

Permutation Groups

May 4, 2022

Contents

1	Permutations	3
1.1	Question 1	3
2	Groups	3
2.1	Question 2	3
2.2	Question 3	3
3	Orbit and Stabilizer	4
3.1	Question 4	4
3.2	Question 5 - add tables	4
4	Schreier's Theorem and the Final Algorithm	5
4.1	Question 6	5
4.2	Question 7 - comment on complexity	5
4.3	Question 8 - add tables	6
4.4	Question 9 - add tables	6
5	Code	8
5.1	Question 1	8
5.2	Question 3	8
5.3	Question 5	11
5.4	Question 7	13
5.5	Question 8	22
5.6	Question 9	30

Preface

This is my CATAM project, 16.5 for part II. The code for each question can be found in section 5.

1 Permutations

1.1 Question 1

See below tables for outputs of my programs to calculate the inverse and product of permutations of $\{1, \dots, n\}$. Note permutations are given as column vectors where the vector $(a_1 \dots a_n)^T$ corresponds to the permutation $i \mapsto a_i$.

π_1	π_2	$\pi_1 \circ \pi_2$
$\begin{pmatrix} 1 \\ 3 \\ 4 \\ 5 \\ 2 \end{pmatrix}$	$\begin{pmatrix} 2 \\ 4 \\ 3 \\ 5 \\ 1 \end{pmatrix}$	$\begin{pmatrix} 3 \\ 5 \\ 4 \\ 2 \\ 1 \end{pmatrix}$
$\begin{pmatrix} 2 \\ 3 \\ 1 \end{pmatrix}$	$\begin{pmatrix} 1 \\ 3 \\ 2 \end{pmatrix}$	$\begin{pmatrix} 2 \\ 1 \\ 3 \end{pmatrix}$

Table 1: Products of Permutations

π	π^{-1}
$\begin{pmatrix} 1 \\ 5 \\ 4 \\ 2 \\ 6 \\ 3 \end{pmatrix}$	$\begin{pmatrix} 1 \\ 4 \\ 6 \\ 3 \\ 2 \\ 5 \end{pmatrix}$
$\begin{pmatrix} 2 \\ 4 \\ 3 \\ 1 \end{pmatrix}$	$\begin{pmatrix} 4 \\ 1 \\ 3 \\ 2 \end{pmatrix}$

Table 2: Inverses of Permutations

The complexity of the program to calculate inverses is $O(n^2)$ while the complexity of the program to calculate products is $O(n)$.

2 Groups

2.1 Question 2

Let \tilde{G} denote the group generated by the permutations outputted by the Stripping Algorithm of Sims. Firstly we have $\tilde{G} \leq G$ since all elements in \tilde{G} are generated by elements in G . Then given any $g \in G$, there must exist $g_1, \dots, g_l \in \tilde{G}$ such that

$$g_l^{-1} \dots g_1^{-1} g = e \text{ or } g' \in \tilde{G}.$$

In either case we have g is equal to a product of elements in \tilde{G} and so itself is in \tilde{G} . Hence we have $G = \tilde{G}$.

Given $G = \langle g_1, \dots, g_k \rangle$, we must have $|\tilde{G}| \leq k$. This is because at most each g_k corresponds to a single element in \tilde{G} . The complexity of the algorithm is $O(n^5)$.

2.2 Question 3

See below a table of outputs for the Stripping Algorithm of Sims program. Note the input given as a matrix with each column corresponding to a generator. Further each column vector is written as the permutation

sending $i \mapsto v_i$ with v_i the i^{th} entry of the column v .

Original Generators	Reduced Set of Generators
$\begin{pmatrix} 3 & 4 & 2 & 2 & 4 \\ 1 & 1 & 4 & 3 & 2 \\ 4 & 2 & 3 & 4 & 1 \\ 2 & 3 & 1 & 1 & 3 \end{pmatrix}$	$\begin{pmatrix} 0 & 3 & 1 & 2 \\ 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$
$\begin{pmatrix} 3 & 1 & 3 \\ 2 & 2 & 1 \\ 1 & 3 & 2 \end{pmatrix}$	$\begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 3 \\ 0 & 0 & 0 \end{pmatrix}$

Table 3: Output of Stripping Algorithm

3 Orbit and Stabilizer

3.1 Question 4

For G acting on a set Ω and $\alpha \in \Omega$, there is a bijection between G/G_α and $G\alpha$ as follows

$$\begin{aligned} G/G_\alpha &\rightarrow G\alpha \\ gG_\alpha &\mapsto g\alpha \end{aligned}$$

The orbit stabiliser theorem is the above statement. For finite groups there is a corollary of this theorem which is often referred to as the Orbit-Stabiliser Theorem. Under the same set-up as the previously given bijection, this corollary states

$$|G\alpha| = \frac{|G|}{|G_\alpha|}.$$

3.2 Question 5 - add tables

See below for a table of my program which calculates the orbit of an element, along with corresponding witnesses, under the action of permutation group. Note that the orbit of a is given as a row vector, with the entries being the orbit of a . The witnesses are given as a matrix with the i^{th} column being the witness of the i^{th} entry in the orbit vector. The format of the permutation column vectors remains the same as in previous questions.

