```
gap> G:=SymmetricGroup(8);;
gap> LT:=LatticeSubgroups(G);;
gap> CCR:=List(LT!.conjugacyClassesSubgroups,Representative);;
gap> subs:=Filtered(CCR,H->Size(H)>4 and Size(H)<13);;
gap> Set(List(subs,H->StructureDescription(H)));
[ "A4", "C10", "C12", "C2 x C2 x C2", "C3 : C4", "C3 x C3",
  "C4 x C2", "C5", "C6", "C6 x C2", "C7", "C8", "D10",
  "D12", "D8", "Q8", "S3" ]
```

B4:有限群の構成 (部分群選択)

```
gap> G:=SymmetricGroup(8);;
gap> LT:=LatticeSubgroups(G);;
gap> CCR:=List(LT!.conjugacyClassesSubgroups,Representative);;
gap> subs:=Filtered(CCR,H->Size(H)>4 and Size(H)<13);;
gap> Set(List(subs,H->StructureDescription(H)));
[ "A4", "C10", "C12", "C2 x C2 x C2", "C3 : C4", "C3 x C3",
  "C4 x C2", "C5", "C6", "C6 x C2", "C7", "C8", "D10",
  "D12", "D8", "Q8", "S3" ]
```

この例では、８次の**対称群** G から

B4:有限群の構成 (部分群選択)

```
gap> G:=SymmetricGroup(8);;
gap> LT:=LatticeSubgroups(G);;
gap> CCR:=List(LT!.conjugacyClassesSubgroups,Representative);;
gap> subs:=Filtered(CCR,H->Size(H)>4 and Size(H)<13);;
gap> Set(List(subs,H->StructureDescription(H)));
[ "A4", "C10", "C12", "C2 x C2 x C2", "C3 : C4", "C3 x C3",
  "C4 x C2", "C5", "C6", "C6 x C2", "C7", "C8", "D10",
  "D12", "D8", "Q8", "S3" ]
```

この例では、８次の**対称群** G から**部分群の共役類の代表元** CCR を求める。

B4:有限群の構成 (部分群選択)

```
gap> G:=SymmetricGroup(8);;
gap> LT:=LatticeSubgroups(G);;
gap> CCR:=List(LT!.conjugacyClassesSubgroups,Representative);;
gap> subs:=Filtered(CCR,H->Size(H)>4 and Size(H)<13);;
gap> Set(List(subs,H->StructureDescription(H)));
[ "A4", "C10", "C12", "C2 x C2 x C2", "C3 : C4", "C3 x C3",
  "C4 x C2", "C5", "C6", "C6 x C2", "C7", "C8", "D10",
  "D12", "D8", "Q8", "S3" ]
```

この例では、８次の**対称群** G から**部分群の共役類の代表元** CCR を求める。CCR から位数が４より大きく 13 より小さい 71 個の部分群が**選び出され** subs に入ります。

B4:有限群の構成 (部分群選択)

```
gap> G:=SymmetricGroup(8);;
gap> LT:=LatticeSubgroups(G);;
gap> CCR:=List(LT!.conjugacyClassesSubgroups,Representative);;
gap> subs:=Filtered(CCR,H->Size(H)>4 and Size(H)<13);;
gap> Set(List(subs,H->StructureDescription(H)));
[ "A4", "C10", "C12", "C2 x C2 x C2", "C3 : C4", "C3 x C3",
  "C4 x C2", "C5", "C6", "C6 x C2", "C7", "C8", "D10",
  "D12", "D8", "Q8", "S3" ]
```

この例では、８次の**対称群** G から**部分群の共役類の代表元** CCR を求める。CCR から位数が４より大きく 13 より小さい 71 個の部分群が**選び出され** subs に入ります。StructureDescription で 17 個の**異なる型**を表示しています。

