1 Mathematics

1.1 Extended Euclidean

```cpp
// ax + by = gcd(a,b)
pair<long long, long long> extgcd(long long a, long long b)
{
    if (b == 0)
        return {1, 0};
    long long k = a / b;
    pair<long long, long long> p = extgcd(b, a - k * b);
    //cout << p.first << " " << p.second << endl;
    //cout << "商數(k)=  " << k << endl << endl;
    return {p.second, p.first - k * p.second};
}

int main()
{
    int a, b;
    cin >> a >> b;
    pair<long long, long long> xy = extgcd(a, b); //(x0,y0)
    cout << xy.first << " " << xy.second << endl;
    cout << xy.first << " * " << a << " + " << xy.second << " * " << b << endl;
    return 0;
}
// ax + by = gcd(a,b) * r
/*find |x|+|y| -> min*/
int main()
{
    long long r, p, q; /*px+qy = r*/
    int cases;
    cin >> cases;
    while (cases--)
    {
        cin >> r >> p >> q;
        pair<long long, long long> xy = extgcd(q, p); //(x0,y0)
        long long ans = 0, tmp = 0;
        double k, k1;
        long long s, s1;
        k = 1 - (double)(r * xy.first) / p;
        s = round(k);
        ans = llabs(r * xy.first + s * p) + llabs(r * xy.second - s * q);
        k1 = -(double)(r * xy.first) / p;
        s1 = round(k1);
        /*cout << k << endl << k1 << endl;
        cout << s << endl << s1 << endl;
        */
        tmp = llabs(r * xy.first + s1 * p) + llabs(r * xy.second - s1 * q);
        ans = min(ans, tmp);

        cout << ans << endl;
    }
    return 0;
}
```

1.2 Hex to Dec

```cpp
int HextoDec(string num) //16 to 10
{
    int base = 1;
    int temp = 0;
    for (int i = num.length() - 1; i = 0; i--)
    {
        if (num[i] = '0' && num[i] = '9')
        {
            temp += (num[i] - 48) base;
            base = base 16;
        }
        else if (num[i] = 'A' && num[i] = 'F')
        {
            temp += (num[i] - 55) base;
            base = base 16;
        }
    }
    return temp;
}
void DecToHex(int p_intValue) //10 to 16
{
    char l_pCharRes = new (char);
    sprintf(l_pCharRes, % X, p_intValue);
    int l_intResult = stoi(l_pCharRes);
    cout l_pCharRes n;
    return l_intResult;
}
```

1.3 PI

```cpp
#define PI acos(-1)
#define PI M_PI
```

1.4 Prime table

```cpp
// 埃拉托斯特尼篩法
const int maxn = 10000000;
bitset<maxn> prime;
void sieve()
{
    for (int i = 2; i * i < maxn; ++i)
    {
        if (prime[i] == 0)
        {
            for (int j = i * i; j < maxn; j += i)
                prime[j] = 1;
        }
    }
}
/* 0跟1要寫if過濾掉 */
// if(!prime[數字])
//      我是質數
```

1.5 二分逼近法

```cpp
#define eps 1e-14
void half_interval()
{
    double L = 0, R = /*區間*/, M;
    while (R - L >= eps)
    {
        M = (R + L) / 2;
        if (/*函數*/ > /*方程式目標*/)
            L = M;
        else
            R = M;
    }
    printf("%.3lf\n", R);
}
```

1.6 四則運算

```cpp
string s = "";
long long int DFS(int le, int ri) // (0, string final index)
{
    int c = 0;
    for (int i = ri; i >= le; i--)
    {
        if (s[i] == ')')
            c++;
        if (s[i] == '(')
            c--;
        if (s[i] == '+' && c == 0)
            return DFS(le, i - 1) + DFS(i + 1, ri);
        if (s[i] == '-' && c == 0)
            return DFS(le, i - 1) - DFS(i + 1, ri);
    }
    for (int i = ri; i >= le; i--)
    {
        if (s[i] == ')')
            c++;
        if (s[i] == '(')
            c--;
        if (s[i] == '*' && c == 0)
            return DFS(le, i - 1) * DFS(i + 1, ri);
        if (s[i] == '/' && c == 0)
            return DFS(le, i - 1) / DFS(i + 1, ri);
        if (s[i] == '%' && c == 0)
            return DFS(le, i - 1) % DFS(i + 1, ri);
    }
    if ((s[le] == '(') && (s[ri] == ')'))
        return DFS(le + 1, ri - 1); //去除刮號
    if (s[le] == ' ' && s[ri] == ' ')
        return DFS(le + 1, ri - 1); //去除左右兩邊空格
    if (s[le] == ' ')
        return DFS(le + 1, ri); //去除左邊空格
    if (s[ri] == ' ')
        return DFS(le, ri - 1); //去除右邊空格
    long long int num = 0;
    for (int i = le; i <= ri; i++)
        num = num * 10 + s[i] - '0';
    return num;
}
```