Generators	Element	Orbit	Witness
$\begin{pmatrix} 3 & 1 & 4 & 4 \\ 1 & 3 & 1 & 1 \\ 4 & 4 & 1 & 2 \\ 2 & 2 & 3 & 3 \end{pmatrix}$	3	(3 1 4 2)	$\begin{pmatrix} 1 & 4 & 1 & 4 \\ 2 & 3 & 3 & 1 \\ 3 & 1 & 4 & 2 \\ 4 & 2 & 2 & 3 \end{pmatrix}$
$\begin{pmatrix} 4 & 1 \\ 2 & 2 \\ 5 & 5 \\ 1 & 4 \\ 3 & 3 \end{pmatrix}$	1	(1 4)	$\begin{pmatrix} 1 & 4 \\ 2 & 2 \\ 3 & 5 \\ 4 & 1 \\ 5 & 3 \end{pmatrix}$

Table 4: Orbit of an Element Under a Permutation Group

Let $G = \langle g_1, \dots, g_k \rangle$. My program works by noting that any element in the orbit of α will be given by a finite combination of g_i applied α in some order. Hence by calculating the image of α under each g_i and recording some g_i gives rise to each image, then iterating that process for each new element in the image

will give us the orbit of x along with witnesses. Further the program will terminate as by finiteness of Ω , the set G acts on, we will eventually reach every element in the orbit. In which case applying each g_i to this set will give us no new elements and so the program will terminate.

4 Schreier's Theorem and the Final Algorithm

4.1 Question 6

Note that by the definition of t_i we have

$$\begin{aligned} t_1 G_\alpha &= G_\alpha \\ t_{i+1} G_\alpha &= y_i t_i G_\alpha. \end{aligned}$$

Hence we have

$$\begin{aligned} t_{r+1} G_\alpha &= y_r \cdots y_1 t_1 G_\alpha \\ &= x t_1 G_\alpha \\ &= G_\alpha. \end{aligned}$$

Further, we have $t_{r+1} \in \text{im}(\varphi) = T$ and so $t_{r+1} = t_1$. Then for any $x \in G_\alpha$ we have

$$\begin{aligned} x &= y_r \cdots y_1 \\ &= t_{r+1} (t_{r+1}^{-1} y_r t_r) \cdots (t_2^{-1} y_1 t_1) t_1^{-1} \\ &= t_1 (\varphi(y_r t_r)^{-1} y_r t_r) \cdots (\varphi(y_1 t_1)^{-1} y_1 t_1) t_1^{-1}. \end{aligned}$$

Letting $\Lambda = \langle \varphi(yt)^{-1} yt : y \in Y, t \in T \rangle$ and applying this to $x = t_1$ gives $t_1 \in \Lambda$. Hence for any $x \in G_\alpha$ we have

$$x \in t_1 \Lambda t_1^{-1} \subset \Lambda.$$

It follows that $G_\alpha \subset \Lambda$. Then $(\forall y \in Y)(\forall t \in T)$ we have

$$\begin{aligned} \varphi(yt) G_\alpha &= yt G_\alpha \\ \Rightarrow \varphi(yt)^{-1} yt &\in G_\alpha. \end{aligned}$$

Then the reverse inclusion follows and we must have $G_\alpha = \Lambda$.

4.2 Question 7 - comment on complexity

See below for a table of a set of generators of the stabiliser of an element of a permutation group.

Generator	Element	Stabiliser
$\begin{pmatrix} 2 & 2 \\ 1 & 4 \\ 3 & 3 \\ 4 & 1 \end{pmatrix}$	2	$\begin{pmatrix} 4 \\ 2 \\ 3 \\ 1 \end{pmatrix}$
$\begin{pmatrix} 2 & 5 & 1 \\ 4 & 4 & 4 \\ 5 & 1 & 2 \\ 1 & 2 & 5 \\ 3 & 3 & 3 \end{pmatrix}$	4	$\begin{pmatrix} 2 & 3 & 1 & 1 & 1 \\ 3 & 1 & 3 & 5 & 2 \\ 5 & 2 & 5 & 2 & 5 \\ 4 & 4 & 4 & 4 & 4 \\ 1 & 5 & 2 & 3 & 3 \end{pmatrix}$

Table 5: Stabiliser of an Element Under a Permutation Group

4.3 Question 8 - add tables

See below for a table of the size of the group generated by a given set of permutations.

Generators	Group Size
$\begin{pmatrix} 1 & 2 \\ 3 & 1 \\ 2 & 3 \end{pmatrix}$	6
$\begin{pmatrix} 1 & 1 & 4 \\ 3 & 3 & 3 \\ 4 & 2 & 1 \\ 2 & 4 & 2 \end{pmatrix}$	24

Table 6: Size of Permutation Group Generated by Given Set of Permutations

If we didn't run the stripping algorithm at every stage in our program, the efficiency of the program would be greatly reduced. This is because at each stage we would (in most cases) be increasing our number of generators for the group. This would then lead to an increase in time to calculate the orbit and witnesses. This would then further increase the time to calculate the stabiliser and further the number of generators of the stabiliser. This would then spill over into the next repeat of the algorithm, exponentially increasing the number of calculations and storage required for the program to run. For $n = 20$ this would be a massive increase in time to run the program.

4.4 Question 9 - add tables

From IA, we know that $S_n = \langle (1\ 2), (1\ 2\ \dots\ n) \rangle$, hence for all n we must have $P_n > 0$. Further note that $A_n \leq S_n$ and

$$|A_n| = \frac{|S_n|}{2},$$

so the probability of choosing two elements at random from A_n is $\frac{1}{4}$, hence we have for all n that $P_n \leq \frac{3}{4}$. Since if we choose both elements from A_n then the subgroup they generate is contained in A_n which is strictly contained in S_n .

