Paket Teori Grup pada Maple [*)]

Sahid

Laboratorium Komputer Jurdik Matematika FMIPA UNY

6 April 2002

Pendahuluan

Program Maple merupakah salah satu paket aplikasi komputer untuk melakukan berbagai komputasi matematis baik secara numerik maupun analitik. Dengan menggunakan Maple kita dapat menyelesaikan berbagai masalah matematika, mulai dari aritmetika dasar sampai aljabar abstrak. Diantara paket-paket yang tersedia di dalam program Maple adalah paket Teori Grup (**group**), Teori Bilangan (**numbertheory**) dan Kombinatorial (**combinat**), yang dapat digunakan untuk menyelesaikan beberapa masalah dalam teori grup.

Artikel (Tutorial) ini memberikan pengantar dan dasar-dasar penggunaan paket Teori Grup pada Maple. Pada bagian-bagian selanjutnya dijelaskan beberapa fungsi (perintah) Maple dan kegunaannya dalam kaitannya dengan masalah-masalah teori grup, khususnya untuk komputasi grup permutasi. Dengan menguasai dasar-dasar pemakaian paket **group**, seseorang yang menekuni bidang Aljabar Abstrak dapat menggunakan Maple sebagai alat bantu penyelidikan sifat-sifat grup.

Mengapa dengan Komputer?

- Untuk menggalakkan penggunaan komputer guna membantu melakukan pekerjaan matematika kita (*Doing Mathematics with Computer*)
- Sudah tersedia berbagai paket (program) komputer untuk matematika, baik yang komersial maupun gratis.
- Hampir setiap bidang dalam matematika, baik yang abstrak, komputasi, maupun visualisasi, bahkan proses pembuktian teorema, sudah dapat dikerjakan dengan komputer.
- Dengan komputer lebih banyak eksplorasi pengetahuan matematika yang dapat dilakukan dan akan mempercepat proses pengenalan konsep-konsep matematika, baik yang dasar maupun lanjut.
- Dengan komputer kita tidak perlu melakukan komputasi rutin yang akan banyak memakan waktu, sehingga waktu kita dapat digunakan untuk penyelidikan yang lebih luas dan mendalam.

Cara Pemakaian Paket Group

[*)] Makalah ini disampaikan dalam Seminar Nasional Aljabar - Jurdik Matimatika FMIPA UNY, 6 April 2002

Paket Maple yang berkaitan dengan teori grup disebut **group**. Paket ini terdiri atas beberapa **fungsi** (perintah) Maple yang berkitan dengan teori grup. Untuk dapat menggunakan perintah-perintah tersebut, paket **group** harus dipanggil terlebih dahulu melalui baris perintah Maple. Sintaks umum pemakainnya adalah sebagai berikut:

```
  with(group)
fungsi(args)
group[fungsi](args)
```

Perintah with(group)digunakan sekali, pada bagian awal. Setelah memanggil paket **group** tersebut dengan perintah **with**, selanjutnya kita dapat menggunakan fungsi dalam paket group dengan sintaks kedua, fungsi(args).

Contoh -- *menentukan order suatu grup permutasi*

Berikut adalah contoh menghitung order (cacah elemen) suatu grup permutasi berderajad 8 yang dihasilkan oleh permutasi **a** dan **b**.

```
> with(group):
grouporder(permgroup(8, {a=[[1,2]], b=[[1,2,3,4,5,6,7,8]]}));
```

40320

Macam-macam Fungsi dalam Paket Group

- Untuk menggunakan suatu **fungsi** group, kita dapat menggunakan perintah: with(group,fungsi) -- untuk mendefinisikan sebuah fungsi, atau dengan perintah with(group) -- untuk mendefinisikan semua fungsi yang ada.
- Berikut adalah daftar **fungsi** dan kegunaannya yang terdapat di dalam paket **group** pada Maple 7, diurutkan sesuai abjad:

areconjugate	menentukan apakah dua permutasi saling konjugat
center	mencari sentral sebuah grup permutasi
centralizer	mencari penyentral dari sekumpulan permutasi
convert	mengubah notasi permutasi dari bentuk daftar/kata ke bentuk sikel yang saling asing
core	mencari inti subgrup dari grup permutasi
cosets	mendaftar semua koset kanan dari subgrup dari grup permutasi atau grup yang dihasilkan oleh suatu generator terhadap suatu relasi
cosrep	menyatakan suatu elemen grup sebagai hasilkali elemen sebuah subgrup dan suatu koset kanan yang mewakili subgrup tersebut
derived	mencari subgrup turunan dari suatu grup permutasi
DerivedS	mencari barisan turunan dari suatu grup permutasi
elements	menghasilkan elemen-elemen suatu grup permutasi
grelgroup	menyajikan sebuah grup dengan generator dan relasi
groupmember	menguji apakah suatu permutasi anggota suatu grup
grouporder	menghitung order (cacah elemen) suatu grup
inter	mencari irisan dua buah grup permutasi
invperm	mencari invers suatu permutasi
isabelian	menentukan apakah suatu grup permutasi abelian
isnormal	menentukan apakah suatu subgrup merupakan normal
issubgroup	menentukan apakah suatu grup merupakah subgrup grup lain
LCS	mencari barisan sentral bawah suatu grup permutasi

mulperms	mengalikan dua permutasi dalam notasi sikel disjoin
NormalClosure	mencari klosur normal dari subgrup suatu grup permutasi
normalizer	mencari penormal suatu subgrup
orbit	menghitung orbit suatu titik
parity	mencari paritas suatu grup permutasi atau permutasi
permgroup	menyajikan suatu grup permutasi
permrep	mencari bentuk permutasi suatu grup
pres	mencari wakil suatu subgrup dari suatu grup
RandElement	mencari secara acak sebuah elemen suatu grup
SnConjugates	menghitung banyaknya elemen suatu grup yang sejenis dgn suatu sikel yang diberikan
subgrel	menyajikan suatu subgrup dari sebuah grup
Sylow	mencari subgrup Sylow-p dari suatu grup permutasi
transgroup	melacak informasi tentang grup permutasi transitif
type	menentukan apakah suatu permutasi berderajad tertentu dalam pengertian sikel disjoin

