Part II Computational Projects 2020

C3318H

Attach to front of report

Project number:

16.5

16.5

Question 1

Programs to find the inverse of a permutation and product of two permutations can be found in page 7 named `inverse.m` and `product.m` respectively.

Suppose π is a given permutation of $\{1, 2, \cdots, n\}$. For computing it's inverse, we nee

1. n operations for defining $\pi^{-1} = [1\,2 \cdots n]$
2. Then for $1 \leq i \leq n$, defining $\pi^{-1}(\pi(i)) = i$ requires 2 operations (one for each of computing $\pi(i)$ and assigning the value to π^{-1}).

So in total $n + 2 \times n = 3n = O(n)$ operations.

Similarly for computing product of two permutations π_1 and π_2, we just need to define $(\pi_1\pi_2)(i) = \pi_2(\pi_1(i))$ for $1 \leq i \leq n$. This will require $n + 3n = 4n = O(n)$ operations. (n operating for initial assignment of $\pi_1\pi_2$ and 3 operations for computing $\pi_2(\pi_1(i))$ and assigning it to $(\pi_1\pi_2)(i)$ for each i)

Question 2

Suppose $\pi_1, \pi_2, \cdots \pi_k$ are the given permutations generating G. Suppose at a given stage we have $\sigma_1, \cdots \sigma_r$ on the $n \times n$ array and $\pi_l, \cdots \pi_k$ left from the initial generators and in the next move we change π_l to $\sigma_{r+1} = \prod_{i \in I} \sigma_i^{-1} \pi_l$ for some $I \subseteq \{1, \cdots, r\}$. Now we have

$$\sigma_{r+1} = (\prod_{i \in I} \sigma_i)^{-1} \pi_l \in \langle \sigma_1, \cdots \sigma_r, \pi_l, \cdots \pi_k \rangle$$
$$\Rightarrow \langle \sigma_1, \cdots \sigma_r, \sigma_{r+1}, \pi_{l+1}, \cdots \pi_k \rangle \leq \langle \sigma_1, \cdots \sigma_r, \pi_l, \cdots \pi_k \rangle$$

Again

$$\pi_l = (\prod_{i \in I} \sigma_i) \sigma_{r+1} \in \langle \sigma_1, \cdots \sigma_r, \sigma_{r+1}, \pi_{l+1}, \cdots \pi_k \rangle$$
$$\Rightarrow \langle \sigma_1, \cdots \sigma_r, \pi_l, \cdots \pi_k \rangle \leq \langle \sigma_1, \cdots \sigma_r, \sigma_{r+1}, \pi_{l+1}, \cdots \pi_k \rangle$$

So $\langle \sigma_1, \cdots \sigma_r, \pi_l, \cdots \pi_k \rangle = \langle \sigma_1, \cdots \sigma_r, \sigma_{r+1}, \pi_{l+1}, \cdots \pi_k \rangle$. Applying induction on the number of steps, we can deduce that the permutations on the array generate $G = \langle \pi_1, \cdots, \pi_n \rangle$.

Every block in the $n \times n$ array has at most 1 permutation. If a permutation π is put in block (u, v), that means π fixes $1, \cdots, u - 1$ and sends u to v where $u \neq v$. So we must have $v \geq u$. Hence all the blocks (x, y) with $x \geq y$ are empty. Hence there are at most $\frac{n(n-1)}{2}$ permutations in the array which is an upper bound for the number of modified set of generators.

Suppose there are k initial generators. We know that it requires $3n$ operations for computing inverse and $4n$ for product of two permutations. Now for a given generator, we have to do at most $n-1$ inverses and $n-1$ multiplications to put it in the array, which requires at most $3n(n-1)+4n(n-1) = 7n(n-1)$ operations. Also we need to store at most k permutations, hence number of operations needed is at most $7n(n-1)k + nk = kn(7n-6) = O(kn)$.

Question 3

A program to compute the reduced generating set of a group can be found in page 7 named `reducedgeneratingset.m`.

We write a program `alternative_group` (page 8) to produce the alternative group A_n. We will be using it to try our programs on alternative groups.