B4:有限群の構成 (部分群選択)

```
gap> G:=SymmetricGroup(8);;
gap> LT:=LatticeSubgroups(G);;
gap> CCR:=List(LT!.conjugacyClassesSubgroups,Representative);;
gap> subs:=Filtered(CCR,H->Size(H)>4 and Size(H)<13);;
gap> Set(List(subs,H->StructureDescription(H)));
[ "A4", "C10", "C12", "C2 x C2 x C2", "C3 : C4", "C3 x C3",
  "C4 x C2", "C5", "C6", "C6 x C2", "C7", "C8", "D10",
  "D12", "D8", "Q8", "S3" ]
```

この例では、８次の**対称群** G から**部分群の共役類の代表元** CCR を求める。CCR から位数が４より大きく 13 より小さい 71 個の部分群が**選び出され** subs に入ります。StructureDescription で 17 個の**異なる型**を表示しています。(見つかった部分群の総数は、151,221 個)

C1:有限群論の成功

GAP を使った有限群の演習問題プリントの作成

C1:有限群論の成功

GAPを使った有限群の演習問題プリントの作成

有限群を出来るだけ実感してもらうためには、**たくさんの有限群に関連した問題を解いてもらうことが大事**と考えて、有限群の問題を出来るだけ**沢山!** それも**自分の力で** 解く体験を持たせるプリントを**継続的に**作ることにしました。そのためには、...

C1:有限群論の成功

GAPを使った有限群の演習問題プリントの作成

有限群を出来るだけ実感してもらうためには、**たくさんの有限群に関連した問題を解いてもらうことが大事**と考えて、有限群の問題を出来るだけ**沢山!** それも**自分の力で** 解く体験を持たせるプリントを**継続的に**作ることにしました。そのためには、...

1. **各学生には、個別のプリントで十分な問題数を解かせる。**
2. **採点は、出来るだけ楽になるように工夫する。**
3. **プリント作成では、問題作成以外は時間を必要としない。**

C1:有限群論の成功

GAPを使った有限群の演習問題プリントの作成

有限群を出来るだけ実感してもらうためには、**たくさんの有限群に関連した問題を解いてもらうことが大事**と考えて、有限群の問題を出来るだけ**沢山!** それも**自分の力で** 解く体験を持たせるプリントを**継続的に**作ることにしました。そのためには、...

1. 各学生には、個別のプリントで**十分な問題数**を解かせる。
2. **採点は、出来るだけ楽になるように工夫する。**
3. プリント作成では、問題作成以外は時間を**必要としない**。

C1:有限群論の成功

GAPを使った有限群の演習問題プリントの作成

有限群を出来るだけ実感してもらうためには、**たくさんの有限群に関連した問題を解いてもらうことが大事**と考えて、有限群の問題を出来るだけ**沢山!** それも**自分の力で** 解く体験を持たせるプリントを**継続的に**作ることにしました。そのためには、...

1. 各学生には、個別のプリントで**十分な問題数**を解かせる。
2. 採点は、出来るだけ**楽に**なるように工夫する。
3. **プリント作成では、問題作成以外は時間を必要としない。**

C2:個別のプリントで十分な問題数を解かせるには

関数 RandomProblemIndex(n,l,m) を準備する。
数列 [1..n] から長さ l の部分列を m 個作る取り出す。
ただし、1 から n までの数は最高で (l*m)/n+1 回しか使えない。

```
gap> RandomSubsets(15,5,15);
[ [ 2, 4, 9, 12, 15 ], [ 2, 4, 7, 12, 13 ], [ 3, 4, 7, 11, 13 ],
  [ 1, 6, 8, 12, 14 ], [ 3, 4, 7, 10, 15 ], [ 1, 4, 8, 11, 14 ],
  [ 3, 6, 9, 12, 15 ], [ 2, 6, 7, 12, 15 ], [ 2, 4, 8, 10, 14 ],
  [ 2, 5, 7, 12, 13 ], [ 1, 6, 8, 11, 13 ], [ 3, 6, 9, 11, 13 ],
  [ 3, 5, 9, 11, 13 ], [ 3, 6, 8, 10, 14 ], [ 2, 5, 9, 10, 15 ] ]
```