1.7 數字乘法組合

```cpp
void toans(vector<vector<int>> &ans, vector<int> com)
{
    // sort(com.begin(), com.end());
    ans.push_back(com);
    // for (auto i : com)
    //     cout << i << ' ';
    // cout << endl;
}
void finds(int j, int old, int num, vector<int> com, vector<vector<int>> &ans)
{
    for (int i = j; i <= sqrt(num); i++)
    {
        if (old == num)
            com.clear();
        if (num % i == 0)
        {
            vector<int> a;
            a = com;
            a.push_back(i);
            finds(i, old, num / i, a, ans);
            a.push_back(num / i);
            toans(ans, a);
        }
    }
}
int main()
{
    vector<vector<int>> ans;
    vector<int> zero;
    finds(2, num, num, zero, ans);
    // num 為 input 數字
    for (int i = 0; i < ans.size(); i++)
    {
        for (int j = 0; j < ans[i].size() - 1; j++)
            cout << ans[i][j] << " ";
        cout << ans[i][ans[i].size() - 1] << endl;
    }
}
```

1.8 數字加法組合

```cpp
void printCombination(vector<int> const &out
    , int m, vector<vector<int>> &ans)
{
    for (int i : out)
        if (i > m)
            return;
    ans.push_back(out);
}

void recur(int i, int n, int m, vector<int>
    &out, vector<vector<int>> &ans)
{
    if (n == 0)
        printCombination(out, m, ans);
    for (int j = i; j <= n; j++)
    {
        out.push_back(j);
        recur(j, n - j, m, out, ans);
        out.pop_back();
    }
}
int main()
{
    vector<vector<int>> ans;
    vector<int> zero;
    recur(1, num, num, zero, ans);
    // num 為 input 數字
    for (int i = 0; i < ans.size(); i++)
    {
        for (int j = 0; j < ans[i].size() -
            1; j++)
            cout << ans[i][j] << " ";
        cout << ans[i][ans[i].size() - 1] <<
            endl;
    }
}
```

1.9 羅馬數字

```cpp
int romanToInt(string s)
{
    unordered_map<char, int> T;
    T['I'] = 1;
    T['V'] = 5;
    T['X'] = 10;
    T['L'] = 50;
    T['C'] = 100;
    T['D'] = 500;
    T['M'] = 1000;

    int sum = T[s.back()];
    for (int i = s.length() - 2; i >= 0; --i
        )
    {
        if (T[s[i]] < T[s[i + 1]])
            sum -= T[s[i]];
        else
            sum += T[s[i]];
    }
    return sum;
}
```

1.10 質因數分解

```cpp
void cal(int in)
{
    for (long long x = 2; x <= in; x++)
    {
        while (in % x == 0)
        {
            cout << x << "*";
            in /= x;
        }
    }
}
```