We write the following command to run it on S_5 and A_5.

```
>> S5 = perms([1 2 3 4 5 ]) ;
>> reducedgeneratingset(S5)

ans =

5     4     3     2     1
1     2     3     5     4
1     2     4     3     5
1     2     5     3     4
1     3     2     4     5
1     4     2     3     5
1     5     2     3     4
4     5     3     2     1
3     5     4     2     1
2     5     4     3     1
>> A5 = alternative_group(5) ;
>> reducedgeneratingset(A5)

ans =

5     4     3     2     1
1     2     4     5     3
1     2     5     3     4
1     3     2     5     4
1     4     2     3     5
1     5     2     4     3
4     5     3     1     2
3     5     4     2     1
2     5     4     1     3
```

In the first one we get the generators $(2\,3), (3\,4), (4\,5), (5\,1)$. Using the identity $(a\,c) = (a\,b)(b\,c)(a\,b)$, we can show all the transpositions can be achieved, which obviously generates the whole of S_5.

For the second one, easy to see that all the permutations are even. We have $(5\,3\,4)$ and $(5\,3\,4\,1\,2)$ which generate A_5.

Question 4

A program to compute the array of a permutation group can be found in page 3.

Let $\phi : G \to A$ be the map $g \mapsto g\alpha$. Obviously ϕ is surjective. Also

$$\begin{aligned} \phi(g) &= \phi(h) \\ \Leftrightarrow g\alpha &= h\alpha \\ \Leftrightarrow h^{-1}g\alpha &= \alpha \\ \Leftrightarrow h^{-1}g &\in G_\alpha \\ \Leftrightarrow hG_\alpha &= gG_\alpha \end{aligned}$$

Hence ϕ induces a bijective map

$$\begin{aligned} \phi_\alpha : G/G_\alpha &\to A \\ gG_\alpha &\mapsto g\alpha \end{aligned}$$

Orbit-Stabilizer theorem Let G be a group which acts on a finite set X. Suppose $\alpha \in X$ is given. Let A and G_α denote the orbit and stabilizer of α in G. Then we have

$$|A| = [G : G_\alpha] = \frac{|G|}{|G_\alpha|}$$

where $[G : G_\alpha]$ denotes the index of G_α in G.

Question 5

A program to compute the orbit with witnesses of a given element can be found in page 5.

The main idea of the program is to apply the generating permutations on the orbits found so far. If we get any new element, we apply the generators on those new elements and repeat. Since G is finite, at some point we won't find any new element; which completes the orbit.

We can assume $X = \{1, 2, \cdots, n\}$. Suppose A is the table s.t the rows represent the given set of permutations. Also let a given from X.

We use reducedgenset to get a modified set of generators A. We initially set the set of orbits to be $\{a\}$. Suppose s denote the number of total orbits found before the last step and k denote the number of orbits found in the last step. They have initial values 0 and 1.

Now at each step, we apply the permutations in A on the set of newly found k orbits and add those in orbits. Set $s = s + k$ and $k = \mathit{length}(\mathit{orbit}) - s$ and move to the next step. If at some point we have $k = 0$, that means we didn't find any new element for orbits.

For example, we apply it on $A = \{(1\,2), (3\,4\,5)\}$ which generates a group G with orbits $\{1, 2\}$ and $\{3, 4, 5\}$.

```
>> A = [ 2 1 3 4 5 ; 1 2 4 5 3 ];
>> [orbit,witness] = orbit_witness(3,A) ;
>> orbit

orbit =

3     4     5

>> witness

witness =

1     2     3     4     5
1     2     4     5     3
1     2     5     3     4
```

Easy to see that the 3 witnesses are $id, (3\,4\,5\,)$ and $(3\,4\,5\,)^2$ which lie in G and send 3 to $3, 4, 5$ respectively.

