Group

March 28, 2019

Contents

1 Semigroups, Monoids And Groups **1**
- 1.1 Exercise 7 (P29) . . . 1
- 1.2 Exercise 11 (P30) . . . 2
- 1.3 Exercise 15 (P30) . . . 2

2 Homomorphism And Subgroups **3**
- 2.1 Exercise 1 (P33) . . . 3
- 2.2 Exercise 2 (P33) . . . 3
- 2.3 Exercise 3 (P33) . . . 3
  - 2.3.1 non-abelian . . . 3
  - 2.3.2 it is a group . . . 3
  - 2.3.3 why order 8 . . . 4
- 2.4 Exercise 4 (P33) . . . 9
- 2.5 Exercise 5 (P33) . . . 11
- 2.6 Exercise 6 (P34) . . . 12
- 2.7 Exercise 7 (P34) . . . 12

1 Semigroups, Monoids And Groups

1.1 Exercise 7 (P29)

it is easy to see that $(\forall \bar{a}, \bar{b} \in Z_p - \bar{0}) \rightarrow \bar{a}\bar{b} \in Z_p - \bar{0}$. And $(\forall \bar{a} \neq 0) \rightarrow (a, p) = 1$. so $(\exists x, y \in Z) \wedge xa + yp = 1$, i.e. $xa = y^{'} p + 1$. which means that $a^{-1} = x$ exists.

1.2 Exercise 11 (P30)

if $(ab)^n = a^n b^n$ for three consequtive integers n. then, $(ab)^{n-1} = a^{n-1}b^{n-1}$, $(ab)^n = a^n b^n$, $(ab)^{n+1} = a^{n+1}b^{n+1}$. split $(ab)^{n+1}$, we get:

$$
\begin{aligned}
(ab)^{n+1} &= a^{n+1}b^{n+1} && (1)\\
=&(ab)(ab)^n = (ab)a^n b^b && (2)\\
=&(ab)^n(ab) = a^n b^n (ab) && (3)\\
=&(ab)(ab)^{n-1}(ab) = (ab)a^{n-1}b^{n-1}(ab) && (4)\\
=&(ab)^2(ab)^{n-1} = (ab)^2 a^{n-1}b^{n-1} && (5)\\
=&(ab)^{n-1}(ab)^2 = a^{n-1}b^{n-1}(ab)^2 && (6)
\end{aligned}
$$

from (1) and (3) we could get:

$$
\begin{aligned}
a^n b^n (ab) =& a^{n+1}b^{n+1} && (7)\\
b^n ab =& ab^{n+1} && (8)\\
b^n a =& ab^n && (9)
\end{aligned}
$$

from (1) and (2) we could get:

$$
\begin{aligned}
(ab)a^n b^n =& a^{n+1}b^{n+1} && (10)\\
ba^n =& a^n b && (11)
\end{aligned}
$$

from (1) and (4) we could get:

$$
\begin{aligned}
(ab)(ab)^{n-1}(ab) =& a^{n+1}b^{n+1} && (12)\\
(ab)a^{n-1}b^{n-1}(ab) =& a^{n+1}b^{n+1} && (13)\\
ba^{n-1}b^{n-1}a =& a^n b^n && (14)\\
ba^n a^{-1}b^{-1}b^n a =& a^n b^b && (15)
\end{aligned}
$$

from (9), (11) and (15) we could get:

$$
\begin{aligned}
a^n ba^{-1}b^{-1}ab^n =& a^n b^n && (16)\\
ba^{-1}b^{-1}a =& e && (17)\\
a^{-1}b^{-1} =& b^{-1}a^{-1} && (18)
\end{aligned}
$$

1.3 Exercise 15 (P30)

for all $a \in G$, we could define $G' = \{ab | \forall b \in G\}$. because $(\forall a, b, c \in G)ab = ac \rightarrow b = c$, we could define a injection function $f : G \rightarrow G'$ by leting

$f(x) = ax$. It is easy to verify that Img $f = G'$. so f is a bijection and $G = G'$. we could look the $f(x)$ as an edge from x to $f(x)$. then we could get a graph, such that all the verteces in the graph are in a cycle. so we could get a circle like this: $aa = f, af = c, ac = \cdots ax = a$. and we could get $a^k = a$ from this circle. so $(\forall b \in G)a^k b = ab \to a^{k-1}b = b$. so we could let $a^{k-1} = e$ to be the left unit of G. and by knowing that $(\forall x \in G)(\exists y \in G) \wedge xy = e$. we've proved that all the elements in G has a inverse.

