高精度算法

20计科官昕

前言

- 这个学期，你们学到的东西将与上个学期有很大不同，除了难度增加之外，就是内容的特点了。就像高中数学往往会要求证明过程，但高数中的一些内容却是会用就行一样。这学期的很多内容，其实更偏向于工具性。由于ACM比赛是允许携带纸质材料的，所以基本上能够快速判断出题目要求使用什么知识并能够正确使用模板便足以。但是，由于理解的越深刻，使用起来自然就越得心应手，而且对算法本身有所了解，在一些需要修改一下模板或调bug时才能更快捷、有效。所以在平时学习和练习时，最好能够先不使用模板自己去写一遍，然后对照模板，理解模板的每一条语句的意义。如果平时不用，也不去理解这些模板，往往要用时也很难能想起来。

目录

概述

高精度算法是什么?

- 所谓的高精度算法，即是处理大数字的数学计算方法
- 高精度算法往往使用字符串或数组来表示数字，并有一套比较完整的算法体系来代替普通的数字运算。
- 以最常见的高精度加法为例，一般做法是将两个加数都用数组表示起来，然后使用一系列操作，逐渐由两个加数得到结果并返回。

高精度算法有什么用?

- 很简单，正如上边所说，用于大数字的数学计算。
- 所谓大数字，指的就是用常规数据类型所无法保存的数字。
  - 比如int的上限是2^31-1，long long是2^63-1，即便是unsigned long long上限也只到2^64-1，虽然再往上还有_int128之类的，但是总有上限，所以就到了高精度算法出场的时候了。
  - 像是unsigned long long的最大值，其实就只有20几位数，而利用数组，我们可以轻易地表示成千上万位数。利用这个，我们就可以轻松进行对大数字的操作了。

加法

原理

- 高精度加法的原理比较类似于小学时学过的竖式加法，我们将整个大数字用数组存放，每个下标存放其中的一位，然后将两个数字按位相加，并处理进位，这样一位位算下来，就能够得到一个代表两数之和的新数组，这就是高精度加法的原理。

实例

- 我们先举一个例子，比如283+456吧，来实际体验一下
  - 首先，个位相加：3+6=9，所以答案的个位数是9
  - 然后，十位相加：8+5=13，所以答案的十位是3，并且往百位进位一个1
  - 最后，百位相加：2+4=6，再加上进位的1，所以答案的百位是7
  - 于是，我们得到了答案283+456=739

具体实现

- 在具体实现时，我们往往使用两个数组表示两个加数，我们设为a和b，再用一个数组表示结果，我们设为c，有时还会有一个处理进位的数组，不过我一般习惯直接用结果数组c表示进位
- 存储时，由于平常数字表示与数组特性的差异，将数字倒序表示往往会比较方便，即下标为0的是个位、下标为1的是十位等等
  - 以上述例子来说a={3,8,2},b={6,5,4},c={0}
- 由于运算需要终止条件，所以一般在用数组表示数字时，还会有变量表示数字位数，设为lena、lenb和lenc。
  - 以上述例子来说lena=3,lenb=3,lenc=0

具体实现

- 然后从0到lena和lenb中较大的一方，不断按位相加，并处理进位问题：即不断重复c[i+1]=(a[i]+b[i]+c[i])/10;c[i]=(a[i]+b[i]+c[i])%10;并不断更新lenc，直到两个数加完为止。
  - 第一次后：c={9},lenc=1
  - 第二次后：c={9,3,1},lenc=2
  - 第三次后：c={9,3,7},lenc=3
- 这样，高精度加法就算是完成了，不过最后要注意判断，如果最高位发生了进位，导致答案位数增加，最后lenc还要再加上1才行

代码

```
string add(string sa,string sb)
{
    int lena=sa.size(),lenb=sb.size(),ma=max(lena,lenb);
    int a[1005]={0},b[1005]={0},c[1005]={0};
    for(int i=0;i<lena;i++) a[i]=sa[lena-i-1]-'0';
    for(int i=0;i<lenb;i++) b[i]=sb[lenb-i-1]-'0';
    for(int i=0;i<ma;i++){
        c[i]+=a[i]+b[i];
        if(c[i]>9) c[i+1]=c[i]/10,c[i]%=10;
    }
    if(c[ma]!=0) ma++;
    string s="";
    for(int i=ma-1;i>=0;i--) s+=char(c[i]+'0');
    return s;
}
```

减法

原理

- 与高精度加法类似，不过从竖式加法变为了竖式减法，相较于加法而言，在减法中会有更多小问题，比如借位的处理，结果的正负等等，因此，在进行高精度减法的书写时，需要更加的注意。

实现

- 我们加法的方式进行类比思考，比如a-b=c时，设a>=b
  - 若a[i]-b[i]>=0,则c[i]=a[i]-b[i];
  - 若a[i]-b[i]<0,则c[i]=a[i]-b[i]+10;且a[i+1]-1;
- 不断重复上述步骤，直到a的最高位，并正确判断lenc的值
- 但很显然，这样并不能处理含有负数或a<b的情况，因此要么按照题意，在使用高精度算法时注意规避这些情况的出现，要么用一个额外的变量来记录数字的正负属性并对算法进行一些相应改造。
  - 请同学们按讲课的思路自己思考，并完成兼容负数的高精度加减法