Question 6

We have

$$\begin{aligned} t_{i+1}\alpha &= \phi(y_i t_i)\alpha \\ &= y_i t_i \alpha \end{aligned}$$

for $i = 1, \cdots r$. Hence

$$\begin{aligned} t_{r+1}\alpha &= y_r t_r \alpha \\ &= y_r y_{r-1} t_{r-1} \alpha \\ &\vdots \\ &= y_r \cdots y_1 t_1 \alpha \\ &= x t_1 \alpha \\ &= \alpha \\ \therefore t_{r+1} &\in G_\alpha \\ \therefore t_{r+1} &= t_1 \end{aligned}$$

Suppose x is a given element of G_α, then we have $t_1 x t_1^{-1} \in G_\alpha$. Let $t_1 x t_1^{-1} = y_r \cdots y_1$ where $y_i \in Y$. Since $t_{i+1} = \phi(y_i t_i)$, we have $\phi(y_i t_i)^{-1}.y_i.t_i = t_{i+1}.y_i.t_i$. Multiplying for $i = 1.\cdots r$, we get

$$\begin{aligned} \prod_{i=1}^{r} \phi(y_i t_i)^{-1}.y_i.t_i &= t_{r+1}^{-1} y_r \cdots y_1 t_1 \\ &= t_1^{-1} t_1 x t_1^{-1} t_1 \\ &= x \end{aligned}$$

We can deduce $G_\alpha \leq H$.
Also $\phi(yt)^{-1} yt\alpha = \phi(yt)^{-1}\phi(yt)\alpha = \alpha$, hence $\phi(yt)^{-1}yt \in G_\alpha$. So G_α is generated by H.

Question 7

A program to compute the generating set of a stabilizer of a permutation group can be found in page 6.

We run it on S_4 .

```
>> S4 = perms([ 1 2 3 4 ]) ;
>> stabilizer_gen_set ( 4 , S4 )

ans =

3     2     1     4
2     1     3     4
1     3     2     4
```

We see that the derived generating set is $(1\,3), (1\,2), (2\,3)$, each of which fixes 4. Also these are all the transposition of $S_3 < S_4$, hence generate $Stab_G(4)$.

Question 8

A program to generate the order of a permutation group can be found in page 9 named `group_order.m`. We run it on the generating set $B = \{(1\,2\cdots n), (12)\}$, which generates the whole group S_6 from Part IA. We also run it on A_6.

```
>> B = [ 2 3 4 5 6 1 ; 2 1 3 4 5  6 ];
>> group_order(B)
1, 1, 2
3, 3, 6
6, 6, 24
5, 5, 120
2, 2, 720

ans =

        720

>> A6 = alternative_group(6) ;
>> group_order(A6)
2, 2, 3
5, 5, 12
9, 9, 60
360, 14, 360

ans =

360
```

It gives us $720 = 6!$ and $360 = 6!/2$ respectively, as desired.

Suppose the groups we get are $G = G_0 \geq G_1 \geq \cdots \geq G_r = \{id\}$ where $G_{i+1} = Stab_{G_i}(\alpha_i)$ for some $\alpha_i \in G_i$. If we forget to apply the stripping algorithm at every stage, then we would have a generating set of $|Y_i||T_i|$ elements of G_{i+1} where Y_i and T_i are defined for G_i as in Q6. We have $|T_i| = |G_i|/|G_\alpha|$ and $|Y_i|$ =number of given generators of G_{i-1}. Hence

$$|Y_{i+1}| = \frac{|G_i|}{G_{i+1}||Y_i|}$$

So $|Y_r| = \dfrac{|G_0|}{|G_r|}|Y_0| = |G||Y_0|$ where Y_0 is the number of generators given initially.

So if we are given 2 elements of S_{20} which generates the whole S_{20}, then in the last step we will have $2 \times |S_{20}| \approx 4.8 \times 10^{18}$ generatos and we will run out of memory.

Question 9

We know from IA groups that $(1\,2 \cdots n)$ and $(1\,2)$ generate the whole S_n for all n. So we can take $g = (1\,2 \cdots n), h = (1\,2)$. Then $\langle g, h \rangle = S_n$, hence $P_n > 0$.

If we have $g, h \in A_n$, then $\langle g, h \rangle \leq A_n$. Since $|S_n : A_n| = 2$, we have

$$1 - P_n = \mathbb{P}(\langle g, h \rangle \neq S_n) \geq \mathbb{P}((g, h) \in A_n \times A_n) = 1/2 \times 1/2 = 1/4$$
$$\therefore P_n \leq 3/4$$

So we can take $k = 3/4$.

n	P_n
3	0.520000
4	0.410000
5	0.460000
6	0.480000
7	0.600000
8	0.600000
9	0.590000
10	0.750000
11	0.660000
12	0.680000
13	0.700000
14	0.730000

Table 1: values of P_n for $1 \leq n \leq 14$

We run the script `q9.m` in page 9 to generate the table 1 which shows the values of P_n for $1 \leq n \leq 14$. We can see that the values are at most 3/4 which is consistent with what we just proved. Although the sequence is not increasing, the values are somewhat getting closer to 3/4.