Berikut diberikan contoh-contoh pemakaian fungsi-fungsi pada paket **group**. Petunjuk pemakaian setiap perintah atau fungsi dapat dilihat dengan menuliskan perintah **?fungsi** pada baris perintah Maple atau melalui menu **Help**.

Mendefinisikan Group

permgroup - *mendefinisikan suatu grup permutasi yang berderajad n dengan menggunakan himpunan generator H, yang terdiri atas sikel-sikel disjoin.*

Dalam paket **group**, suatu permutasi dapat disajikan dengan notasi sikel disjoin, yang terdiri atas daftar sikel disjoin. Permutasinya adalah hasil kali sikel-sikel tersebut. Notasi **[a[1], a[2], ..., a[n]]** menyajikan permutasi yang memetakan **1** ke **a[1]**, **2** ke **a[2]**, ..., (**n-1**) ke **a[n-1]**, dan **n** ke **a[n], *sedangkan*** notasi **[[a[1], a[2], ..., a[n]]]** menyajikan permutasi yang memetakan **a[1]** ke **a[2]**, **a[2]** ke **a[3]**, ..., **a[n-1]** ke **a[n]**, dan **a[n]** ke **a[1]**. Elemen identitas disajikan dengan daftar kosong **[]**. Jadi, **[[1,2], [4,5]]** dan **[[5,4,3,2,1]]** menyajikan dua buah permutasi. Permutasi pertama merupakan hasilkali dua buah sikel disjoin dan memetakan 1->2, 2->1, 3->3, 4->5, dan 5->4, sedangkan permutasi kedua merupakan suatu sikel 5->4->3->2->1->5..

Paket **group** menggunakan aturan permutasi bekerja dari kiri, artinya jika **p1** dan **p2** adalah dua buah permutasi, maka hasilkali **p1** dan **p2,** yakni **(p1 * p2)** didefinisikan sedemikian hingga **(p1 * p2)(i) = p2(p1(i))** untukr **i=1..n.**

Contoh:

```
> G5:=permgroup(5, {a=[[1,2], [4,5]], b=[[5,4,3,2,1]]});
 G5 := permgroup( 5, {[[ 1, 4, 2, 3, 6]] = [[ 1, 2], [ 4, 5]], b = [[ 5, 4, 3, 2, 1]]} )

> G6:=permgroup(6, {[[1,2]], [[1,2,3,4,5,6]]});
 G6 := permgroup( 6, {[[ 1, 2]], [[ 1, 2, 3, 4, 5, 6]]} )

> grouporder(G5);
 60

> grouporder(G6);
 720
```

Cara berikut tidak dibenarkan:

```
> permgroup(5, {x=[[3,4], y=[7,2]]});
Error, (in permgroup) generators must represent products of disjoint cycles, but [[3, 4], y = [7, 2]] does not
```

grelgroup - *mendefinisikan suatu grup yang dihasilkan dari sebuah generator dan relasi antar elemen dalam generator*

Contoh:

```
> G:=grelgroup({a,b}, {[a,a,a], [b,b], [a,b,1/a,1/b]});
```

$$G := \text{grelgroup}\left(\{a, b\}, \{[a, a, a], [b, b], \left[a, b, \frac{1}{a}, \frac{1}{b}\right]\}\right)$$

```
> grouporder(G);
 6
```

Cara berikut salah:

```
> grelgroup({a,b}, {[a,1/c,a], [b,a]});
Error, (in grelgroup) relator [a, 1/c, a] contains something (1/c) not a generator or its inverse
```

subgrel - *menyajikan suatu subgrup dari sebuah grup yang dihasilkan oleh suatu generator*

Contoh:

```
> G:=grelgroup({a,b}, {[a,a,a], [b,b]});
```

$$G := \text{grelgroup}(\{a, b\}, \{[a, a, a], [b, b]\})$$

```
> subgrel({x=[a,b,1/a]},G);
```

$$\text{subgrel}\left(\{x = \left[a, b, \frac{1}{a}\right]\}, \text{grelgroup}(\{a, b\}, \{[a, a, a], [b, b]\})\right)$$

Cara berikut salah:

```
> subgrel({y=[a,b,c]}, grelgroup({a,b}, {[a,a], [b,a]}));
Error, (in subgrel) invalid arguments
```

permrep - *mencari bentuk permutasi suatu grup*

Contoh:

```
> G := grelgroup({x,y}, {[x,x,y,x,y,y,y],[y,y,x,y,x,x,x]});
```

$$G := \text{grelgroup}(\{x, y\}, \{[x, x, y, x, y, y, y], [y, y, x, y, x, x, x]\})$$

```
> H := subgrel({y=[y]},G);
```

$$H := \text{subgrel}(\{y = [y]\}, \text{grelgroup}(\{x, y\}, \{[x, x, y, x, y, y, y], [y, y, x, y, x, x, x]\}))$$

```
> PG:=permrep(H);
```

$$PG := \text{permgroup}(8, \{x = [[1, 2, 3, 7, 5, 6, 4]], y = [[2, 5, 8, 6, 7, 3, 4]]\})$$

```
> grouporder(PG);
 56
```

pres - *mencari wakil suatu subgrup dari suatu grup*

Hasilnya dinyatakan dalam bentuk fungsi **grelgroup** dengan generator subgrup tersebut dan relasi antar elemen-elemen dalam generator.

