Contents

Basic

BigInt

```
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);
  }
  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; fill(v, v+vl, 0);
    //    v.resize(nl); // 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 ? 1 : -1;
    if (s == -1) return -(-*this).cp3(-b);
    if (len() != b.len()) return len()>b.len()?1:-1;
    for (int i=len()-1; i>=0; i--)
      if (v[i]!=b.v[i]) return v[i]>b.v[i]?1:-1;
    return 0;
  }
  bool operator < (const Bigint &b)const{ return cp3(b)
      ==-1; }
  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)
      ==1; }
  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;
  }
};
```

Mathmatics

Geometry

Flow

Graph

Data Structure

String

Dark Code

Search

Others

Persistence