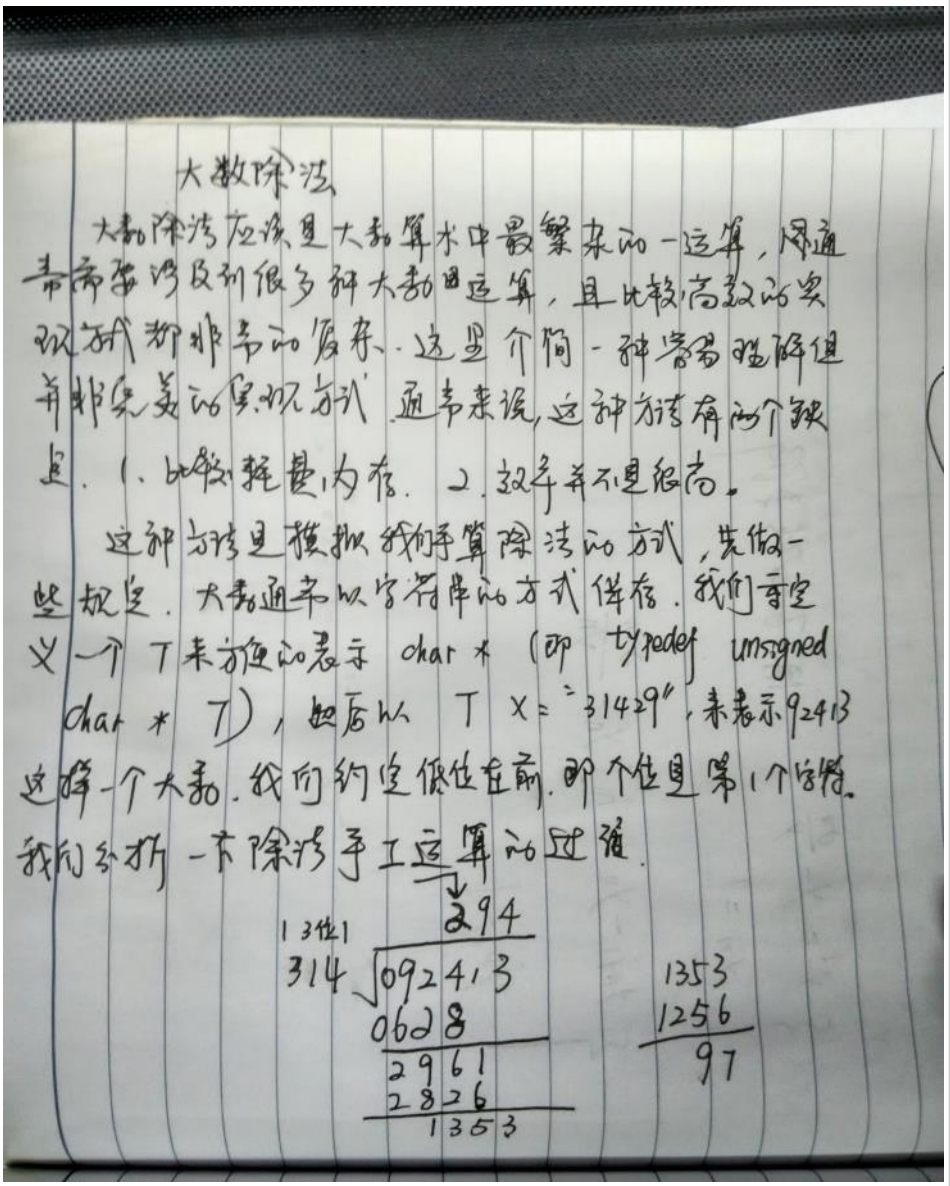
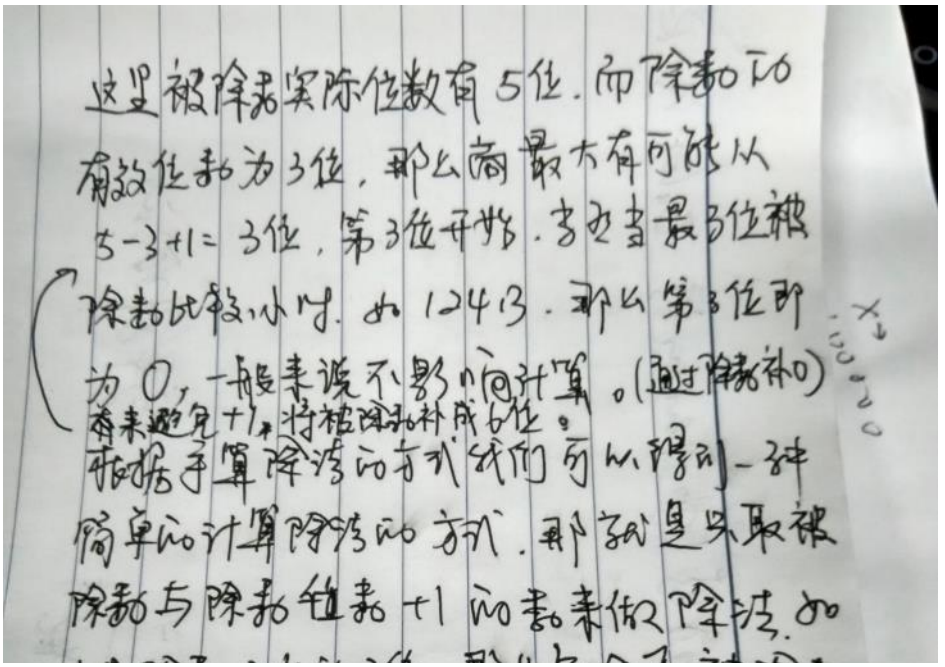
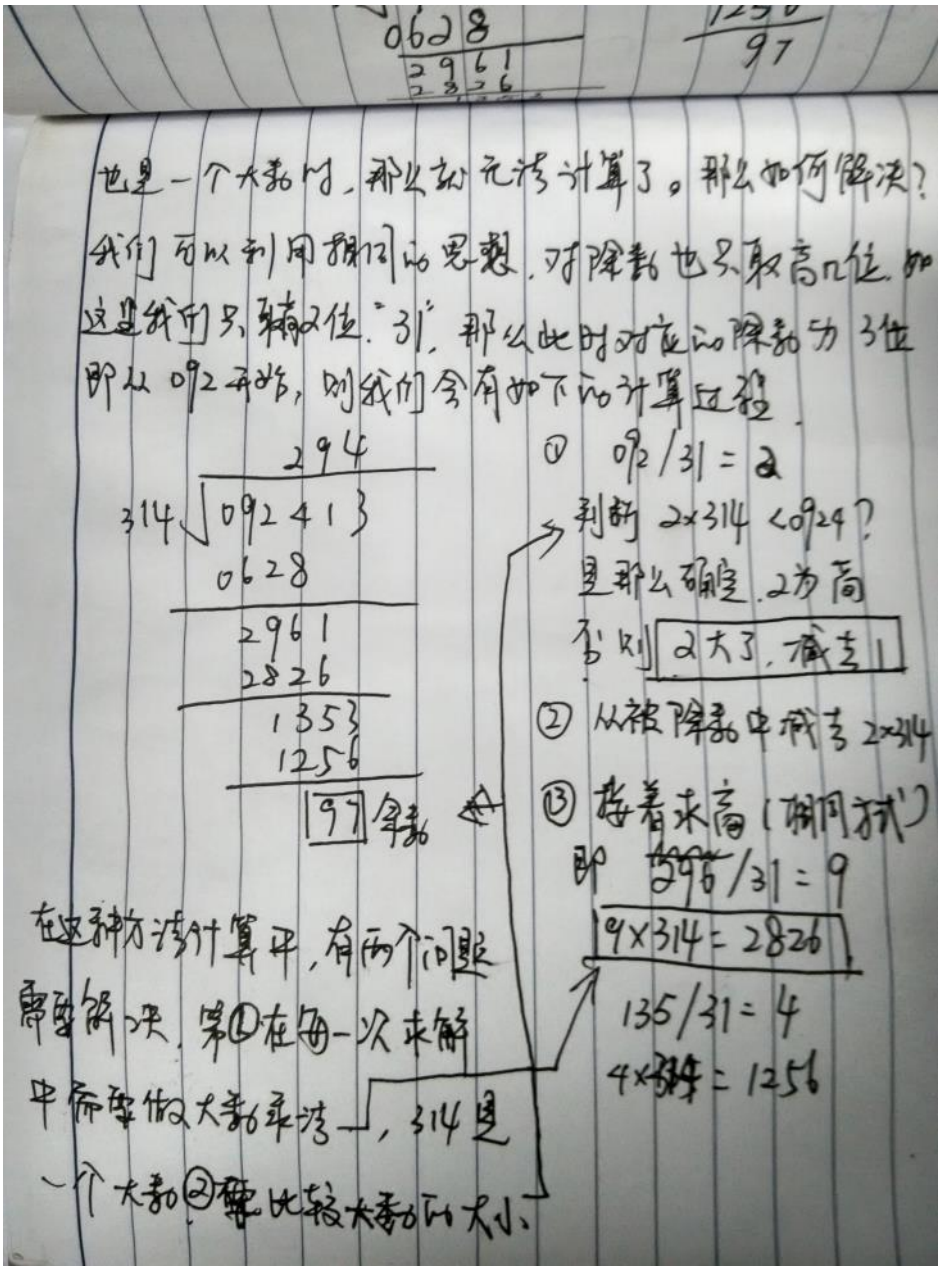
大数相除

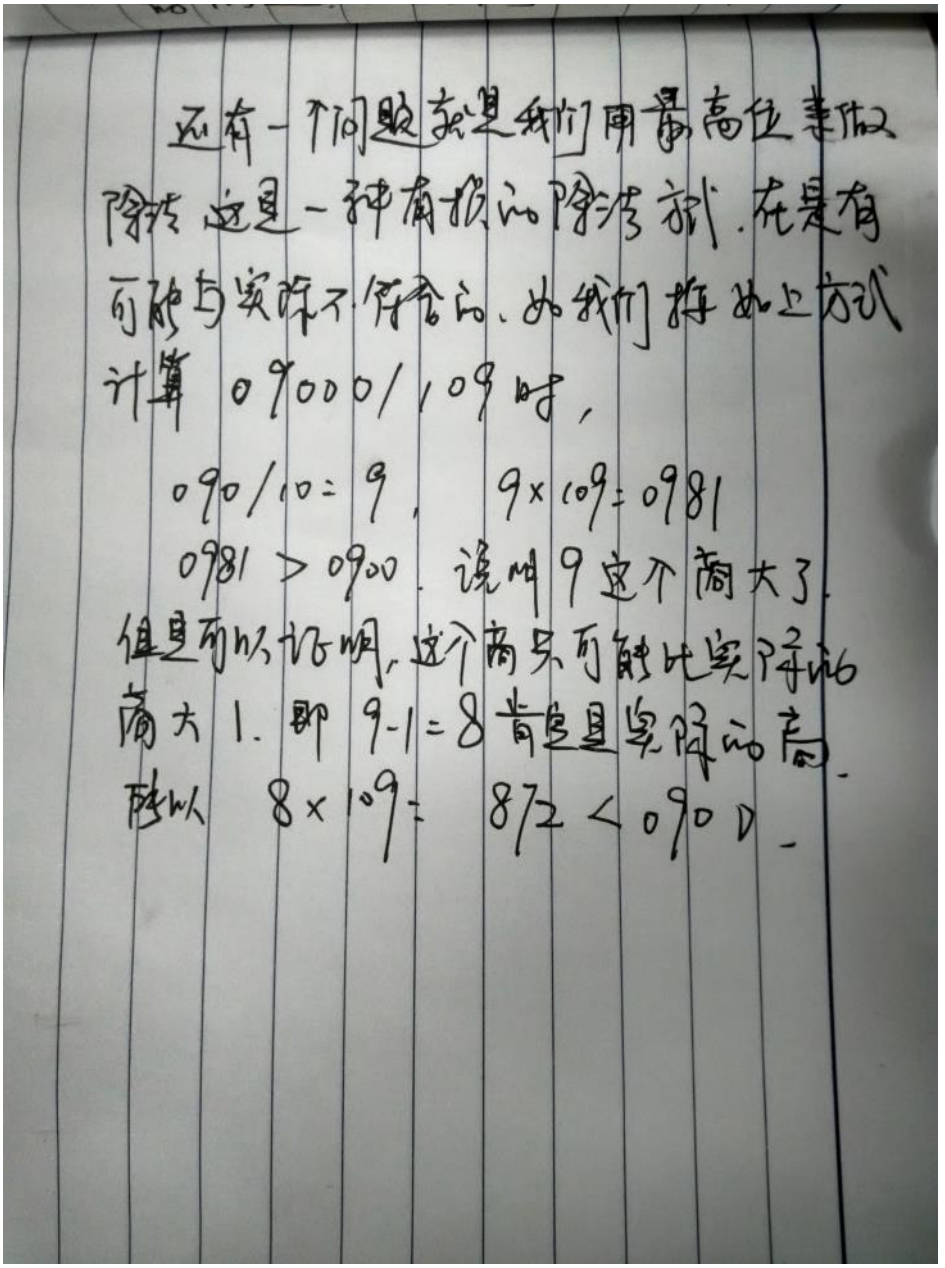
2016年7月24日　　15:57





除数与除数位数+1位数来做除法。如这是除数314为3位，那么每次取被除数4位，即0924，2961，1353，来与除数相除。那么不管被除法有多大，我每一次相除只需要4位，在一个INT类型的数据内部完全能够存入，便可以计算。用这种方法巧妙解决了被除数是大数的问题。但是我们如果遇到除数314





x为被除数，y为除数，n和m分别为被除数和除数字符串空间的总
大小，（并非实际有几位数),q为商的结果，r为余数的结果，tmp为存储被除数+临时商的空间的临时