Contoh:

```
> G := grelgroup({a,b,c,d},
```

{[a,b,c,1/d],[b,c,d,1/a],[c,d,a,1/b],[d,a,b,1/c]});

$$G := \text{grelgroup}\left(\{a, b, d, c\}, \{\left[a, b, c, \frac{1}{d}\right], \left[b, c, d, \frac{1}{a}\right], \left[c, d, a, \frac{1}{b}\right], \left[d, a, b, \frac{1}{c}\right]\}\right)$$

> **SG := subgrel({x=[a,b],y=[a,c]},G);**

$$SG := \text{subgrel}\left(\{x = [a, b], y = [a, c]\}, \text{grelgroup}\left(\{a, b, d, c\}, \{\left[a, b, c, \frac{1}{d}\right], \left[b, c, d, \frac{1}{a}\right], \left[c, d, a, \frac{1}{b}\right], \left[d, a, b, \frac{1}{c}\right]\}\right)\right)$$

> **WSG:=pres(SG);**

$$WSG := \text{grelgroup}\left(\{x, y\}, \{\left[\frac{1}{x}, y, y, \frac{1}{x}, \frac{1}{y}, x, x, \frac{1}{y}\right], \left[y, \frac{1}{x}, \frac{1}{x}, \frac{1}{x}, y, x, \frac{1}{y}, x, \frac{1}{y}, x\right], \left[\frac{1}{x}, y, \frac{1}{x}, \frac{1}{y}, x, y, x, \frac{1}{y}\right]\}\right)$$

> **PSG:=permrep(SG);**

$$PSG := \text{permgroup}(4, \{d = [[1, 3, 4, 2]], b = [[1, 3, 4, 2]], a = [[1, 2, 4, 3]], c = [[1, 3, 4, 2]]\})$$

> **permrep(WSG);**

Error, (in group/permrep) invalid parameters.

> **grouporder(SG);**

Error, (in group/groupord) invalid parameters.

> **grouporder(PSG);**

4

> **elements(PSG);**

$$\{[\ ], [[1, 3, 4, 2]], [[1, 2, 4, 3]], [[1, 4], [2, 3]]\}$$

LCS - *mencari barisan sentral bawah suatu grup permutasi*

Setiap sentral dalam barisan yang dihasilkan dinyatakan dalam fungsi **permgroup**. Fungsi ini juga dapat digunakan untuk mencari kelas grup **G** dan menentukan apakah grup **G** bersifat **nilpoten** atau tidak.
Grup permutasi **G** dikatakan **nilpoten** jika **G_(k)=[]** untuk suatu **k>=1** dengan **G_(k)** adalah barisan subgrup **G** yang didefinisikan:

(a) **G_(1)** adalah subgrup komutator **G,** yakni subgrup terkecil **G** yang memuat semua elemen dalam bentuk **xyx^(-1)y^(-1)**, untuk semua **x** dan **y** anggota **G**.
(b) **G_(i)** adalah subgrup **G** yang dihasilkan oleh semua elemen dalam bentuk **aba^(-1)b^(-1)** dengan **a** anggota **G** dan **b** anggota **G_(i-1)**.

Contoh:

> **LCS(permgroup(5, {[[1,2,3,4,5]], [[2,5],[3,4]]}));**

$$[\text{permgroup}(5, \{[[1, 2, 3, 4, 5]], [[2, 5], [3, 4]]\}), \text{permgroup}(5, \{[\ ], [[1, 4, 2, 5, 3]]\})]$$

transgroup - *menampilkan informasi tentang grup permutasi transitif*

Cara Pemakaian

transgroup([d, n], nama)
transgroup("dTn", nama)
transgroup(d, nama)

dengan masukan:

d - derajad suatu grup permutasi transitif.

n - cacah kelas konjugasi grup tersebut.
nama - barisan nama yang menyatakan informasi yang diperlukan.

Penjelasan

- Fungsi ini menampilkan informasi yang ditentukan oleh barisan **nama** tentang grup permutasi transitif berderajad **d** dan memiliki kelas konjugasi sebanya **n**. Informasi yang dihasilkan berupa suatu barisan dengan elemen ke-**i** adalah informasi yang ditentukan oleh elemen ke-**i** pada barisan **nama**.
- Dua bentuk pertama fungsi **transgroup** tersebut menggunakan kedua nilai **d** dan **n**, sehingga hasilnya berupa informasi tentang sebuah kelas konjugasi khusus dari grup-grup transitif dalam grup simetris pada himpunan **{1, 2,...,d}**.

  Penomoran kelas mengikuti aturan yang dijelaskan pada paper "On Transitive Permutation Groups" oleh J.H. Conway, A. Hulpke, dan J. McKay, dalam *London Mathematical Society Journal of Computation and Mathematics*.

  Sebagai contoh, **[9,15]** adalah kelas ke-15 dari grup-grup berderajad 9, dan dapat ditulis dengan notasi lain yang ekivalen, **"9T15"**, seperti pada sintaks kedua.
- Masukan **nama** dapat berupa satu atau lebih dari string di bawah ini:

  'generators' : untuk menampilkan himpunan generator grup wakil kelas konjugasi tersebut,

  'names' : untuk menampilkan himpunan nama-nama grup sesuai Skema Penamaan Grup Transitif.