2 data structure

2.1 Bigint

```cpp
//台大
struct Bigint{
    static const int LEN = 60;
    static const int BIGMOD = 10000;
    int s;
    int vl, v[LEN];
    //  vector<int> v;
    Bigint() : s(1) { vl = 0; }
    Bigint(long long a) {
        s = 1; vl = 0;
        if (a < 0) { s = -1; a = -a; }
        while (a) {
            push_back(a % BIGMOD);
            a /= BIGMOD;
        }
    }
    Bigint(string str) {
        s = 1; vl = 0;
        int stPos = 0, num = 0;
        if (!str.empty() && str[0] == '-') {
            stPos = 1;
            s = -1;
        }
        for (int i=SZ(str)-1, q=1; i>=stPos;
            i--) {
            num += (str[i] - '0') * q;
            if ((q *= 10) >= BIGMOD) {
                push_back(num);
                num = 0; q = 1;
            }
        }
        if (num) push_back(num);
        n();
    }
    int len() const {
        return vl;//return SZ(v);
    }
    bool empty() const { return len() == 0;
        }
    void push_back(int x) {
        v[vl++] = x; //v.PB(x);
    }
    void pop_back() {
        vl--; //v.pop_back();
    }
    int back() const {
        return v[vl-1]; //return v.back();
    }
    void n() {
        while (!empty() && !back()) pop_back
            ();
    }
    void resize(int nl) {
        vl = nl; //v.resize(nl);
        fill(v, v+vl, 0); //fill(ALL(v), 0);
    }
    void print() const {
        if (empty()) { putchar('0'); return;
            }
        if (s == -1) putchar('-');
        printf("%d", back());
        for (int i=len()-2; i>=0; i--)
            printf("%.4d",v[i]);
    }
    friend std::ostream& operator << (std::
        ostream& out, const Bigint &a) {
        if (a.empty()) { out << "0"; return
            out; }
        if (a.s == -1) out << "-";
        out << a.back();
        for (int i=a.len()-2; i>=0; i--) {
            char str[10];
            snprintf(str, 5, "%.4d", a.v[i])
                ;
            out << str;
        }
        return out;
    }
    int cp3(const Bigint &b)const {
        if (s != b.s) return s - b.s;
        if (s == -1) return -(-*this).cp3(-b
            );
        if (len() != b.len()) return len()-b
            .len();//int
        for (int i=len()-1; i>=0; i--)
            if (v[i]!=b.v[i]) return v[i]-b.
                v[i];
        return 0;
    }
    bool operator<(const Bigint &b)const
    { return cp3(b)<0; }
    bool operator<=(const Bigint &b)const
    { return cp3(b)<=0; }
    bool operator==(const Bigint &b)const
    { return cp3(b)==0; }
    bool operator!=(const Bigint &b)const
    { return cp3(b)!=0; }
    bool operator>(const Bigint &b)const
    { return cp3(b)>0; }
    bool operator>=(const Bigint &b)const
    { return cp3(b)>=0; }
    Bigint operator - () const {
        Bigint r = (*this);
        r.s = -r.s;
        return r;
    }
    Bigint operator + (const Bigint &b)
        const {
        if (s == -1) return -(-(*this)+(-b))
            ;
        if (b.s == -1) return (*this)-(-b);
        Bigint r;
        int nl = max(len(), b.len());
        r.resize(nl + 1);
        for (int i=0; i<nl; i++) {
            if (i < len()) r.v[i] += v[i];
            if (i < b.len()) r.v[i] += b.v[i
                ];
            if(r.v[i] >= BIGMOD) {
                r.v[i+1] += r.v[i] / BIGMOD;
                r.v[i] %= BIGMOD;
            }
        }
        r.n();
        return r;
    }
    Bigint operator - (const Bigint &b)
        const {
        if (s == -1) return -(-(*this)-(-b))
            ;
        if (b.s == -1) return (*this)+(-b);
        if ((*this) < b) return -(b-(*this))
            ;
        Bigint r;
        r.resize(len());
        for (int i=0; i<len(); i++) {
            r.v[i] += v[i];
            if (i < b.len()) r.v[i] -= b.v[i
                ];
            if (r.v[i] < 0) {
                r.v[i] += BIGMOD;
                r.v[i+1]--;
            }
        }
        r.n();
        return r;
    }
    Bigint operator * (const Bigint &b) {
        Bigint r;
        r.resize(len() + b.len() + 1);
        r.s = s * b.s;
        for (int i=0; i<len(); i++) {
            for (int j=0; j<b.len(); j++) {
                r.v[i+j] += v[i] * b.v[j];
                if(r.v[i+j] >= BIGMOD) {
                    r.v[i+j+1] += r.v[i+j] /
                        BIGMOD;
                    r.v[i+j] %= BIGMOD;
                }
            }
        }
        r.n();
        return r;
    }
    Bigint operator / (const Bigint &b) {
        Bigint r;
        r.resize(max(1, len()-b.len()+1));
        int oriS = s;
        Bigint b2 = b; // b2 = abs(b)
```

```
        s = b2.s = r.s = 1;
        for (int i=r.len()-1; i>=0; i--) {
            int d=0, u=BIGMOD-1;
            while(d<u) {
                int m = (d+u+1)>>1;
                r.v[i] = m;
                if((r*b2) > (*this)) u = m
                    -1;
                else d = m;
            }
            r.v[i] = d;
        }
        s = oriS;
        r.s = s * b.s;
        r.n();
        return r;
    }
    Bigint operator % (const Bigint &b) {
        return (*this)-(*this)/b*b;
    }
};
```