15 問の問題で 15 人の学生にそれぞれ 5 問づつ問題を解かせる。
実際には、60 問の問題で 65 人の学生にそれぞれ 5 問づつ解かせる。

C3:採点を、出来るだけ楽にするには

○採点する問題の種類を**減らす**。－ RandomProblemIndex で実現。

C3:採点を、出来るだけ楽にするには

○採点する問題の種類を**減らす**。− RandomProblemIndex で実現。

○問題に使う採点に**都合の良い**有限群を厳選する。

```
gap> pbs:=[];;
gap> for g in subs do
>       lt:=LatticeSubgroups(g);
>       ns:=Sum(lt!.conjugacyClassesSubgroups,Size);
>       if ns<7 and ns>2 then
>         Add(pbs,[ns,g]);
>       fi;
>     od;
gap> pbs[10];
[ 6, S3 ]
```

C4:問題作成以外は時間を不必要にするには

C4:問題作成以外は時間を不必要にするには

○プリント作成プログラムを３つのパートに分けて実行する。

- **関数** `TestHertGenerator`
  問題に必要な代数構造を構成し、問題数分だけ厳選する。

- **関数** `TestGenerator`
  問題を文章化する。

- **関数** `TestTexGenerator`
  問題文を結合して、プリント TEX ソースを作成する。

C4:問題作成以外は時間を不必要にするには

○プリント作成プログラムを３つのパートに分けて実行する。

- **関数** `TestHertGenerator`
  問題に必要な代数構造を構成し、問題数分だけ厳選する。
  (問題作成で一番時間がかかる)

- **関数** `TestGenerator`
  問題を文章化する。

- **関数** `TestTexGenerator`
  問題文を結合して、プリント TEX ソースを作成する。

C4:問題作成以外は時間を不必要にするには

○プリント作成プログラムを３つのパートに分けて実行する。

- **関数** `TestHertGenerator`
  問題に必要な代数構造を構成し、問題数分だけ厳選する。
  (問題作成で一番時間がかかる)

- **関数** `TestGenerator`
  問題を文章化する。

- **関数** `TestTexGenerator`
  問題文を結合して、プリント TEX ソースを作成する。

C4:問題作成以外は時間を不必要にするには

○プリント作成プログラムを３つのパートに分けて実行する。

- **関数** `TestHertGenerator`
  問題に必要な代数構造を構成し、問題数分だけ厳選する。
  (　問題作成で一番時間がかかる)

- **関数** `TestGenerator`
  問題を文章化する。(　問題に合わせて問題文を変える)

- **関数** `TestTexGenerator`
  問題文を結合して、プリント TEX ソースを作成する。

C4:問題作成以外は時間を不必要にするには

○プリント作成プログラムを３つのパートに分けて実行する。

- **関数** `TestHertGenerator`
  問題に必要な代数構造を構成し、問題数分だけ厳選する。
  (　問題作成で一番時間がかかる)

- **関数** `TestGenerator`
  問題を文章化する。(　問題に合わせて問題文を変える)

- **関数** `TestTexGenerator`
  問題文を結合して、プリント TEX ソースを作成する。

C4:問題作成以外は時間を不必要にするには

○プリント作成プログラムを３つのパートに分けて実行する。

- **関数** `TestHertGenerator`
  問題に必要な代数構造を構成し、問題数分だけ厳選する。
  (問題作成で一番時間がかかる)

- **関数** `TestGenerator`
  問題を文章化する。(問題に合わせて問題文を変える)

- **関数** `TestTexGenerator`
  問題文を結合して、プリント TEX ソースを作成する。
  (問題による変更無し)

C4:問題作成以外は時間を不必要にするには

○プリント作成プログラムを３つのパートに分けて実行する。

- **関数** `TestHertGenerator`
  問題に必要な代数構造を構成し、問題数分だけ厳選する。
  (　問題作成で一番時間がかかる)