  'order' : untuk menampilkan order grup

  'parity' : untuk menampilkan paritas grup, 1 berarti genap, -1 berarti ganjil

  'SnConjugates' : untuk menampilkan barisan dengan elemen ke-**i** menyatakan cacah elemen grup berjenis sikel yang dinyatakan oleh partisi **partisi** = combinat[decodepart](**d, i).**

  'SnConjugates(partisi)': untuk menampilkan cacah elemen grup berjenis sikel yang dinyatakan dengan partisi **partisi**

Contoh:

```
> transgroup("5T3", 'names');
 {"F(5)", "5:4"}

> transgroup([5,3], 'names', 'parity');
 {"F(5)", "5:4"}, -1

> transgroup("6T12", 'order', 'parity', 'generators');
 60, 1, {[[1, 2, 3, 4, 6]], [[5, 6], [1, 4]]}

> transgroup(10, 'number');
 45

> transgroup(5,'number','order');
 5, [5, 10, 20, 60, 120]

> transgroup([4,3], 'SnConjugates');
 [1, 2, 3, 0, 2]

> transgroup([4,3],'SnConjugates([2,2])');
 3

> transgroup(6,'SnConjugates([2,2,2])');
 [1, 3, 4, 0, 3, 1, 0, 6, 6, 0, 7, 0, 6, 10, 0, 15]
```

Elemen-elemen suatu Grup dan Sifat-sifatnya

convert/disjcyc & **convert/permlist** - *mengubah notasi permutasi dari bentuk daftar/kata ke bentuk sikel yang saling asing dan sebaliknya*

Contoh:

```
> convert([3,4,1,2,7,6,5],'disjcyc');
 [[1, 3], [2, 4], [5, 7]]

> pg := permgroup(7, {a=[[1,2]], b = [[1,2,3,4,5,6,7]]}):
convert([a,b,1/a],'disjcyc',pg);
 [[1, 3, 4, 5, 6, 7, 2]]

> convert([[2,4,1], [7,3]],'permlist',7);
 [2, 4, 7, 1, 5, 6, 3]

> a:=convert([[2,3,4], [1,6]],'permlist',6);
 a := [6, 3, 4, 2, 5, 1]

> convert(a,'disjcyc');
 [[1, 6], [2, 3, 4]]
```

mulperms - *mengalikan dua buah permutasi dan hasilnya dinyatakan dalam notasi sikel-sikel disjoin*

Aturan perkalian dua buah permutasi adalah menggunakan permutasi pertama diikuti permutasi kedua.

Contoh:

```
> a:=mulperms([[2,3,4], [1,6]], [[4,6]]);
 a := [[1, 4, 2, 3, 6]]

> pl:=convert(a,'permlist',6);
 pl := [4, 3, 6, 2, 5, 1]

> convert(pl,'disjcyc');
 [[1, 4, 2, 3, 6]]
```

Hasil berikut bukan [[2,3]] melainkan [[1,3]].

```
> mulperms([[1,2]], [[1,2,3]]);
 [[1, 3]]
```

type/disjcyc - *menentukan apakah suatu permutasi dalam notasi sikel disjoin merupakan permutasi berderajad n, yakni permutasi dari n obyek*

Contoh:

```
> type([1,2,3,4,5],'disjcyc'(6));
 false

> convert([1,2,3,4,5],'disjcyc');
 [ ]

> type([[1,2,4],[3,9,6]],'disjcyc'(7));
 false

> convert([[1,2,4],[3,9,6]],'permlist',7);
Error, (in group/cyctolist) invalid arguments
```

```
> convert([[1,2,4],[3,9,6]],'permlist',9);
 [2, 4, 9, 1, 5, 3, 7, 8, 6]

> type([[1,2,4],[3,5,6]],'disjcyc'(7));
 true

> type([[1,2,4],[3,5,6]],'disjcyc'(6));
 true

> type([[1,2,4],[3,5,6]],'disjcyc'(5));
 false

> convert([[1,2,4],[3,5,6]],'permlist',6);
 [2, 4, 5, 1, 6, 3]

> convert([[1,2,4],[3,5,6]],'permlist',7);
 [2, 4, 5, 1, 6, 3, 7]

> convert([3,4,1,2,7,6,5],'disjcyc');
 [[1, 3], [2, 4], [5, 7]]

> G := permgroup(7, {a=[[1,2]], b = [[1,2,3,4,5,6,7]]}):
convert([a,b,1/a],'disjcyc',G);
 [[1, 3, 4, 5, 6, 7, 2]]

> convert([[2,4,1], [7,3]],'permlist',7);
```

grouporder - *menghitung order (cacah elemen) suatu grup*

Contoh:

```
> grouporder(permgroup(7, {[[1,2,3]], [[3,4,5,6,7]]}));
 2520

> grouporder(grelgroup({x,y}, {[x,x,y,x,y,y,y],[y,y,x,y,x,x,x]}));
 56
```

elements - *mendaftar/menampilkan elemen-elemen suatu grup permutasi*

Contoh:

Elemen-elemen suatu grup yang dihasilkan oleh dua buah permutasi:

```
> G := permgroup(4, {[[1,4],[2,3]], [[1,2],[3,4]]}):
elements(G);
 {[ ], [[1, 3], [2, 4]], [[1, 2], [3, 4]], [[1, 4], [2, 3]]}

> elements({[[1,4],[2,3]], [[1,2],[3,4]]});
 {[ ], [[1, 3], [2, 4]], [[1, 2], [3, 4]], [[1, 4], [2, 3]]}

> grouporder(G);
 4
```

groupmember - *menguji apakah suatu permutasi anggota suatu grup*

Contoh:

```
> G := permgroup(7, {[[1,2,3]], [[3,4,5,6,7]]}):
p:=[[1,5],[3,6]];
 p := [[1, 5], [3, 6]]

> groupmember(p,G);
 true
```

```
> groupmember([[3,2,4,7]], G);
 false
```