2.2 分數

```
class Rational
{
  friend istream &operator>>(istream &,
      Rational & );
  friend ostream &operator<<(ostream &,
      const Rational & );
public:
  Rational()  //constructor one
  {
    m_numeitor = 0;
    m_denominator = 1;
  }
  Rational(int a, int b)  //constructor two
  {
    if (b < 0 || b == 0)  //avoids negative
        denominators. && prevents a 0
        denominator
    {
      cout << "This Rational number can't be
          used.\n\n";
      m_numeitor = 0;
      m_denominator = 0;
    }
    else
    {
      cout << "This Rational number can be
          used.\n\n";
      m_numeitor = a;
      m_denominator = b;
    }
  }
  Rational operator+(const Rational& a);  //
      加
  Rational operator-(const Rational& a);  //
      減
  Rational operator*(const Rational& a);  //
      乘
  Rational operator/(const Rational& a);  //
      除
  bool operator==(const Rational& a);   //相
      等
  void reduce();  //化簡
private:
  int m_numeitor;
  int m_denominator;
};
istream &operator>>(istream &input, Rational
    &test )
{
  char temp;

  input >> test.m_numeitor;
  input >> temp;
  input >> test.m_denominator;
  Rational final(test.m_numeitor, test.
      m_denominator);  //final用來告訴使用者
      這數字符不符合!
  if (test.m_denominator < 0 || test.
      m_denominator == 0)   //不符合(再輸入
      一次)
  {
    while (test.m_denominator < 0 || test.
        m_denominator == 0)   //有可能輸入的
        東西還是不符合,所以用迴圈
    {
      cout << "Enter another Rational number
          (n/d): ";
      input >> test.m_numeitor;
      input >> temp;
      input >> test.m_denominator;
      Rational final(test.m_numeitor, test.
          m_denominator);  //final用來告訴使
          用者這數字符不符合!
    }
    return input;
  }
  else
    return input;
}
ostream &operator<<(ostream &output, const
    Rational &test )
{
  output << test.m_numeitor;
  if(test.m_numeitor == 0)
    return output;
  if (test.m_denominator == 1)
    return output;
  else
  {
    output << "/";
    output << test.m_denominator;
  }
  return output;
}
Rational Rational::operator+(const Rational&
    a)
{
  Rational c;
  c.m_denominator = this->m_denominator * a.
      m_denominator;  //通分(同乘)
  c.m_numeitor = (this->m_numeitor * a.
      m_denominator) + (a.m_numeitor * this
      ->m_denominator);
  c.reduce();
  return c;
}
Rational Rational::operator-(const Rational&
    a)
{
  Rational c;
  c.m_denominator = this->m_denominator * a.
      m_denominator;
  c.m_numeitor = (this->m_numeitor * a.
      m_denominator) - (a.m_numeitor * this
      ->m_denominator);
  c.reduce();
  return c;
}
Rational Rational::operator*(const Rational&
    a)
{
  Rational c;
  c.m_denominator = this->m_denominator * a.
      m_denominator;
  c.m_numeitor = this->m_numeitor * a.
      m_numeitor;
  c.reduce();
  return c;
}
Rational Rational::operator/(const Rational&
    a)
{
  Rational c;
  c.m_denominator = this->m_denominator * a.
      m_numeitor;
  c.m_numeitor = this->m_numeitor * a.
      m_denominator;
  c.reduce();
  return c;
}
bool Rational::operator==(const Rational& a)
{
  if (m_numeitor == a.m_numeitor)
  {
    if (m_denominator == a.m_denominator)
      return true;
    else
      return false;
  }
  else
    return false;
}
void Rational::reduce()
{
  int i;
  int max;
  if(m_numeitor> m_denominator)
    max = m_numeitor;
  else
    max = m_denominator;
  for (i = 2; i <= max; i++)
  {
    if (m_denominator % i == 0 && m_numeitor
        % i == 0)
    {
      m_denominator /= i;
      m_numeitor /= i;
      i = 1;
      max = m_denominator;
      continue;
    }
  }
}
```

ACM ICPC Team Reference - Angry Crow Takes Flight!

Contents