Since $S_2 = C_2$, $P_2 = \frac{3}{4}$. Then for $n = 3$ we have that S_3 is generated by any 3 and 2 cycle, any two 2 cycles and these are precisely the combinations of two permutations which work. To prove this, first note for any $a \neq b \neq c \neq a$ $(a\ b\ c), (a\ b)$ generate S_3 as previously discussed. Then we have

$$\begin{aligned} (a\ b\ c)(a\ c) &= (b\ c) \text{ and so } S_3 \text{ is generated by any 2 and 3 cycle,} \\ (a\ b)(b\ c) &= (a\ b\ c) \text{ and so any two different 2 cycles generate } S_3, \\ (a\ b\ c)(a\ c\ b) &= (a\ b\ c)(a\ b\ c) = e \text{ and so any two 3 cycles don't generate } S_3. \end{aligned}$$

Further S_3 can't be generated by any single permutation and so we have the desired result. By counting the possible generators, it then follows

$$P_3 = \frac{18}{36} = \frac{1}{2}.$$

My program generates random permutations by setting $\sigma(1) =$ a random number in $\{1, \dots, n\}$. Then iteratively generating a new random integer l in $\{1, \dots, n\}$ and if it is not equal to any previous $\sigma(i')$ then set $\sigma(i) = l$.

See below for my estimates of P_n for $5 \leq n \leq 10$.

n	5	6	7	8	9	10
P_n	0.35	0.38	0.54	0.56	0.63	0.63

Table 7: Values of P_n for various n

5 Code

5.1 Question 1

```
1 function [outputArg1,outputArg2] = timespi(pia,pib)
2 n=size(pia,1);
3
4
5 i=1;
6
7 pipiab=zeros(n,1);
8
9 while i<= n
10     pipiab(i)=pia(pib(i));
11     i=i+1;
12 end
13 pipiab
14
15
16 end
```

```
1 function [outputArg1,outputArg2] = invpi(pi)
2 n=size(pi,1);
3
4 i=1;
5
6 inv=zeros(n,1);
7 while i<= n
8     term=0;
9     j=1;
10     while term == 0
11
12         if pi(j)==i;
13             inv(i)=j;
14             term=1;
15         else
16             j=j+1;
17         end
18     end
19     i=i+1;
20 end
21 inv
22
23
24 end
```

5.2 Question 3

```
1 function [outputArg1,outputArg2] = stripping(Pi)
2
3 %input(pi_1,pi_2, ... ,pi_k)
4 %Going to have to change input to a matrix with each collumn being a
   generator
```



```

5
6 k=size(Pi,2);
7
8 n=size(Pi,1);
9
10 l=1;
11
12 A=zeros(n,n);
13
14 while l <= k
15
16     r=1;
17     finish=0;
18
19     while finish == 0
20
21         if r~=n
22
23             if Pi(r,l) ~= r
24
25                 if A(r,Pi(r,l)) == 0
26
27                     A(r,Pi(r,l))=1;
28                     finish=1;
29
30                 else
31
32                     pi=Pi(:,A(r,Pi(r,l)));
33                     %"run invert pi"
34
35                 i=1;
36
37                 inv=zeros(n,1);
38                 while i<= n
39                     term=0;
40                     j=1;
41                     while term == 0
42
43                         if pi(j)==i;
44                             inv(i)=j;
45                             term=1;
46                         else
47                             j=j+1;
48                         end
49                     end
50                     i=i+1;
51                 end
52
53                 pia=inv;
54                 pib=Pi(:,l);
55                 %"run multiply pia,pib program"
56
57                 i=1;
58

```

```

59         pipiab=zeros(n,1);
60
61         while i<= n
62             pipiab(i)=pia(pib(i));
63             i=i+1;
64         end
65         Pi(:,l)= pipiab;
66
67
68         %Ran multiply pia,pib program
69             r=r+1;
70
71         end
72
73         else
74
75             r=r+1;
76
77         end
78
79         else
80
81             finish = 1;
82
83         end
84
85     end
86     l=l+1;
87
88 end
89 A;
90 i=1;
91 kk=1;
92 Qi=zeros(n,1);
93 while i<= size(A,1)
94     j=1;
95     while j<= size(A,2)
96         if A(i,j) ~= 0
97             Qi(:, kk)=Pi(:,A(i,j));
98             kk=kk+1;
99         else
100             end
101             j=j+1;
102         end
103         i=i+1;
104     end
105     Pi=Qi;
106     Pi;
107     A
108
109
110
111
112

```

```
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142 end
```

5.3 Question 5

```
1 function [outputArg1,outputArg2] = orbit(A,e1)
2
3 k=size(A,2);
4 n=size(A,1);
5 orbit=[e1,0];
6 newsize=1;
7 oldsize=0;
8
9 while oldsize < newsize
10
11
12
13     i=oldsize+1;
14     oldsize=size(orbit,1);
15     c=1;
16
17     while i <= oldsize
18
19         j=1;
20         while j<= k
```

```

21         j;
22
23         if ismembertol(A(orbit(i,1),j),orbit(:,1))
24
25         else
26             orbit;
27             holdvec=[orbit(i,:),j];
28             orbit(:,size(orbit,2)+1)=zeros(size(orbit,1),1);
29             orbit(oldsizes+c,:)=holdvec;
30             orbit(oldsizes+c,1)=A(orbit(i,1),j);
31             c=c+1;
32         end
33         j=j+1;
34     end
35     i=i+1;
36 end
37 newsize=size(orbit,1);
38 end
39 orbit
40
41
42 %Calculate table of witnesses
43 xx=size(orbit,2);
44 orbsize=size(orbit,1);
45 i=2;
46 witness=[1:n].';
47 B=zeros(size(A,1),size(A,2)+1);
48 B(:,1)=[1:n];
49 B(:,[2:size(A,2)+1])=A;
50
51 while i<=orbsize
52
53     j=2;
54     witness(:,i)=B(:,1);
55     while j<= xx
56         %"do witness(:,i)=B(:,j+1)*witness(:,i)
57
58         pia=B(:,orbit(i,j)+1);
59         pib=witness(:,i);
60         pipiab=zeros(n,1);
61         ii=1;
62
63         while ii<= n
64             pipiab(ii)=pia(pib(ii));
65             ii=ii+1;
66         end
67         witness(:,i)=pipiab;
68         j=j+1;
69     end
70     i=i+1;
71 end
72 witness
73
74