RandElement - *mencari secara acak sebuah elemen suatu grup*

Contoh:

```
> G := permgroup(5, {[[1,2,3,4,5]], [[2,5],[3,4]]});
 G := permgroup( 5, {[[ 1, 2, 3, 4, 5 ]], [[ 2, 5 ], [ 3, 4 ]]} )

> grouporder(G);
 10

> elements(G);
 {[  ], [[ 1, 5 ], [ 2, 4 ]], [[ 1, 2, 3, 4, 5 ]], [[ 2, 5 ], [ 3, 4 ]], [[ 1, 3 ], [ 4, 5 ]], [[ 1, 5, 4, 3, 2 ]], [[ 1, 4 ], [ 2, 3 ]],
      [[ 1, 2 ], [ 3, 5 ]], [[ 1, 3, 5, 2, 4 ]], [[ 1, 4, 2, 5, 3 ]]}

> e := RandElement(G);
 e := [[ 2, 5 ], [ 3, 4 ]]

> groupmember(e,G);
 true

 true
```

areconjugate - *menentukan apakah dua permutasi saling konjugat*

Misalkan **G** suatu grup yang memuat elemen-elemen **a** dan **b**. Kedua elemen tersebut saling konjugat jika terdapat elemen **c** dalam **G** yang bersifat $b = c^{(-1)}\, a\, b$.

Contoh:

Misalkan **G** adalah suatu grup permutasi berderajad 5 yang dihasilkan oleh permutasi-permutasi yang disajikan dengan sikel **a=(1,2)** dan **b=(1,2,3,4,5)**. Apakah permutasi-permutasi yang diwakili oleh sikel **d=(1,5,3)** dan **e=(2,4,5)** saling konjugat?

```
> G := permgroup(5, {[[1,2]],[[1,2,3,4,5]]});
 G := permgroup( 5, {[[ 1, 2, 3, 4, 5 ]], [[ 1, 2 ]]} )

> areconjugate(G, [[1,5,3]], [[2,4,5]]);
 true
```

invperm - *mencari invers suatu permutasi, hasilnya dinyatakan dalam bentuk notasi sikel-sikel disjoin*

Contoh:

```
> p:=[[4,6,5], [2,11]];
 p := [[ 4, 6, 5 ], [ 2, 11 ]]

> ip:=invperm(p);
 ip := [[ 2, 11 ], [ 4, 5, 6 ]]

> mulperms(ip,p);
 [  ]
```

orbit - *menghitung orbit suatu titik (obyek) terhadap suatu grup permutasi*

Hasilnya adalah himpunan titik-titik.

Contoh:

```
> G := permgroup(8, {[[1,2]], [[1,2,5,6,7,8]]});
 G := permgroup( 8, {[[ 1, 2, 5, 6, 7, 8 ]], [[ 1, 2 ]]} )

> P := {[[1,2]],[[5,1,8]]};
 P := {[[ 5, 1, 8 ]], [[ 1, 2 ]]}

> orbit(G, 1);
 {1, 2, 5, 6, 7, 8}

> orbit(G, 5);
 {1, 2, 5, 6, 7, 8}

> orbit(P, 1);
 {1, 2, 5, 8}

> orbit(G, 8);
 {1, 2, 5, 6, 7, 8}

> orbit(P, 5);
 {1, 2, 5, 8}
```

parity - *menentukan paritas suatu grup permutasi, permutasi, atau sebuah permutasi berjenis sikel yang diberikan oleh suatu partisi.*

Hasil fungsi **parity** adalah 1 jika paritasnya genap, -1 jika paritasnya ganjil.

Contoh:

```
> G1:=permgroup(6,{[[1,2,3]],[[2,3,4]],[[3,4,5]],[[4,5,6]]});
 G1 := permgroup( 6, {[[ 2, 3, 4 ]], [[ 3, 4, 5 ]], [[ 4, 5, 6 ]], [[ 1, 2, 3 ]]} )

> G2:=permgroup(6,{[[1,6]],[[2,6]],[[3,6]],[[4,6]],[[5,6]]});
 G2 := permgroup( 6, {[[ 5, 6 ]], [[ 3, 6 ]], [[ 4, 6 ]], [[ 1, 6 ]], [[ 2, 6 ]]} )

> parity(G1);
 1

> parity(G2);
 -1

> parity([[1,2],[3,4]]);
 1

> parity([2,2]);
 1

> parity([[1,2],[3,4,5]]);
 -1

> parity([2,3]);
 -1
```

SnConjugates - *menghitung banyaknya elemen suatu grup yang sejenis dengan*

jenis sikel suatu permutasi yang diberikan, dinyatakan dengan notasi sikel disjoin, atau partisi.

Jenis sikel suatu permutasi dapat dinyatakan dengan notasi sikel disjoin atau dengan suatu partisi bilangan yang menyatakan order grup permutasi yang bersangkutan.