Listing 1: `inverse.m`

```
function u = inverse(pi)

n = length(pi);
u = 1:n;

if ~isequal( 1:n , sort(pi) )
    error( 'Input must be a permutation')
else
    for i=1:n
        u(pi(i)) = i;
    end
end
end
```

Listing 2: `product.m`

```
function u = product( pi_1,pi_2)

n = length(pi_1);

if length(pi_2) ~= n
    error('Permutations have different sizes')

elseif ~isequal( 1:n , sort(pi_1) )
    error( 'Input 1 must be a permutation')

elseif ~isequal( 1:n , sort(pi_2) )
    error( 'Input 2 must be a permutation')

else
    u = 1:n ;
    for i=1:n
        u(i) = pi_1(pi_2(i));
    end
end
end
```

Listing 3: `reducedgeneratingset.m`

```
function B = reducedgeneratingset(A)

[m,n] = size(A);

C = zeros(n,n);

B = zeros(0,n);

for i=1:m
    j = 1;
    while j < n

        if A(i,j) == j
            j = j+1;

        elseif C(j,A(i,j)) == 0
            C(j,A(i,j)) = i;
            B = [B;A(i,:)];
            break

        else
            k = C(j,A(i,j));
            A(i,:) = product(inverse(A(k,:)),A(i,:));
            j = j+1;
        end
    end
end
```

Listing 4: `alternative_group.m`

```
function An = alternative_group(n)

Sn = perms(1:n) ;
I = eye(n) ;
An = zeros(0,n) ;
for i=1:factorial(n)
    if det(I(:,Sn(i,:))) == 1
        An = [ An ; Sn(i,:) ] ;
    end
end
```

Listing 5: `orbit_witness.m`

```
function [orbit,witness] = orbit_witness(a,A)

A = reducedgeneratingset(A) ;
[m,n] = size(A);

orbit = a ;
witness = 1:n ;

s = 0;
k = 1;
while k > 0
    for j = 1:k
        for i=1:m
            if ismember( A(i,orbit(s+j)) , orbit ) == 0
                orbit = [ orbit , A(i,orbit(s+j)) ] ;
                witness = [ witness ; product( A(i,:), witness(s+j,:)) ] ;
            end
        end
    end
    s = s + k ;
    k = length(orbit) - s ;
end
```

Listing 6: `stabilizer_gen_set.m`

```
function [B,p] = stabilizer_gen_set ( a , Y )

[orbit,T] = orbit_witness(a,Y);

[m,n] = size(Y);
[p,~] = size(T);
B = zeros(m*p,n);

for i = 1:m
    for j = 1:p

```

```
        yt = product(Y(i,:),T(j,:));

        k = find(orbit == yt(a));
        B((i-1)*p+j,:) = product( inverse(T(k,:)) , yt )  ;
    end
end

B = reducedgeneratingset(B);

end
```

Listing 7: group_order.m

```
function [N,X] = group_order ( A )

[m0,n] = size(A);
X = zeros(0,3);
A = reducedgeneratingset(A);
[m,~] = size(A);

if  m == 0
    N = 1;
else
    a = 1;
    while a < n
        if A(:,a) == a*ones(1,n)
            a = a + 1;
        else
            break
        end
    end

    [B,p] = stabilizer_gen_set ( a , A );
    N = p * group_order ( B );
    %fprintf('%d, %d, %d \n', m0 ,m , N )

end
end
```

Listing 8: q9.m

```
fileID1 = fopen('q9.csv','w');
fprintf(fileID1,'n,p \n') ;

for n = [ 3:14]
    A = zeros(200,n);
    n_fact = factorial(n) ;
    for i = 1:200
        A(i,:) = randperm(n) ;
    end
    order = zeros(1,100) ;
    p = 0;
    for i = 1:100
        if group_order( A(2*i-1:2*i , :)) == n_fact
            p = p + 1 ;
        end
    end
    fprintf(fileID1,' %d, %f \n',n,p/100) ;
end
```