2 Homomorphism And Subgroups

2.1 Exercise 1 (P33)

construct a monoid G as follow:

	a	b	e
a	a	a	a
b	b	b	b
e	a	b	e

it is easy to find that G is a monoid. And we construct a homomorphism $f : G \to G$ as follow: $f(a) = a, f(b) = a, f(e) = a$.

2.2 Exercise 2 (P33)

$f(x)f(y) = x^{-1}y^{-1} = (yx)^{-1} \neq (xy)^{-1} = f(xy)$

2.3 Exercise 3 (P33)

2.3.1 non-abelian

$$AB = \begin{pmatrix} i & 0 \\ 0 & -i \end{pmatrix}$$

$BA = \begin{pmatrix} -i & 0 \\ 0 & i \end{pmatrix}$ so, Q_8 is not abelian.

2.3.2 it is a group

And we know that $A^4 = B^4 = I$. Because jBB$(\forall i, j \in Z)A^i B^j B^{4-j} A^{4-i} = I \Rightarrow (A^i B^j)^{-1} = B^{4-j} A^{4-i}$, we claim that all the elements in this set have a inverse.

2.3.3 why order 8

you could write a program to check this.

```
template <class Type>
struct Matrix {
#define FOR(i, l, r) for (int (i) = (l); (i) <= (r); (i)++)
    int row, col;
    complex<Type> w[maxn][maxn];
    Matrix (int row, int col):row(row), col(col) {
        FOR(i, 1, row) FOR(j, 1, col) w[i][j] = 0; // initalize
    }
    Matrix (const Matrix &other) {
        row = other.row, col = other.col;
        FOR(i, 1, row) FOR(j, 1, col) w[i][j] = other.w[i][j];
    }
    friend Matrix operator * (const Matrix &a, const Matrix &b) {
        assert(a.col == b.row);
        Matrix c(a.row, b.col);
        FOR(i, 1, a.row) FOR(j, 1, b.col) {
            c.w[i][j] = 0;
            FOR(k, 1, a.col) c.w[i][j] += a.w[i][k] * b.w[k][j];
        }
        return c;
    }
    friend bool operator < (const Matrix &a, const Matrix &b) {
        assert(a.row == b.row);
        assert(a.col == b.col);
        FOR(i, 1, a.row) FOR(j, 1, a.col) {
            if (a.w[i][j] != b.w[i][j]) {
                if (a.w[i][j].real() == b.w[i][j].real())
                    return a.w[i][j].imag() < b.w[i][j].imag();
                return a.w[i][j].real() < b.w[i][j].real();
            }
        }
        return false;
    }
    friend bool operator == (const Matrix &a, const Matrix &b) {
        if (a.row != b.row || a.col != b.col) return false;
        FOR(i, 1, a.row) FOR(j, 1, a.col) {
            if (a.w[i][j] != b.w[i][j]) return false;
```

```
        }
        return true;
    }
    void Log() {
        FOR(i, 1, row) {
            FOR(j, 1, col) {
                printf("%d + %di ", w[i][j].real(), w[i][j].imag());
            } printf("\n");
        }
    }
#undef FOR
};

void search() {
    initI(); initA(); initB();
    set<Matrix<int> > st; st.clear();
    queue<Matrix<int> > q;
    q.push(I);
    st.insert(I);
    while(!q.empty()) {
        Matrix<int> tt = q.front(); q.pop();
        tt.Log(); puts("");
        Matrix<int> ta = tt * A;
        Matrix<int> tb = tt * B;
        Matrix<int> at = A * tt;
        Matrix<int> bt = B * tt;
        if (!st.count(ta)) {
            st.insert(ta);
            q.push(ta);
        }
        if (!st.count(tb)) {
            st.insert(tb);
            q.push(tb);
        }
        if (!st.count(at)) {
            st.insert(at);
            q.push(at);