- **関数** `TestGenerator`
  問題を文章化する。(　問題に合わせて問題文を変える)

- **関数** `TestTexGenerator`
  問題文を結合して、プリント TEX ソースを作成する。
  (　問題による変更無し)

○ GAP から TEX を起動して PDF ファイルを作ってしまう。

C5:出来上がったプリント

代数学 II 演習問題 07-1

次の置換群の真部分集合で自明でない群となるもの全て書いて、その総数も書きなさい。

03 { (), (2,3), (1,5), (1,5)(2,3)}

24 { (), (1,3)(4,6), (1,4,3,6), (1,6,3,4)}

33 { (), (2,3,6), (2,6,3), (1,4)(3,6), (1,4)(2,3), (1,4)(2,6)}

41 { (), (3,5), (1,2,6), (1,2,6)(3,5), (1,6,2), (1,6,2)(3,5)}

55 { (), (2,3,4), (2,4,3), (1,5,6), (1,5,6)(2,3,4), (1,5,6)(2,4,3), (1,6,5), (1,6,5)(2,3,4), (1,6,5)(2,4,3)}

得点	学籍番号	名前

代数学 II 演習問題 07 解答

01	3 : [[(), (1,6)(2,5)], [(), (3,4)], ...]
02	3 : [[(), (1,4)], [(), (3,5)], [(), (1,4)(3,5)]]
03	3 : [[(), (2,3)], [(), (1,5)], [(), (1,5)(2,3)]]
04	3 : [[(), (2,3)(4,5)], [(), (1,6)(4,5)], ...]
05	3 : [[(), (1,6)(2,4)(3,5)], [(), (2,3)(4,5)], ...]
06	3 : [[(), (1,6)(3,5)], [(), (2,4)], ...]
07	3 : [[(), (2,4)(5,6)], [(), (1,3)], ...]
08	3 : [[(), (1,5)(2,3)(4,6)], [(), (2,4)(3,6)], ...]
09	3 : [[(), (1,5)(3,4)], [(), (2,6)], ...]
10	3 : [[(), (1,6)(2,4)(3,5)], [(), (1,4)(2,6)], ...]
11	3 : [[(), (1,4)(2,5)(3,6)], [(), (1,5)(2,4)], ...]
12	3 : [[(), (1,4)(2,5)(3,6)], [(), (1,6)(3,4)], ...]
13	3 : [[(), (1,5)(2,3)(4,6)], [(), (1,4)(5,6)], ...]
14	3 : [[(), (1,5)(2,3)(4,6)], [(), (1,6)(4,5)], ...]
15	1 : [[(), (3,4)(5,6)]]
16	1 : [[(), (2,5)(3,6)]]
17	1 : [[(), (2,6)(4,5)]]
18	1 : [[(), (1,2)(5,6)]]
19	1 : [[(), (1,2)(3,4)]]

C6:有限群論プリントの結果

- **実施プリント:** 30 種類 (No.1〜No.30) のプリントを作成
- **実施方法:** 毎回プリント２枚を渡し、出来た学生からその場で採点し、出来なかった問題は次回までに解いて提出する。
- **注目点:** 途中から時間内にプリントを終えてしまう学生とプリントがどんどん貯る学生が出現する。終えてしまった学生が、まだ解けずに居る学生を教える場面が増える。
- **実施結果:** 受講学生 56 人中 41 人が No.24 まで完了した。
- **感想:** TA1 名に手伝ってもらって採点出来たため、採点の合間に学生からの質問にも答えられた。群論の理論を使って計算を簡単にする学生が出てきたが、その情報が他の学生にまで伝わってしまい、理由も分からず答えを出してしまう学生も現れた。