Bilangan-bilangan **n1, n2, ..., nr** dengan **n1<=n2<=...<=nr** merupakan suatu **partisi** bilangan **n** jika **n=n1+n2+...+nr**. Suatu permutasi **p** dalam **Sn** dapat dinyatakan oleh partisi **{n1, n2, ..., nr}** dari **n** dengan **n1<=n2<=...<=nr** dan **n=n1+n2+...+nr** jika **p** adalah hasilkali sikel-sikel dijoint yang panjangnya **n1, n2, ...,**

nr. Misalnya, permutasi **[[1,2],[3,4],[5,6,7]]** dan partisi **[2,2,3]** merujuk jenis sikel yang sama.

Contoh:

```
> G:=permgroup(4, {[[1,4]], [[1,2],[3,4]]}):
SnConjugates(G,[[1,2],[3,4]]);
 3

> SnConjugates(G,[2,2]);
 3

> SnConjugates(G,[[1,2,3]]);
 0

> SnConjugates(G,[3]);
 0
```

Subgrup dan Sifat-sifatnya

issubgroup - *menentukan apakah suatu grup merupakah subgrup dari grup lain*

Contoh:

```
> G := permgroup(8, {[[1,2]], [[1,2,3,4,5,6,7,8]]}):
H := permgroup(8, {[[1,2,3,4]], [[1,2]], [[5,6,7,8]], [[5,6]]}):
issubgroup(H,G);
 true
```

isabelian - *menentukan apakah suatu grup permutasi abelian (komutatif)*

Contoh:

```
> G1 := permgroup(8, {[[1,2]], [[1,2,3,4,5,6,7,8]]}):
G2 := permgroup(8, {[[1,2]], [[3,4]]}):
isabelian(G1);
 false

> isabelian(G2);
 true
```

isnormal - *menentukan apakah suatu subgrup merupakan subgrup normal atauapakah suatu grup merupakan subgrup normal dari suatu grup permutasi yang dihasilkan oleh gabungan grup tersebut dengan grup lain yang sederajad.*

Subgrup **N** dari grup **G** dikatan **normal** jika **g*n*g^(-1)** adalah anggota **N** untuk setiap **g** dalam **G** dan **n** dalam **N**.

Contoh:

```
> G := permgroup(8, {[[1,2]], [[1,2,3,4,5,6,7,8]]});
 G := permgroup( 8, {[[ 1, 2 ]], [[ 1, 2, 3, 4, 5, 6, 7, 8 ]]} )

> H1 := permgroup(8, {[[1,2,3,4]], [[1,2]], [[5,6,7,8]], [[5,6]]});
 H1 := permgroup( 8, {[[ 5, 6 ]], [[ 5, 6, 7, 8 ]], [[ 1, 2, 3, 4 ]], [[ 1, 2 ]]} )

> isnormal(G,H1);
 false
```

```
> H2 := permgroup(8, {[[1,2,3]], [[2,3,4,5,6,7,8]]});
 H2 := permgroup( 8, {[[ 2, 3, 4, 5, 6, 7, 8 ]], [[ 1, 2, 3 ]]} )

> isnormal(G,H2);
 true

> G := grelgroup({a,b}, {[a,a,a,a,a], [b,b,b], [a,b,1/a,1/b]});
 G := grelgroup( {b, [[ 1, 4, 2, 3, 6 ]]}, {[b, b, b],
      [[[ 1, 4, 2, 3, 6 ]], [[ 1, 4, 2, 3, 6 ]], [[ 1, 4, 2, 3, 6 ]], [[ 1, 4, 2, 3, 6 ]], [[ 1, 4, 2, 3, 6 ]]],
      [ [[ 1, 4, 2, 3, 6 ]], b, 1/[[ 1, 4, 2, 3, 6 ]], 1/b ]} )

> isnormal(subgrel({x=[a]}, G));
 true
```

center - *mencari sentral sebuah grup permutasi*

Sentral suatu grup **G** adalah subgrup terbesar **Z** dari **G** yang memiliki elemen bersifat komutatif terhadap semua elemen **G.**

Contoh:

```
> G:=permgroup(5, {[[1,2,3,4,5]], [[2,5],[3,4]]});
 G := permgroup( 5, {[[ 1, 2, 3, 4, 5 ]], [[ 2, 5 ], [ 3, 4 ]]} )

> Z:=center(G);
 Z := permgroup( 5, { } )
```

centralizer - *mencari penyentral dari sekumpulan permutasi*

Misalkan **H** subgrup dari grup **G**. Penyentral **C(H)** dari **H** adalah himpunan semua elemen **G** yang bersifat komutatif terhadap semua elemen **H**. **C(H)** merupakan suatu subgrup **G.**

Contoh:

```
> G:=permgroup(7, {[[1,2]], [[1,2,3,4,5,6,7]]}):
H:= {[[3,6]]}:
> CH:=centralizer(G,H);
 CH := permgroup( 7, {[[ 1, 2, 5 ], [ 3, 6 ], [ 4, 7 ]], [[ 2, 4 ]], [[ 1, 2 ]]} )

> center(CH);
 permgroup( 7, {[[ 3, 6 ]]} )

> G:=permgroup(5, {[[1,2,3]], [[3,4,5]]}):
H:=[[1,2]]:
> CH:=centralizer(G,H);
 CH := permgroup( 5, {[[ 3, 4, 5 ]], [[ 1, 2 ], [ 4, 5 ]]} )

> center(CH);
 permgroup( 5, { } )
```

core - *mencari subgrup normal terbesar dari suatu grup permutasi G yang termuta di dalam subgrup (himpunan bagian) H*