```

```
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120 end
```

5.4 Question 7

```
1 function [outputArg1,outputArg2] = stabiliser(Y,a)
2
3
4 Pi=Y;
```

```

5  e1=a;
6
7  %Run stripping algorithm"
8
9      k=size(Pi,2);
10
11      n=size(Pi,1);
12
13      l=1;
14
15      A=zeros(n,n);
16
17      while l <= k
18
19          r=1;
20          finish=0;
21
22          while finish == 0
23
24              if r~=n
25
26                  if Pi(r,l) ~= r
27
28                      if A(r,Pi(r,l)) == 0
29
30                          A(r,Pi(r,l))=1;
31                          finish=1;
32
33                      else
34
35                          pi=Pi(:,A(r,Pi(r,l)));
36                          %"run invert pi"
37                          i=1;
38
39                          inv=zeros(n,1);
40                          while i<= n
41                              term=0;
42                              j=1;
43                              while term == 0
44
45                                  if pi(j)==i;
46                                      inv(i)=j;
47                                      term=1;
48                                  else
49                                      j=j+1;
50                                  end
51                                  end
52                                  i=i+1;
53                              end
54
55                          pia=inv;
56                          pib=Pi(:,l);
57                          %"run multiply pia,pib program"
58

```

```

59
60         i=1;
61
62         pipiab=zeros(n,1);
63
64         while i<= n
65             pipiab(i)=pia(pib(i));
66             i=i+1;
67         end
68         Pi(:,l)= pipiab;
69
70
71         %Ran multiply pia,pib program
72         r=r+1;
73
74         end
75
76         else
77
78             r=r+1;
79
80         end
81
82         else
83
84             finish = 1;
85
86         end
87
88     end
89     l=l+1;
90
91 end
92 A;
93 i=1;
94 kk=1;
95 Qi=zeros(n,1);
96 while i<= size(A,1)
97     j=1;
98     while j<= size(A,2)
99         if A(i,j) ~= 0
100             Qi(:, kk)=Pi(:,A(i,j));
101             kk=kk+1;
102         else
103             end
104             j=j+1;
105         end
106         i=i+1;
107     end
108     Pi=Qi;
109
110
111 %Ran stripping algorithm
112 Y=Pi;

```

```

113 y=size(Y,2);
114
115 A=Y;
116 %Run orbit algorithm;
117
118     k=size(A,2);
119     n=size(A,1);
120     orbit=[e1,0];
121     newsize=1;
122     oldsize=0;
123
124     while oldsize < newsize
125
126
127
128         i=oldsize+1;
129         oldsize=size(orbit,1);
130         c=1;
131
132         while i <= oldsize
133
134             j=1;
135             while j<= k
136                 j;
137
138                 if ismembertol(A(orbit(i,1),j),orbit(:,1))
139
140                     else
141                         orbit;
142                         holdvec=[orbit(i,:),j];
143                         orbit(:,size(orbit,2)+1)=zeros(size(orbit,1),1);
144                         orbit(oldsize+c,:)=holdvec;
145                         orbit(oldsize+c,1)=A(orbit(i,1),j);
146                         c=c+1;
147                     end
148                 j=j+1;
149             end
150             i=i+1;
151         end
152     newsize=size(orbit,1);
153     end
154     orbit;
155
156
157     %Calculate table of witnesses
158     xx=size(orbit,2);
159     orbsize=size(orbit,1);
160     i=2;
161     witness=[1:n].';
162     B=zeros(size(A,1),size(A,2)+1);
163     B(:,1)=[1:n];
164     B(:,[2:size(A,2)+1])=A;
165

```



```

166         while i<=orbsize
167
168             j=2;
169             witness(:,i)=B(:,1);
170             while j<= xx
171                 %"do witness(:,i)=B(:,j+1)*witness(:,i)
172
173                 pia=B(:,orbit(i,j)+1);
174                 pib=witness(:,i);
175                 pipiab=zeros(n,1);
176                 ii=1;
177
178                 while ii<= n
179                     pipiab(ii)=pia(pib(ii));
180                     ii=ii+1;
181                 end
182                 witness(:,i)=pipiab;
183                 j=j+1;
184             end
185             i=i+1;
186         end
187         witness;
188
189
190
191
192
193 %Ran orbit program
194 t=size(orbit,1);
195 T=witness;
196
197
198
199
200
201
202
203 yi=1;
204
205
206 stab=zeros(n,1);
207
208 while yi <= y
209
210     ti=1;
211
212     while ti<= t
213
214
215         %Calculate Y(:,yi)*T(:,ti)
216
217         pia=Y(:,yi);
218         pib=T(:,ti);
219