        }
        if (!st.count(bt)) {
            st.insert(bt);
```

```
            q.push(bt);
        }
    }
    cout << st.size() << endl;
}

#include <iostream>
#include <cstring>
#include <cstdio>
#include <complex>
#include <map>
#include <set>
#include <queue>
using namespace std;

const int maxn = 10;

// load the Matrix Class
template <class Type>
struct Matrix {
#define FOR(i, l, r) for (int (i) = (l); (i) <= (r); (i)++)
    int row, col;
    complex<Type> w[maxn][maxn];
    Matrix (int row, int col):row(row), col(col) {
        FOR(i, 1, row) FOR(j, 1, col) w[i][j] = 0; // initalize
    }
    Matrix (const Matrix &other) {
        row = other.row, col = other.col;
        FOR(i, 1, row) FOR(j, 1, col) w[i][j] = other.w[i][j];
    }
    friend Matrix operator * (const Matrix &a, const Matrix &b) {
        assert(a.col == b.row);
        Matrix c(a.row, b.col);
        FOR(i, 1, a.row) FOR(j, 1, b.col) {
            c.w[i][j] = 0;
            FOR(k, 1, a.col) c.w[i][j] += a.w[i][k] * b.w[k][j];
        }
        return c;
    }
    friend bool operator < (const Matrix &a, const Matrix &b) {
```

```
        assert(a.row == b.row);
        assert(a.col == b.col);
        FOR(i, 1, a.row) FOR(j, 1, a.col) {
            if (a.w[i][j] != b.w[i][j]) {
                if (a.w[i][j].real() == b.w[i][j].real())
                    return a.w[i][j].imag() < b.w[i][j].imag();
                return a.w[i][j].real() < b.w[i][j].real();
            }
        }
        return false;
    }
    friend bool operator == (const Matrix &a, const Matrix &b) {
        if (a.row != b.row || a.col != b.col) return false;
        FOR(i, 1, a.row) FOR(j, 1, a.col) {
            if (a.w[i][j] != b.w[i][j]) return false;
        }
        return true;
    }
    void Log() {
        FOR(i, 1, row) {
            FOR(j, 1, col) {
                printf("%d + %di ", w[i][j].real(), w[i][j].imag());
            } printf("\n");
        }
    }
#undef FOR
};

Matrix<int> I(2, 2), A(2, 2), B(2, 2);
void initI() {
    I.w[1][1] = 1;
    I.w[2][2] = 1;
}
void initA() {
    A.w[1][2] = 1;
    A.w[2][1] = -1;
}
void initB() {
    B.w[1][2] = complex<int>(0, 1);
    B.w[2][1] = complex<int>(0, 1);
```

```
}

void search() {
    initI(); initA(); initB();
    set<Matrix<int> > st; st.clear();
    queue<Matrix<int> > q;
    q.push(I);
    st.insert(I);
    while(!q.empty()) {
        Matrix<int> tt = q.front(); q.pop();
        tt.Log(); puts("");
        Matrix<int> ta = tt * A;
        Matrix<int> tb = tt * B;
        Matrix<int> at = A * tt;
        Matrix<int> bt = B * tt;
        if (!st.count(ta)) {
            st.insert(ta);
            q.push(ta);
        }
        if (!st.count(tb)) {
            st.insert(tb);
            q.push(tb);
        }
        if (!st.count(at)) {
            st.insert(at);
            q.push(at);
        }
        if (!st.count(bt)) {
            st.insert(bt);
            q.push(bt);
        }
    }
    cout << st.size() << endl;
}

int main() {
    search();
    return 0;
}
```