Contoh:

```
> G := permgroup(7, {[[1,2]],[[1,2,3,4,5,6,7]]});
 G := permgroup( 7, {[[ 1, 2, 3, 4, 5, 6, 7 ]], [[ 1, 2 ]]} )
```

```
> H:= permgroup(7, {[[1,2,3]],[[3,4,5,6,7]]});
 H := permgroup( 7, {[[ 1, 2, 3 ]], [[ 3, 4, 5, 6, 7 ]]} )

> N:=core(H, G);
 N := permgroup( 7, {[[ 1, 2, 3 ]], [[ 3, 4, 5, 6, 7 ]]} )

> grouporder(G);
 5040

> grouporder(H);
 2520

> grouporder(N);
 2520
```

cosets - *mendaftar semua koset kanan dari subgrup H dari grup permutasi G atau grup yang dihasilkan oleh suatu generator terhadap suatu relasi*

Suatu koset kanan dari subgrup **H** pada grup **G** adalah himpunan **Hg={h*g| h** anggota **H}** dengan **g** anggota **G**. Hasil dari fungsi **cosets** adalah himpunan permutasi dalam notasi sikel-sikel disjoin.

Contoh:

```
> G := grelgroup({a,b,c}, {[a,b,c,a,1/b],[b,c,a,b,1/c],[c,a,b,c,1/a]});
```

$$G := \text{grelgroup}\left(\{b, c, [[1, 4, 2, 3, 6]]\}, \{\left[c, [[1, 4, 2, 3, 6]], b, c, \frac{1}{[[1, 4, 2, 3, 6]]}\right], \left[[[1, 4, 2, 3, 6]], b, c, [[1, 4, 2, 3, 6]], \frac{1}{b}\right], \left[b, c, [[1, 4, 2, 3, 6]], b, \frac{1}{c}\right]\}\right)$$

```
> cosets(subgrel({y=[a,b,c]}, G));
 {[ ], [[[ 1, 4, 2, 3, 6 ]]], [[[ 1, 4, 2, 3, 6 ]], b ]}

> G1 := permgroup(7, {[[1,2]], [[1,2,3,4,5,6,7]]});
 G1 := permgroup( 7, {[[ 1, 2, 3, 4, 5, 6, 7 ]], [[ 1, 2 ]]} )

> G2 := permgroup(7, {[[1,2,3]], [[3,4,5,6,7]]});
 G2 := permgroup( 7, {[[ 1, 2, 3 ]], [[ 3, 4, 5, 6, 7 ]]} )

> cosets(G1,G2);
 {[ ], [[ 6, 7 ]]}
```

Fungsi **cosets** dapat digunakan untuk menghasilkan semua elemen sebuah grup dengan mencari koset dari elemen identitas grup.

```
> G := permgroup(4, {[[1,2]], [[1,4]]} );
 G := permgroup( 4, {[[ 1, 4 ]], [[ 1, 2 ]]} )

> E := permgroup (4, {[]} );
 E := permgroup( 4, {[ ]} )

> cosets(G, E);
 {[ ], [[ 1, 4 ]], [[ 1, 4, 2 ]], [[ 1, 2, 4 ]], [[ 2, 4 ]], [[ 1, 2 ]]}

> grouporder(G);
 6
```

cosrep - *menyatakan suatu elemen grup, g, sebagai hasilkali elemen sebuah subgrup H dan suatu koset kanan yang mewakili subgrup H tersebut*

Hasilnya adalah sebuah list (daftar) yang terdiri atas dua buah elemen. Elemen pertama adalah elemen subgrup **H** tersebut yang dinyatakan sebagai sebuah kata dalam generator subgrup **H** atau berupa sebuah permutasi dalam subgrup **H** tersebut. Elemen kedua adalah penyajian koset kanan dari subgrup **H** tersebut

yang berupa sebuah elemen dari himpunan **cosets(Sn,H)**, dengan **Sn** adalah grup simetris yang berderajad sama dengan **H**.

Contoh:

```
> with(group):
g := grelgroup({a,b,c}, {[a,b,c,a,1/b],[b,c,a,b,1/c],[c,a,b,c,1/a]}):
cosrep([c], subgrel({y=[a,b,c]}, g));
 [[y, y, y, y, y], [[[1, 4, 2, 3, 6]]]]

> pg := permgroup(7, {[[1,2,3]], [[3,4,5,6,7]]}):
cosrep([[3,4,5,6]],pg);
 [[[3, 4, 5, 7, 6]], [[6, 7]]]
```

derived - *mencari subgrup dari suatu grup permutasi*

Contoh:

```
> G:=permgroup(5, {[[1,2,3,4,5]], [[2,5],[3,4]]});
 G := permgroup( 5, {[[ 1, 2, 3, 4, 5 ]], [[ 2, 5 ], [ 3, 4 ]]} )

> derived(G);
 permgroup( 5, {[  ], [[ 1, 4, 2, 5, 3 ]]} )

> issubgroup(derived(G),G);
 true
```

DerivedS - *mencari barisan subgrup dari suatu grup permutasi G*

Fungsi **DerivedS** dapat digunakan untuk menentukan apakah **G** bersifat *solvable*, yakni **terdapat subgrup-subgrup G=N_0 memuat N_1 memuat N_2 ... memuat N_r-[] sedemikian hingga N_i merupakan normal dalam N_(i-1) dan himpunan koset kanan N_i dalam N_(i-1), N_(i-1)|N_i, membentuk subgrup abelian.**
Hasilnya disajikan dalam fungsi **permgroup**.