```

```

220         %"Run timespi program"
221
222         i=1;
223
224         pipiab=zeros(n,1);
225
226         while i<= n
227             pipiab(i)=pia(pib(i));
228             i=i+1;
229         end
230
231
232
233         %Calculate varphi(piapib)
234
235
236         ne=pipiab(a);
237
238         i=1;
239         foundinverse=0;
240         while foundinverse==0
241             if orbit(i,1)==ne
242                 nu=i;
243                 foundinverse=1;
244             else
245                 i=i+1;
246             end
247         end
248
249
250         varphi=T(:,nu);
251
252         %Calculate inverse of varphi
253
254         pi=varphi;
255
256         %"run invpi program "
257
258         i=1;
259
260         inv=zeros(n,1);
261         while i<= n
262             term=0;
263             j=1;
264             while term == 0
265
266                 if pi(j)==i;
267                     inv(i)=j;
268                     term=1;
269                 else
270                     j=j+1;
271                 end
272             end
273             i=i+1;

```

```

274         end
275
276
277         %Calculate stab
278
279         pia=inv;
280         pib=piabiab;
281
282         %"Run timespi program"
283
284         i=1;
285
286         pipiab=zeros(n,1);
287
288         while i<= n
289             pipiab(i)=pia(pib(i));
290             i=i+1;
291         end
292
293
294
295         stab(:,(yi-1)*y+ti)=pipiab;
296
297
298
299
300         ti=ti+1;
301     end
302     yi=yi+1;
303 end
304
305 stab;
306 Pi=stab;
307
308 %Run stripping algorithm
309
310
311     k=size(Pi,2);
312
313     n=size(Pi,1);
314
315     l=1;
316
317     A=zeros(n,n);
318
319     while l <= k
320
321         r=1;
322         finish=0;
323
324         while finish == 0
325
326             if r~=n

```

```

328         if Pi(r,1) ~= r
329
330             if A(r,Pi(r,1)) == 0
331
332                 A(r,Pi(r,1))=1;
333                 finish=1;
334
335             else
336
337                 pi=Pi(:,A(r,Pi(r,1)));
338                 %"run invert pi"
339                 i=1;
340
341                 inv=zeros(n,1);
342                 while i<= n
343                     term=0;
344                     j=1;
345                     while term == 0
346
347                         if pi(j)==i;
348                             inv(i)=j;
349                             term=1;
350                         else
351                             j=j+1;
352                         end
353                     end
354                     i=i+1;
355                 end
356
357
358                 pia=inv;
359                 pib=Pi(:,1);
360                 %"run multiply pia,pib program"
361
362                 i=1;
363
364                 pipiab=zeros(n,1);
365
366                 while i<= n
367                     pipiab(i)=pia(pib(i));
368                     i=i+1;
369                 end
370                 Pi(:,1)= pipiab;
371
372
373                 %Ran multiply pia,pib program
374                 r=r+1;
375
376             end
377
378         else
379
380             r=r+1;
381

```

```

382         end
383
384     else
385
386         finish = 1;
387
388     end
389
390     end
391     l=l+1;
392
393     end
394     A;
395     i=1;
396     kk=1;
397     Qi=zeros(n,1);
398     while i<= size(A,1)
399         j=1;
400         while j<= size(A,2)
401             if A(i,j) ~= 0
402                 Qi(:, kk)=Pi(:,A(i,j));
403                 kk=kk+1;
404             else
405                 end
406                 j=j+1;
407             end
408             i=i+1;
409         end
410         Pi=Qi;
411
412
413
414
415
416
417 %Ran stripping algorithm
418
419 stab=Pi;
420
421 stab
422
423
424
425
426
427
428
429
430
431
432
433
434 end

```

5.5 Question 8

```
1 function [outputArg1,outputArg2] = permgroupsize(G)
2 %Finding the size of G
3 oldGsize(1)=size(G,2);
4 iii=1;
5 nn(iii)=1;
6 nontrivorb=0;
7
8
9 while nontrivorb == 0
10
11
12 %Step 1: Reduce generators of G
13     clear Pi
14     Pi=G;
15     %Run stripping algorithm
16     k=size(Pi,2);
17
18     n=size(Pi,1);
19
20     l=1;
21
22     A=zeros(n,n);
23
24     while l <= k && ~ismember(0,Pi)
25
26
27         r=1;
28         finish=0;
29
30         while finish == 0
31
32             if r~=n
33
34                 if Pi(r,l) ~= r
35                     Pi(r,l);
36                     if A(r,Pi(r,l)) == 0
37
38                         A(r,Pi(r,l))=1;
39                         finish=1;
40
41                     else
42
43                         pi=Pi(:,A(r,Pi(r,l)));
44                         %"run invert pi"
45                         i=1;
46
47                         inv=zeros(n,1);
48                         while i<= n
49                             term=0;
50                             j=1;
51                             while term == 0
52
```

```

53         if pi(j)==i;
54             inv(i)=j;
55             term=1;
56         else
57             j=j+1;
58         end
59     end
60     i=i+1;
61 end
62
63
64     pia=inv;
65     pib=Pi(:,1);
66     %"run multiply pia,pib program"
67
68     i=1;
69
70     pipiab=zeros(n,1);
71
72     while i<= n
73         pipiab(i)=pia(pib(i));
74         i=i+1;
75     end
76     pipiab;
77     Pi(:,1)= pipiab;
78
79
80     %Ran multiply pia,pib program
81     r=r+1;
82
83     end
84
85     else
86
87         r=r+1;
88
89     end
90
91     else
92
93         finish = 1;
94
95     end
96
97     end
98     l=l+1;
99
100 end
101 A;
102 i=1;
103 kk=1;
104
105 Qi=zeros(n,1);
106 while i<= size(A,1)

