数学

1.1高精度

const int maxn = 1000;

struct bign{

int d[maxn], len;

void clean() { while(len > 1 && !d[len-1]) len--; }

bign() { memset(d, 0, sizeof(d)); len = 1; }

bign(int num) { \*this = num; }

bign(char\* num) { \*this = num; }

bign operator = (const char\* num){

memset(d, 0, sizeof(d)); len = strlen(num);

for(int i = 0; i < len; i++) d[i] = num[len-1-i] - '0';

clean();

return \*this;

}

bign operator = (int num){

char s[20]; sprintf(s, "%d", num);

\*this = s;

return \*this;

}

bign operator + (const bign& b){

bign c = \*this; int i;

for (i = 0; i < b.len; i++){

c.d[i] += b.d[i];

if (c.d[i] > 9) c.d[i]%=10, c.d[i+1]++;

}

while (c.d[i] > 9) c.d[i++]%=10, c.d[i]++;

c.len = max(len, b.len);

if (c.d[i] && c.len <= i) c.len = i+1;

return c;

}

bign operator - (const bign& b){

bign c = \*this; int i;

for (i = 0; i < b.len; i++){

c.d[i] -= b.d[i];

if (c.d[i] < 0) c.d[i]+=10, c.d[i+1]--;

}

while (c.d[i] < 0) c.d[i++]+=10, c.d[i]--;

c.clean();

return c;

}

bign operator \* (const bign& b)const{

int i, j; bign c; c.len = len + b.len;

for(j = 0; j < b.len; j++) for(i = 0; i < len; i++)

c.d[i+j] += d[i] \* b.d[j];

for(i = 0; i < c.len-1; i++)

c.d[i+1] += c.d[i]/10, c.d[i] %= 10;

c.clean();

return c;

}

bign operator / (const bign& b){

int i, j;

bign c = \*this, a = 0;

for (i = len - 1; i >= 0; i--)

{

a = a\*10 + d[i];

for (j = 0; j < 10; j++) if (a < b\*(j+1)) break;

c.d[i] = j;

a = a - b\*j;

}

c.clean();

return c;

}

bign operator % (const bign& b){

int i, j;

bign a = 0;

for (i = len - 1; i >= 0; i--)

{

a = a\*10 + d[i];

for (j = 0; j < 10; j++) if (a < b\*(j+1)) break;

a = a - b\*j;

}

return a;

}

bign operator += (const bign& b){

\*this = \*this + b;

return \*this;

}

bool operator <(const bign& b) const{

if(len != b.len) return len < b.len;

for(int i = len-1; i >= 0; i--)

if(d[i] != b.d[i]) return d[i] < b.d[i];

return false;

}

bool operator >(const bign& b) const{return b < \*this;}

bool operator<=(const bign& b) const{return !(b < \*this);}

bool operator>=(const bign& b) const{return !(\*this < b);}

bool operator!=(const bign& b) const{return b < \*this || \*this < b;}

bool operator==(const bign& b) const{return !(b < \*this) && !(b > \*this);}

string str() const{

char s[maxn]={};

for(int i = 0; i < len; i++) s[len-1-i] = d[i]+'0';

return s;

}

};

istream& operator >> (istream& in, bign& x)

{

string s;

in >> s;

x = s.c\_str();

return in;

}

ostream& operator << (ostream& out, const bign& x)

{

out << x.str();

return out;

}

1.2快速幂

typedef long long ll;

ll fpow(ll a, ll b, ll c) {

ll res = 1;

while (b) {

if (b & 1) res = (res \* a) % c;

b >>= 1;

a = (a \* a) % c;

}

return res;

}

1.3进制转换

1.4格雷码

1.5快速傅立叶

1.6分数类

1.7全排列散列

2.1