C6:有限群論プリントの結果

- **実施プリント:** 30 種類 (No.1〜No.30) のプリントを作成
- **実施方法:** 毎回プリント２枚を渡し、出来た学生からその場で採点し、出来なかった問題は次回までに解いて提出する。
- **注目点:** 途中から時間内にプリントを終えてしまう学生とプリントがどんどん貯る学生が出現する。終えてしまった学生が、まだ解けずに居る学生を教える場面が増える。
- **実施結果:** 受講学生 56 人中 41 人が No.24 まで完了した。
- **感想:** TA1 名に手伝ってもらって採点出来たため、採点の合間に学生からの質問にも答えられた。群論の理論を使って計算を簡単にする学生が出てきたが、その情報が他の学生にまで伝わってしまい、理由も分からず答えを出してしまう学生も現れた。

C6:有限群論プリントの結果

- **実施プリント:** 30 種類 (No.1〜No.30) のプリントを作成
- **実施方法:** 毎回プリント２枚を渡し、出来た学生からその場で採点し、出来なかった問題は次回までに解いて提出する。
- **注目点:** **途中から時間内にプリントを終えてしまう学生とプリントがどんどん貯る学生が出現する。終えてしまった学生が、まだ解けずに居る学生を教える場面が増える。**
- **実施結果:** 受講学生 56 人中 41 人が No.24 まで完了した。
- **感想:** TA1 名に手伝ってもらって採点出来たため、採点の合間に学生からの質問にも答えられた。群論の理論を使って計算を簡単にする学生が出てきたが、その情報が他の学生にまで伝わってしまい、理由も分からず答えを出してしまう学生も現れた。

C6:有限群論プリントの結果

- **実施プリント:** 30 種類 (No.1〜No.30) のプリントを作成
- **実施方法:** 毎回プリント２枚を渡し、出来た学生からその場で採点し、出来なかった問題は次回までに解いて提出する。
- **注目点:** 途中から時間内にプリントを終えてしまう学生とプリントがどんどん貯る学生が出現する。終えてしまった学生が、まだ解けずに居る学生を教える場面が増える。
- **実施結果:** **受講学生 56 人中 41 人が No.24 まで完了した。**
- **感想:** TA1 名に手伝ってもらって採点出来たため、採点の合間に学生からの質問にも答えられた。群論の理論を使って計算を簡単にする学生が出てきたが、その情報が他の学生にまで伝わってしまい、理由も分からず答えを出してしまう学生も現れた。

C6:有限群論プリントの結果

- **実施プリント:** 30 種類 (No.1〜No.30) のプリントを作成
- **実施方法:** 毎回プリント２枚を渡し、出来た学生からその場で採点し、出来なかった問題は次回までに解いて提出する。
- **注目点:** 途中から時間内にプリントを終えてしまう学生とプリントがどんどん貯る学生が出現する。終えてしまった学生が、まだ解けずに居る学生を教える場面が増える。
- **実施結果:** 受講学生 56 人中 41 人が No.24 まで完了した。
- **感想:** **TA1 名に手伝ってもらって採点出来たため、採点の合間に学生からの質問にも答えられた。群論の理論を使って計算を簡単にする学生が出てきたが、その情報が他の学生にまで伝わってしまい、理由も分からず答えを出してしまう学生も現れた。**

C7:有限群論の失敗

GAP を使った有限群のプログラミング実習

C7:有限群論の失敗

GAPを使った有限群のプログラミング実習

単純計算では、なかなか結果が得られない (計算が止まらない) 問題を群論の理論を使ったプログラミングをすることで、驚ほど**速く**結果が得られることを実感することで、一般論の凄さを実感させる。

C7:有限群論の失敗

GAPを使った有限群のプログラミング実習

単純計算では、なかなか結果が得られない (計算が止まらない) 問題を群論の理論を使ったプログラミングをすることで、驚ほど**速く**結果が得られることを実感することで、一般論の凄さを実感させる。