缓存，这些变量都需要需要自己申请空间，然后传入。这里会多耗费一个被除数+一个除数大小的空间（即tmp）。

```
typedef unsigned char *T;
int XP_div(int n,T q,T x,int m, T y,T r, T tmp)
{
    int nx=n,my=m;
    n=XP_length(n,x);  //求取被除数实际位数
    m=XP_length(m,y);  //求取除数实际位数
   int k;
   unsigned char *rem =tmp,*dq=tmp+n+1;//用rem指向被除数第一个数，用dq指向结尾后两个数
   assert(2<=m&&m<=n);
     memcpy(rem,x,n); //将被除数赋值给rem
     rem[n]=0; //最高位补零，为方便计算
     //正式开始计算
     for(k=n-m;k>=0;k--) //商是从n-m位开始的，然后依此往低位移动 即k
     {
      int qk;
      int i;
       assert(2<=m&&m<=k+m&&k+m<=n);
       {
        int km=k+m;
        unsigned long y2=y[m-1]*BASE+y[m-2];//除数最高两位的值 BASE此处为10，即10进制
        unsigned long r3=rem[km]*(BASE*BASE)+rem[km-1]*BASE+rem[km-2];//从k+m开始一次后移1位的三位数的值，即被除数的
```

```
高三位
        qk=r3/y2; //被除数高三位/除数高二位
        if(qk>=BASE)  //当出现大于10
            qk=BASE-1;
      }
    dq[m]=XP_product(m,dq,y,qk);  //计算商为qk时 qk*y的大小存入dq  即商*除数y的大小 即dq=y*qk
    for(i=m;i>0;i--) //判断dq是否比被除数小
        if(rem[i+k]!=dq[i])   //从最高位开始比较，当不相同时就退出
            break;
    if(rem[i+k]<dq[i])   //退出后判断不相同的这一位是被除数rem大还是dq大
    {
        dq[m]=XP_product(m,dq,y,--qk);  //如果是dq大 说明商qk取大了，--qk ，重新计算dq
    }
    q[k]=qk; //将商qk存下
    {
        int borrow;
        assert(0<=k&&k<=k+m);
        borrow=XP_sub(m+1,&rem[k],&rem[k],dq,0); //从除数中高k+m位减去dq 即目前的得到的余数的大小即rmk=rmk-dq
        assert(borrow==0);
    }
   //后移动一位重复这种计算。
}
```