```

```

107         j=1;
108         while j<= size(A,2)
109             if A(i,j) ~= 0
110                 Qi(:, kk)=Pi(:, A(i,j));
111                 kk=kk+1;
112             else
113                 end
114             j=j+1;
115         end
116         i=i+1;
117     end
118     Qi;
119     Pi=Qi;
120     G=Pi;
121
122
123     newGsize(iii)=size(G,2);
124     %call reduced generators G too
125
126     if G(1,1)==0
127         nontrivorb=1;
128     end
129
130     if nontrivorb==0
131
132 %Step 2: Find non-trivial orbits of G
133
134         pi=G(:,1);
135         i=1;
136         foundel=0;
137         while foundel==0
138
139             if pi(i)==i
140                 i=i+1;
141             else
142                 foundel=1;
143                 a=i;
144             end
145         end
146
147
148
149
150         %Step 3: Find orbit of a
151
152         %Run stabiliser program
153
154         Y=G;
155         y=size(Y,2);
156
157         A=Y;
158         el=a;
159         %Run orbit algorithm;
160

```



```

161 k=size(A,2);
162 n=size(A,1);
163 orbit=[e1,0];
164 newsize=1;
165 oldsize=0;
166
167 while oldsize < newsize
168
169
170
171     i=oldsize+1;
172     oldsize=size(orbit,1);
173     c=1;
174
175     while i <= oldsize
176
177         j=1;
178         while j<= k
179             j;
180
181             if ismembertol(A(orbit
182                             (i,1),j),orbit(:,1)
183                             )
184
185                 else
186                     orbit;
187                     holdvec=[orbit(i
188                                 :,j)];
189                     orbit(:,size(orbit
190                                 ,2)+1)=zeros(
191                                 size(orbit,1)
192                                 ,1);
193                     orbit(oldsize+c,:)
194                         =holdvec;
195                     orbit(oldsize+c,1)
196                         =A(orbit(i,1),j
197                         );
198                     c=c+1;
199                 end
200             end
201             j=j+1;
202         end
203         i=i+1;
204     end
205     newsize=size(orbit,1);
206 end
207 orbit;
208
209 %Calculate table of witnesses
210 xx=size(orbit,2);
211 orbsize=size(orbit,1);
212 i=2;
213 witness=[1:n].';
214 B=zeros(size(A,1),size(A,2)+1);

```

```

206         B(:,1)=[1:n];
207         B(:, [2:size(A,2)+1])=A;
208
209         while i<=orbsize
210
211             j=2;
212             witness(:,i)=B(:,1);
213             while j<= xx
214                 %"do witness(:,i)=B(:,j+1)
215                 %*witness(:,i)
216
217                 pia=B(:,orbit(i,j)+1);
218                 pib=witness(:,i);
219                 pipiab=zeros(n,1);
220                 ii=1;
221
222                 while ii<= n
223                     pipiab(ii)=pia(pib
224                     (ii));
225                     ii=ii+1;
226                 end
227                 witness(:,i)=pipiab;
228                 j=j+1;
229             end
230             i=i+1;
231         end
232         witness;
233
234
235
236         %Ran orbit program
237         t=size(orbit,1);
238         T=witness;
239
240         %Find inverse of orbit
241         %pi=orbit(:,1)
242         %"Run invpi program
243
244
245
246         %i=1;
247
248         %inv=zeros(n,1);
249         %while i<= n
250             % term=0;
251             %j=1;
252             % while term == 0
253
254             % if pi(j)==i;
255                 % inv(i)=j;
256                 % term=1;
257             % else

```

```

258         % j=j+1;
259     % end
260 % end
261 % i=i+1;
262 % end
263
264
265
266 % inverseorbit=inv;
267 %Found inverse orbit
268
269
270
271
272 yi=1;
273
274 clear stab
275 stab=zeros(n,1);
276 stab;
277
278 while yi <= y
279
280     ti=1;
281
282     while ti<= t
283
284
285         %Calculate Y(:,yi)*T(:,ti)
286
287         pia=Y(:,yi);
288         pib=T(:,ti);
289
290         %"Run timespi program"
291
292         i=1;
293
294         pipiab=zeros(n,1);
295
296         while i<= n
297             pipiab(i)=pia(pib(i));
298             i=i+1;
299         end
300
301
302
303         %Calculate varphi(piapib)
304
305         ne=pipiab(a);
306         i=1;
307         foundinverse=0;
308         while foundinverse==0
309             if orbit(i,1)==ne
310                 nu=i;
311                 foundinverse=1;

```

```

312         else
313             i=i+1;
314         end
315     end
316
317
318
319     varphi=T(:,nu);
320
321     %Calculate inverse of varphi
322
323     pi=varphi;
324
325     %"run invpi program "
326
327     i=1;
328
329     inv=zeros(n,1);
330     while i<= n
331         term=0;
332         j=1;
333         while term == 0
334
335             if pi(j)==i
336                 inv(i)=j;
337                 term=1;
338             else
339                 j=j+1;
340             end
341         end
342         i=i+1;
343     end
344
345
346     %Calculate stab
347
348     pia=inv;
349     pib=pi*piab;
350
351     %"Run timespi program"
352
353     i=1;
354
355     pipiab=zeros(n,1);
356
357     while i<= n
358         pipiab(i)=pia(pib(i));
359         i=i+1;
360     end
361     pipiab;
362
363
364     (yi-1)*y+(ti);
365

```

```

366
367         stab(:,(yi-1)*y+ti)=pipiab;
368
369
370
371
372
373         ti=ti+1;
374     end
375     yi=yi+1;
376 end
377
378     stab;
379
380
381
382
383
384         nn(iii)=size(orbit,1);
385
386
387         G;
388         G=stab;
389     G;
390     %remove 0 cols of G
391     i=1;
392     width=size(G,2);
393
394     while i<= width
395         if norm(G(:,i))==0
396             G=G(:,[1:i-1],[i+1:width]);
397             width=width-1;
398         else
399             i=i+1;
400         end
401
402     end
403
404
405
406
407     G;
408     oldGsize(iii+1)=size(G,2);
409
410     %DONT REDUCE GENERATORS OF G yet
411 else
412 end
413     iii=iii+1;
414 end
415 nn;
416 pgrousize=prod(nn)
417
418
419 end