プログラムを駆使して、有限群のデータベースから与えられた条件を満たすものを捜し出す。

C8:有限群の解剖実習の結果

- **実施方法:** GAP の利用方法と基本的なプログラムの組み方を教えた後、課題となる有限群を調べる。
- **注目点:** 3 年生を対象としたが、OS の基本操作等でつまずく学生もいて、GAP を利用する前に学習すべきところが多かった。
- **実施結果:** 受講学生 40 人中の半数が 3 回目の講義から来なくなった。残った学生も計算機に対する基本知識に差があり、プログラムの基本的な部分が修得できてないため、与えられたプログラムを打ち込むのが精一杯で、自分でプログラムを構成できない学生がかなり見受けられた。
- **感想:** 学生の感想を聞いたところ、「プログラムだけでも難しいのに、良く分かっていない群論と組み合わせても、与えられたプリントを入力することで手一杯でとても難しかった。」

C8:有限群の解剖実習の結果

- **実施方法:** GAP の利用方法と基本的なプログラムの組み方を教えた後、課題となる有限群を調べる。
- **注目点:** ３年生を対象としたが、OS の基本操作等でつまずく学生もいて、GAP を利用する前に学習すべきところが多かった。
- **実施結果:** 受講学生 40 人中の半数が３回目の講義から来なくなった。残った学生も計算機に対する基本知識に差があり、プログラムの基本的な部分が修得できてないため、与えられたプログラムを打ち込むのが精一杯で、自分でプログラムを構成できない学生がかなり見受けられた。
- **感想:** 学生の感想を聞いたところ、「プログラムだけでも難しいのに、良く分かっていない群論と組み合わせても、与えられたプリントを入力することで手一杯でとても難しかった。」

C8:有限群の解剖実習の結果

- **実施方法:** GAP の利用方法と基本的なプログラムの組み方を教えた後、課題となる有限群を調べる。
- **注目点:** 3 年生を対象としたが、OS の基本操作等でつまずく学生もいて、GAP を利用する前に学習すべきところが多かった。
- **実施結果:** 受講学生 40 人中の半数が 3 回目の講義から来なくなった。残った学生も計算機に対する基本知識に差があり、プログラムの基本的な部分が修得できてないため、与えられたプログラムを打ち込むのが精一杯で、自分でプログラムを構成できない学生がかなり見受けられた。
- **感想:** 学生の感想を聞いたところ、「プログラムだけでも難しいのに、良く分かっていない群論と組み合わせても、与えられたプリントを入力することで手一杯でとても難しかった。」

C8:有限群の解剖実習の結果

- **実施方法:** GAP の利用方法と基本的なプログラムの組み方を教えた後、課題となる有限群を調べる。
- **注目点:** 3 年生を対象としたが、OS の基本操作等でつまずく学生もいて、GAP を利用する前に学習すべきところが多かった。
- **実施結果:** 受講学生 40 人中の半数が 3 回目の講義から来なくなった。残った学生も計算機に対する基本知識に差があり、プログラムの基本的な部分が修得できてないため、与えられたプログラムを打ち込むのが精一杯で、自分でプログラムを構成できない学生がかなり見受けられた。
- **感想:** 学生の感想を聞いたところ、「プログラムだけでも難しいのに、良く分かっていない群論と組み合わせても、与えられたプリントを入力することで手一杯でとても難しかった。」

まとめ& お知らせ

まとめ& お知らせ

計算機を使った実習において、エディタの使い方、ファイルの読込・保存方法、プログラムの基本などの前提が必要であった。

この前提がない場合、「計算機を用いた実習」より、「計算機を利用した教材作り」の方がより良い結果を得られるようです。

まとめ& お知らせ

計算機を使った実習において、エディタの使い方、ファイルの読込・保存方法、プログラムの基本などの前提が必要であった。

この前提がない場合、「計算機を用いた実習」より、「計算機を利用した教材作り」の方がより良い結果を得られるようです。

第７回「代数学と計算」研究集会（AC2007）

2007年12月５日（水）- ７日（金）

首都大学東京 国際交流会館

詳しくは、

http://tnt.math.metro-u.ac.jp/ac/2007/