代码

```
string subtract(string sa,string sb)//a>=b时可用
{
    int lena=sa.size(),lenb=sb.size(),lenc=0;
    int a[1005]={0},b[1005]={0},c[1005]={0};
    for(int i=0;i<lena;i++) a[i]=sa[lena-i-1]-'0';
    for(int i=0;i<lenb;i++) b[i]=sb[lenb-i-1]-'0';
    for(int i=0;i<lena;i++){
        c[i]=a[i]-b[i];
        while(c[i]<0) c[i]+=10,a[i+1]-=1;
        if(c[i]!=0) lenc=i+1;
    }
    string s="";
    for(int i=lenc-1;i>=0;i--) s+=char(c[i]+'0');
    return s;
}
```

拓展

其他

- 除了上面详细讲解的两种算法，其实高精度算法还有很多其他的算法，例如：
  - 运算上还有乘法、除法、取模等
  - 大于、等于等比较运算也是或不可少的
  - 对高精度运算的进一步优化，例如压位高精...
- 接下来将选择其中一部分进行大略的讲解，希望同学们在课后能够自己尝试各种高精度算法，将其真正掌握。

乘除与取模

- 高精乘法时，也是按照竖式的方式，按位相乘，再将每一位乘出的结果按照进位相加，要注意每一位乘起来的结果不能直接相加，一定要按照数位相加
  - 例如：123*23，按照竖式按位相乘得到369和246，但相加时，十位的2与123相乘得到的246要比个位的3与123相乘得到的369要高一位，实际应是2460+369才对
- 高精除法比较复杂，要按照竖式的做法，从高位到低位并一步步将余数落下，请按照之前讲的流程，试着自己摸索出做法
- 高精取模中有一个小技巧，根据高斯定理，每位取模后相加再取模结果与直接取模相同，因此可以从高位开始一步步逐渐完成取模的操作

小数运算

- 不光整数有高精度，小数位数过多时也不能用默认的数据类型表示，所以小数也有高精度算法，一般处理方式是将小数点移位，进行整数高精度运算，然后再将小数点按照规则回移，如：
  - 0.8+0.7，先将小数点右移一位，按照8+7算为15，再将小数点左移一位即为1.5
  - 请思考一下算乘法时又该如何操作？

压位高精

- 压位高精是高精度算法的一种优化方式，当我们用数组表示数字时，请同学们思考一下，一个下标真的就只能存储一位数字吗？
- 即使使用int数组，每一位也可以存储9~10位数字，因此实际上我们可以在每一位上存储更多位数的数字（一般压8位或压9位），进而在不超过数据类型的限制的情况下，减少储存数字所用的空间并减少循环运算的次数，进而使算法的时间与空间复杂度大大减小
- 压位高精实际上可以看做一种进制的转换，普通高精是10进制，而压位高精时，可以看做我们使用的则是100进制、1000进制甚至更多

_int128

- 有的时候，我们要处理的数字虽然超过了long long的上限，但并没有超过很多，或者虽然处理的数字是long long范围内的，但在运算过程中可能会超过上限造成错误，由于这确实超过了正常处理上限，但用高精度又似乎有些小题大做，因此我们可以使用_int128类型来进行操作，以规避使用高精度算法。
- _int128在使用时类似于其他整数类型的数据类型，只有在输入输出时可能要略作调整，使用字符串方式进行读写操作。

_int128的输入输出函数

引用自18级师哥给我们讲课的课件

```cpp
#include<stdio.h>
typedef __int128 Int128;
int scan(Int128 &x){
    x=0;
    int sgn=1;
    char ch;
    while(ch=getchar()){
        if(ch==EOF) return EOF;
        else if(ch=='-') sgn=-sgn;
        else if(ch>='0' && ch<='9'){ x=x*10+(ch-'0'); break; }
    }
    while((ch=getchar())>='0' && ch<='9'){
        x=x*10+(ch-'0');
    }
    x*=sgn;
    return 1;
}
```

```cpp
void _print(Int128 x){
    if(x>9) _print(x/10);
    putchar(x%10+'0');
}
void print(Int128 x){
    if(x<0){ x=-x; putchar('-'); }
    _print(x);
}

int main(){
    Int128 a,b;
    while(scan(a)!=EOF){
        scan(b);
        print(a+b);        # 使用方法与int类型完全一样
        printf("\n");
    }
}
```

其他语言的高精度算法

- 说到高精，自然就是Python，在Python语言中，所有的变量本质上都是字符串，并没有非常严格的类型之分。因此，在Python中进行运算时，实质上相当于语言自带高精度算法。因此在解决一些较为简单但需要使用高精度算法时，如果会Python语言，可以直接使用，ACM比赛是允许使用包括C、C++、Python、JAVA等多种语言进行作答的。
- 在这种解决方式中，JAVA也可以作为候选，因为在JAVA的math包中是有高精度对应的类和函数可以直接调用，会JAVA的同学也可以试着使用。
- 不过无论是Python还是JAVA，都可能使运行时间增加，请一定注意

模板

两个模板

- 这里是从网上找到的模板，同学们可以以此为参考进行学习，或者去网上自己寻找其他模板，高精度算法其实是一种比较基础简单的算法，所以在写法上本质差异不大，但也有各种优化方式，希望同学们之后能够自主学习这方面的内容。
  - 模板使用了一些和类有关的内容，如果不能理解可以先暂时略过，着重去理解每个函数内部的写法，领悟其本质的内容。
  - 高精模板
  - 压位高精

讲解结束，同学们可以开始快乐做题了