2.4 Exercise 4 (P33)

```
#include <iostream>
#include <cstring>
#include <cstdio>
#include <complex>
#include <map>
#include <set>
#include <queue>
using namespace std;

const int maxn = 10;

// load the Matrix Class
template <class Type>
struct Matrix {
#define FOR(i, l, r) for (int (i) = (l); (i) <= (r); (i)++)
    int row, col;
    complex<Type> w[maxn][maxn];
    Matrix (int row, int col):row(row), col(col) {
        FOR(i, 1, row) FOR(j, 1, col) w[i][j] = 0; // initalize
    }
    Matrix (const Matrix &other) {
        row = other.row, col = other.col;
        FOR(i, 1, row) FOR(j, 1, col) w[i][j] = other.w[i][j];
    }
    friend Matrix operator * (const Matrix &a, const Matrix &b) {
        assert(a.col == b.row);
        Matrix c(a.row, b.col);
        FOR(i, 1, a.row) FOR(j, 1, b.col) {
            c.w[i][j] = 0;
            FOR(k, 1, a.col) c.w[i][j] += a.w[i][k] * b.w[k][j];
        }
        return c;
    }
    friend bool operator < (const Matrix &a, const Matrix &b) {
        assert(a.row == b.row);
        assert(a.col == b.col);
        FOR(i, 1, a.row) FOR(j, 1, a.col) {
            if (a.w[i][j] != b.w[i][j]) {
```

```
                if (a.w[i][j].real() == b.w[i][j].real())
                    return a.w[i][j].imag() < b.w[i][j].imag();
                return a.w[i][j].real() < b.w[i][j].real();
            }
        }
        return false;
    }
    friend bool operator == (const Matrix &a, const Matrix &b) {
        if (a.row != b.row || a.col != b.col) return false;
        FOR(i, 1, a.row) FOR(j, 1, a.col) {
            if (a.w[i][j] != b.w[i][j]) return false;
        }
        return true;
    }
    void Log() {
        FOR(i, 1, row) {
            FOR(j, 1, col) {
                printf("%d + %di ", w[i][j].real(), w[i][j].imag());
            } printf("\n");
        }
    }
#undef FOR
};

Matrix<int> I(2, 2), A(2, 2), B(2, 2);
void initI() {
    I.w[1][1] = 1;
    I.w[2][2] = 1;
}
void initA() {
    A.w[1][2] = 1;
    A.w[2][1] = -1;
}
void initB() {
    B.w[1][2] = complex<int>(1, 0);
    B.w[2][1] = complex<int>(1, 0);
}

void search() {
    initI(); initA(); initB();
```

```
    set<Matrix<int> > st; st.clear();
    queue<Matrix<int> > q;
    q.push(I);
    st.insert(I);
    while(!q.empty()) {
        Matrix<int> tt = q.front(); q.pop();
        tt.Log(); puts("");
        Matrix<int> ta = tt * A;
        Matrix<int> tb = tt * B;
        Matrix<int> at = A * tt;
        Matrix<int> bt = B * tt;
        if (!st.count(ta)) {
            st.insert(ta);
            q.push(ta);
        }
        if (!st.count(tb)) {
            st.insert(tb);
            q.push(tb);
        }
        if (!st.count(at)) {
            st.insert(at);
            q.push(at);
        }
        if (!st.count(bt)) {
            st.insert(bt);
            q.push(bt);
        }
    }
    cout << st.size() << endl;
}

int main() {
    search();
    return 0;
}
```

2.5 Exercise 5 (P33)

Reflexivity: $aa^{-1} \in S \Leftrightarrow e \in S$. Symmetry: $ab^{-1} \in S \rightarrow ba^{-1} \in S \Leftrightarrow$ every elements in S has an inverse. Transitivity: $ab^{-1} \in S \wedge bc^{-1} \in S \rightarrow ac^{-1} \in$

$S \Leftrightarrow$ operation on S is close.

2.6 Exercise 6 (P34)

name the subset as S. $\forall a \in S$, let $S_a = \{a^n | n \in Z\} \subset S$. then we could find that $(\exists n, m \in Z) a^n = a^m$ (because of finite). we could use this infomation to construct e and the inverse.

2.7 Exercise 7 (P34)

let $S = \{kn | k \in Z\}$. construct a homomorphism $f : S \to Z$ by letting $f(x) = x/n$. $f(xnyn) = \frac{xn}{n}\frac{yn}{n} = f(xn)f(yn)$.

1. f is a surjection.
2. $f(x) = f(y) \Rightarrow x/n = y/n \Rightarrow x = y$, so f is a injectoin.

so f is a bijection. so $S \cong Z$.