```

5.6 Question 9

```
1 function [outputArg1,outputArg2] = estimateP_n(range)
2 rangeindex=1;
3 maximumsizeofrange=size(range,2);
4 while rangeindex<= maximumsizeofrange
5     n=range(rangeindex);
6     nnnn=1;
7     prob=0;
8     oldGsize=0;
9     newGsize=0;
10    while nnnn<= 10000
11        G=zeros(n,1);
12        %Generate G_(:,1)
13        i=2;
14        G(1,1)=randi(n);
15
16        while i<= n
17            x=randi(n);
18            if ismembertol(x,G([1:i-1],1))
19            else
20                G(i,1)=x;
21                i=i+1;
22            end
23        end
24
25        %Generated G(:,1)
26
27        %Generate G_(:,1)
28        i=2;
29        G(1,2)=randi(n);
30
31        while i<= n
32            x=randi(n);
33            if ismembertol(x,G([1:i-1],2))
34            else
35                G(i,2)=x;
36                i=i+1;
37            end
38        end
39
40        %Generated G(:,1)
41
42
43
44
45        G;
46        nn=1;
47        clear oldGsize newGsize
48
49
50        %run perm group size program
51            oldGsize(1)=size(G,2);
52            iii=1;
```

```

53
54 nn(iii)=1;
55 nontrivorb=0;
56
57
58 while nontrivorb == 0
59
60
61 %Step 1: Reduce generators of G
62     clear Pi
63     Pi=G;
64     %Run stripping algorithm
65     k=size(Pi,2);
66
67     n=size(Pi,1);
68
69     l=1;
70
71     A=zeros(n,n);
72
73     while l <= k
74
75         r=1;
76         finish=0;
77
78         while finish == 0
79
80             if r~=n
81
82                 if Pi(r,l) ~= r
83                     Pi;
84                     if A(r,Pi(r,l)) == 0
85
86                         A(r,Pi(r,l))=1;
87                         finish=1;
88
89                     else
90
91                         pi=Pi(:,A(r,Pi(r,l)));
92                         %"run invert pi"
93                         i=1;
94
95                         inv=zeros(n,1)
96                         ;
97                         while i<= n
98                             term=0;
99                             j=1;
100                             while term
101                                 == 0
102
103                                 if pi(j)==
104                                     i;
105                                     inv(i)
106                                     =j;

```

```

103                                     term
104                                     =1;
105                                     else
106                                     j=j+1;
107                                     end
108                                     end
109                                     i=i+1;
110                                     end
111
112                                     pia=inv;
113                                     pib=Pi(:,l);
114                                     %"run multiply pia,pib
115                                     program"
116                                     i=1;
117
118                                     pipiab=zeros(n,1);
119
120                                     while i<= n
121                                     pipiab(i)=pia(
122                                     pib(i));
123                                     i=i+1;
124                                     end
125                                     Pi(:,l)= pipiab;
126
127                                     %Ran multiply pia,pib
128                                     program
129                                     r=r+1;
130
131                                     end
132
133                                     else
134
135                                     r=r+1;
136
137                                     end
138
139                                     else
140
141                                     finish = 1;
142
143                                     end
144
145                                     end
146                                     l=l+1;
147
148                                     end
149                                     A;
150                                     i=1;
151                                     kk=1;
152                                     Qi=zeros(n,1);
153                                     while i<= size(A,1)

```



```

153         j=1;
154         while j<= size(A,2)
155             if A(i,j) ~= 0
156                 Qi(:, kk)=Pi(:,A(i,j));
157                 kk=kk+1;
158             else
159                 end
160             j=j+1;
161         end
162         i=i+1;
163     end
164     Pi=Qi;
165     G=Pi;
166
167
168     newGsize(iii)=size(G,2);
169     %call reduced generators G too
170
171     if G(1,1)==0
172         nontrivorb=1;
173     end
174
175     if nontrivorb==0
176
177         %Step 2: Find non-trivial orbits of G
178
179         pi=G(:,1);
180         i=1;
181         foundel=0;
182         while foundel==0
183
184             if pi(i)==i
185                 i=i+1;
186             else
187                 foundel=1;
188                 a=i;
189             end
190         end
191
192
193
194
195         %Step 3: Find orbit of a
196
197         %Run stabiliser program
198
199         Y=G;
200         y=size(Y,2);
201
202         A=Y;
203         el=a;
204         %Run orbit algorithm;
205
206         k=size(A,2);

```

207
208
209
210
211
212

213
214
215
216
217

218
219
220

221
222
223

224
225
226

227
228
229

230

231

```
n=size(A,1);
orbit=[e1,0];
newsize=1;
oldsize=0;

while oldsize <
    newsize

    i=oldsize+1;
    oldsize=size(
        orbit,1);
    c=1;

    while i <=
        oldsize

        j=1;
        while j<=
            k
            j;

            if
                ismembertol
                (A(
                    orbit
                    (i
                    ,1)
                    ,j)
                    ,
                    orbit
                    (:,1)
                    )

            else
                orbit
                ;

                holdvec
                =[
                    orbit
                    (
                    i
                    ,:)
                    ,
                    j
                    ];

                orbit
                (:,
                size
                (
                orbit
```

