HOMEWORK REPORT

个人信息

姓名	学号
吴宇杰	19215028

题目内容

给定两个长整数，输出整数乘法结果

解题思路

将两个整数a1,a2分割,a1分割为a,b两块,a2分割为c,d两块
因为a1×a2=ab×10^n+(a+b)(c+d)×10^n/2+cd
即可实现利用三次较少位数的乘法来得到原来的乘运算结果

实现代码

```
#include <iostream>
#include "BigInt.h"

using namespace std;

class BigINT:public BI::BigInt
{
public:
    BigINT ():BigInt() {}

    BigINT (const BigInt & n):BigInt(n) {}

    BigINT (basic_string<char> c):BigInt(string(c)) {}

    void operator << (int n) {
        values += string(n, '0');
    }

    string to_string () {
        return values;
    }

    BigINT operator* (BigINT &n) {
        if (values.size() == 1 || n.to_string().size()) {
            return BigInt(values) * BigInt(n);
```

```
        }

        BigINT a = values.substr(0, values.size()/2+1);
        BigINT b = values.substr(values.size()/2);

        string n2 = n.to_string();
        BigINT c = n2.substr(0, n2.size()/2 + 1);
        BigINT d = n2.substr(n2.size()/2);

        BigINT ac = a * c;
        BigINT bd = b * d;
        BigINT cross = (a+b)*(c+d);

        BigINT acpow = ac;
        acpow << (values.size() + n2.size()) /2;

        BigINT crosspow = cross - ac - bd;
        crosspow << (values.size() + n2.size()) / 4;

        return bd + crosspow + acpow;
    }
};

string padZero (string s, int size) {
    int pad_num = size - s.size();

    s = string(pad_num, '0') + s;

    return s;
}

string longPlus (string s1, string s2, bool neg) {
    int size = s1.size() > s2.size() ? s1.size() : s2.size();

    s1 = padZero(s1, size);
    s2 = padZero(s2, size);

    int carry = 0;
    string res;

    while (size > 0) {
        int num1 = s1[size - 1] - '0';
        int num2 = s2[size - 1] - '0';

        num2 = neg ? 10-num2 : num2;

        int sum = num1 + num2 + carry;

        res = to_string(sum % 10) + res;
        carry = neg ? sum > 10 ? 0: -1 : sum / 10;

        size--;
    }
```

```
    return carry && !neg ? "1" + res : res;
}

string multiply (string s1, string s2) {
    int size = s1.size() > s2.size() ? s1.size() : s2.size();

    if (size == 1) {
        return to_string(stoi(s1) * stoi(s2));
    }

    size = size % 2 ? size+1 : size ;

    s1 = padZero(s1, size);
    s2 = padZero(s2, size);

    string a = s1.substr(0, size/2);
    string b = s1.substr(size/2);
    string c = s2.substr(0, size/2);
    string d = s2.substr(size/2);

    string ac = multiply(a, c);
    string bd = multiply(b ,d);
    string cross_mul = multiply(longPlus(a,b, false), longPlus(c, d, false));

    string acpow = ac + string(size, '0');
    string crosspow = longPlus(longPlus(cross_mul, ac, true), bd, true) + string(siz

    return longPlus(longPlus(bd, crosspow ,false), acpow, false);
}

int main () {
    BigINT s1, s2;

    cout << "first number:";
    cin >> s1;

    cout << "second number:";
    cin >> s2;

    cout << s1*s2;

    return 0;
}
```

测试样例

```
输入:
first number: 432678212
second number: 32312312
```

输出:
13980833381746144

总结

用字符串可以实现高精度数的存储
使用分治将原来的乘法分解

$$T(n) = 3T(n/2) + O(n) = O(n^{log2\,3})$$

可将原来需要$O(n^2)$时间复杂度的乘法运算减少为$O(n^{log2\,3})$