Contoh:

```
> DerivedS(permgroup(5, {[[1,2,3,4,5]], [[2,5],[3,4]]}));
 [permgroup( 5, {[[ 1, 2, 3, 4, 5 ]], [[ 2, 5 ], [ 3, 4 ]]} ), permgroup( 5, {[  ], [[ 1, 4, 2, 5, 3 ]]} ),
       permgroup( 5, {[  ]} )]
```

inter - *mencari irisan dua buah grup permutasi berderajad sama*

Contoh:

```
> G1 := permgroup(7, {[[2,3,4]], [[3,4,5,6,7]]}):
G2 := permgroup(7, {[[1,2]], [[1,2,3,4,5,6]]}):
G3 := inter(G1,G2);
 G3 := permgroup( 7, {[[ 2, 4, 3 ]], [[ 2, 4, 6 ]], [[ 2, 4, 5 ]]} )

> grouporder(G1);
 360

> grouporder(G2);
 720

> grouporder(G3);
 60
```

normalizer - *mencari subgrup terbesar di dalam grup G yang memuat subgrup N sebagai subgrup normal*

Contoh:

```
> G := permgroup(7, {[[1,2,3]], [[3,4,5,6,7]]});
 G := permgroup( 7, {[[ 1, 2, 3 ]], [[ 3, 4, 5, 6, 7 ]]} )

> N := permgroup(7, {[[1,2,3]],[[3,4,5]]});
 N := permgroup( 7, {[[ 3, 4, 5 ]], [[ 1, 2, 3 ]]} )

> H:=normalizer(G,N);
 H := permgroup( 7, {[[ 4, 5 ], [ 6, 7 ]], [[ 3, 4, 5 ]], [[ 1, 2, 3 ]]} )

> grouporder(G);
 2520

> grouporder(N);
 60

> grouporder(H);
 120
```

NormalClosure - *mencari klosur normal dari subgrup suatu grup permutasi*

Hasilnya adalah subgrup normal terkecil dari grup permutasi **G** yang memuat subgrup **H** dinyatakan dalam fungsi **permgroup**.

Contoh:

```
> G := permgroup(7, {[[1,2],[[1,2,3,4,5,6,7]]});
> H := permgroup(7,{[[1,2,3]],[[3,4,5,6,7]]});
 H := permgroup( 7, {[[ 1, 2, 3 ]], [[ 3, 4, 5, 6, 7 ]]} )

> NormalClosure(H, G);
 permgroup( 7, {[[ 1, 2, 3 ]], [[ 3, 4, 5, 6, 7 ]]} )
```

Sylow - *mencari subgrup Sylow-p dari suatu grup permutasi*

Suatu subgrup **H** dari grup **G,** yang berorder **p^m** dengan **p^m** membagi **o(G)** tetapi **p^(m+1)** tidak membagi **o(G)** disebut **subgrup Sylow-p**.

Cara Pemakaian

Sylow(G, p)

dengan masukan:

G - suatu grup di mana subgrup Sylow-p hendak dicari

p - suatu pembagi prima dari order **G**

Penjelasan

- Fungsi menghasilkan sebuah p-grup maksimal, yang termuat di dalam grup permutasi **G,** dinyatakan dalam bentuk fungsi **permgroup**. Grup permutasi **G** harus memilikil order dan derajad kecil.

Contoh:

```
> with(group):
Sylow(permgroup(5, {[[1,2]], [[1,2,3,4,5]]}), 2);
 permgroup( 5, {[[ 1, 5 ]], [[ 2, 4 ]], [[ 1, 2 ], [ 4, 5 ]]} )
```

Penutup

Telah diuraikan cara pemakaian paket **Grup** pada Maple dan beberapa perintah yang dapat diguna-kan untuk membantu penyelesaian masalah-masalah dalam teori grup. Pada dasarnya perintah-perintah dalam paket **Grup** dapat dikelompokkan menjadi tiga bagian, yakni (1) perintah-perintah untuk mendefinisikan suatu grup atau subgrup, (2) perintah-perintah untuk mengetahui sifat-sifat elemen suatu grup (atau subgrup), dan perintah-perintah untuk mengetahui sifat-sifat suatu grup (atau subgrup).

Dengan bantuan paket **Grup** kita dapat mengerjakan soal-soal atau menyelesaikan masalah-masalah dalam teori grup secara mudah. Meskipun demikian, untuk dapat menggunakan perintah-perintah Maple tersebut, setiap grup harus dinyatakan sebagai grup permutasi dan setiap elemen suatu grup dinyatakan sebagai sebuah permutasi. Oleh karena setiap grup berhingga dapat dinyatakan sebagai suatu grup permutasi, maka pada akhirnya, paket grup dapat digunakan untuk bekerja dengan sebarang grup abstrak dalam matematika.

Kemampuan Maple tersebut dapat digunakan untuk membantu dosen dan mahasiswa mempelajari dan mengkaji konsep-konsep dalam teori grup secara lebih mendalam, dengan tanpa harus banyak membuat waktu untuk pekerjaan komputasi rutin. Dengan sedikit kemapuan pemrograman Maple, kemampuan dasar pada paket **Grup** dapat dikembangkan untuk bekerja dengan konsep-konsep grup yang lain, yang tidak tercakup secara langsung oleh paket tersebut. Pemrograman Maple bersifat sangat sederhana, sehingga mudah dipelajari. Mudah-mudahan tulisan sederhana ini dapat memberikan informasi dan motivasi kepada para pihak yang banyak bekerja dengan teori grup untuk dapat menggunakan Maple sebagai alat bantu.

Daftar Pustaka

Herstein,I.N. *Topics in Algebra*.second edition. John Wiley & Sons. Singapore, 1975

Waterloo Maple Inc. *Maple 7 User Guide*. http://www.maplesoft.com. Waterloo, 2001