		,2)
		+1)
		=
		zeros
		(
		size
		(
		orbit
		,1)
		,1)
		;
232		orbit
		(
		oldsize
		+
		c
		,:)
		=
		holdvec
		;
233		orbit
		(
		oldsize
		+
		c
		,1)
		=
		A
		(
		orbit
		(
		i
		,1)
		,
		j
)
		;
234		c=
		c
		+1;
235		end
236		j=j+1;
237		end
238		i=i+1;
239		end
240		newsize=size(orbit
		,1);
241		end
242		orbit;
243		

244	
245	
246	%Calculate table
247	of witnesses
248	xx=size(orbit,2);
249	orbsize=size(orbit
250	,1);
251	i=2;
252	witness=[1:n].';
253	B=zeros(size(A,1),
254	size(A,2)+1);
255	B(:,1)=[1:n];
256	B(:,[2:size(A,2)
257	+1])=A;
258	
259	while i<=orbsize
260	
261	j=2;
262	witness(:,i)=B
263	(:,1);
264	while j<= xx
265	%"do
266	witness
267	(:,i)=B
	(:,j+1)
	*
	witness
	(:,i)
	pia=B(:,
	orbit(i
	,j)+1);
	pib=
	witness
	(:,i);
	pipiab=
	zeros(n
	,1);
	ii=1;
	while
	ii
	<=
	n
	pipiab
	(
	ii
)
	=
	pia
	(
	pib
	(
	ii
)

```

268                                     ii
                                     =
                                     ii
                                     +1;

269                                     end
270                                     witness
                                     (:,
                                     i)=
                                     pipiab
                                     ;

271                                     j=j+1;
272                                     end
273                                     i=i+1;
274                                     end
275                                     witness;
276
277
278
279
280
281                                     %Ran orbit program
282                                     t=size(orbit,1);
283                                     T=witness;
284
285                                     %Find inverse of orbit
286                                     %pi=orbit(:,1)
287                                     %"Run invpi program
288
289
290
291                                     %i=1;
292
293                                     %inv=zeros(n,1);
294                                     %while i<= n
295                                     % term=0;
296                                     %j=1;
297                                     % while term == 0
298
299                                     % if pi(j)==i;
300                                     %   inv(i)=j;
301                                     %   term=1;
302                                     % else
303                                     %   j=j+1;
304                                     % end
305                                     % end
306                                     % i=i+1;
307                                     % end
308
309
310

```

```

311         % inverseorbit=inv;
312         %Found inverse orbit
313
314
315
316
317         yi=1;
318
319
320         stab=zeros(n,1);
321
322         while yi <= y
323
324             ti=1;
325
326             while ti<= t
327
328
329                 %Calculate Y(:,yi)*T
330                 (:,ti)
331
332                 pia=Y(:,yi);
333                 pib=T(:,ti);
334
335                 %"Run timespi
336                 program"
337
338                 i=1;
339
340                 pipiab=zeros(n
341                 ,1);
342
343                 while i<= n
344                     pipiab(i)=
345                     pia(pib
346                     (i));
347                     i=i+1;
348                 end
349
350                 %Calculate varphi(
351                 piapib)
352
353                 ne=pipiab(a);
354                 i=1;
355                 foundinverse=0;
356                 while foundinverse
357                     ==0
358                     if orbit(i,1)
359                         ==ne
360                         nu=i;
361                         foundinverse
362                         =1;

```

```

356         else
357             i=i+1;
358         end
359     end
360
361
362
363     varphi=T(:,nu);
364
365     %Calculate inverse of
    varphi
366
367     pi=varphi;
368
369     %"run invpi
    program "
370
371     i=1;
372
373     inv=zeros(n,1);
374     while i<= n
375         term=0;
376         j=1;
377         while term ==
            0
378
379             if pi(j)==i
380                 inv(i)=j;
381                 term=1;
382             else
383                 j=j+1;
384             end
385         end
386         i=i+1;
387     end
388
389
390     %Calculate stab
391
392     pia=inv;
393     pib=pi*piab;
394
395     %"Run timespi
    program"
396
397     i=1;
398
399     pi*piab=
        zeros(n
            ,1);
400
401     while i<=
        n

```

```

402                                     pipiab
                                     (i)
                                     =
                                     pia
                                     (
                                     pib
                                     (i)
                                     );
403                                     i=i+1;
404                                     end
405
406
407
408                                     stab(:,(yi-1)*y+ti
                                     )=pipiab;
409
410
411
412
413                                     ti=ti+1;
414                                     end
415                                     yi=yi+1;
416                                     end
417
418                                     stab;
419
420
421
422                                     %Found orbit of a
423
424
425
426                                     nn(iii)=size(orbit,1);
427
428
429
430                                     G=stab;
431
432                                     %remove 0 cols of G
433                                     i=1;
434                                     width=size(G,2);
435
436                                     while i<= width
437                                         if norm(G(:,i))==0
438                                             G=G(:, [[1:i-1],[i+1:width]]);
439                                             width=width-1;
440                                         else
441                                             i=i+1;
442                                         end
443
444                                     end
445                                     G;
446                                     oldGsize(iii+1)=size(G,2);
447

```



```

448             %DONT REDUCE GENERATORS OF G yet
449             else
450             end
451             iii=iii+1;
452         end
453         nn;
454         pgroupsize=prod(nn);
455         pgroupsize;
456
457
458
459
460
461         %ran perm group size program
462
463         if pgroupsize == factorial(n)
464             prob=prob+1;
465         end
466         nnnn=nnnn+1;
467     end
468     P_n=prob/10000;
469     P_n;
470     probvec(rangeindex)=P_n;
471     rangeindex=rangeindex+1;
472 end
473 probvec
474
475
476
477
478
479
480 end

```