模拟与高精度

**模拟：**是指让程序完整地按照题目叙述的方式运行得到最终答案。

在各类算法竞赛中，常作为签到送分题。但也有很多模拟题目的状态和操作非常复杂繁琐，实现难度和代码长度很大。

**高精度运算：**用数组来模拟非常长的整数。

1. **A+B Problem高精。分别再两行内输入两个500位以内的十进制非负整数，求它们的和。**

#include<bits/stdc++.h>  
using namespace std;  
#define maxn 520  
int a[maxn],b[maxn],c[maxn];  
int main(){  
 string A,B;  
 cin>>A>>B;  
 int len=max(A.length(),B.length());  
 for(int i=A.length()-1,j=1;i>=0;i--,j++)  
 a[j]=A[i]-'0';  
 for(int i=B.length()-1,j=1;i>=0;i--,j++)  
 b[j]=B[i]-'0';  
 for(int i=1;i<=len;i++) {  
 c[i]+=a[i]+b[i];  
 c[i+1]=c[i]/10;//模拟进位  
 c[i]%=10;  
 }  
 if(c[len+1])//最后进位可能会导致位数增加  
 len++;  
 for(int i=len;i>=1;i--)  
 cout<<c[i];  
}

1. **A\*B Problem。分别在两行内输入两个2000位以内的十进制非负整数，求它们的积。**

#include<bits/stdc++.h>  
using namespace std;  
#define maxn 5010  
int a[maxn],b[maxn],c[maxn];  
int main(){  
 string A,B;  
 cin>>A>>B;  
 int lena=A.length(),lenb=B.length();  
 for(int i=lena-1;i>=0;i--)a[lena-i]=A[i]-'0';  
 for(int i=lenb-1;i>=0;i--)b[lenb-i]=B[i]-'0';  
 for(int i=1;i<=lena;i++)  
 for(int j=1;j<=lenb;j++)  
 c[i+j-1]+=a[i]\*b[j];//计算贡献  
 int len=lena+lenb;//乘积的位数不超过两数的位数之和  
 for(int i=1;i<=len;i++){  
 c[i+1]+=c[i]/10;//处理进位  
 c[i]%=10;  
 }  
 for(;!c[len];)  
 len--;//去掉前导零  
 for(int i=max(1,len);i>=1;i--)  
 cout<<c[i];  
}

这样的高精度算法的复杂度是O(n^2)，其中n是数字的位数。如果位数过大，那么计算速度还是比较慢。可以利用快速傅里叶变换的方式加速高精度乘法，将其复杂度优化到O(nlogn)，其思维和实现难度都比较大。

1. **阶乘之和。用高精度计算出S=1!+2!+3!+……+n!(n<=50)。**

#include<bits/stdc++.h>  
using namespace std;  
#define maxn 100  
struct Bigint{  
 int len,a[maxn];/\*为了兼顾效率与代码复杂度，用len记录位数，a记录每个数位\*/  
 Bigint(int x=0){//通过初始化使得这个大整数能够表示整型x,默认为0   
 memset(a,0,sizeof(a));  
 for(len=1;x;len++)  
 a[len]=x%10,x/=10;  
 len--;  
 }  
 int &operator[](int i){  
 return a[i];//重载[],可以直接用x[i]代表x.a[i],编写时更加自然   
 }  
 void flatten(int L){/\*一口气处理1到L范围内的进位并重置长度。需要保证L不小于有效长度\*/  
 //因为相当于把不是一位数的位都处理成一位数，故取名为“展平”   
 len=L;  
 for(int i=1;i<=len;i++)  
 a[i+1]+=a[i]/10,a[i]%=10;  
 for(;!a[len];)//重置长度成为有效长度   
 len--;  
 }  
 void print(){//输出   
 for(int i=max(len,1);i>=1;i--)  
 printf("%d",a[i]);  
 }  
};  
Bigint operator+(Bigint a,Bigint b){/\*表示两个Bigint类相加，返回一个Bigint类\*/  
 Bigint c;  
 int len=max(a.len,b.len);  
 for(int i=1;i<=len;i++)  
 c[i]+=a[i]+b[i];//计算贡献   
 c.flatten(len+1);/\*答案不超过len+1位，所以用len+1做一遍“展平”处理进位\*/  
 return c;  
}  
Bigint operator\*(Bigint a,int b){/\*表示Bigint类乘整型变量，返回一个Bigint类\*/  
 Bigint c;  
 int len=a.len;  
 for(int i=1;i<=len;i++)  
 c[i]=a[i]\*b;//计算贡献   
 c.flatten(len+11);/\*int类型最长10位，所以可以这样做一遍“展平”处理进位\*/  
 return c;  
}  
int main(){  
 Bigint ans(0),fac(1);/\*分别用0和1初始化ans与fac，如果要将常数赋值给大整数，  
 可以使用类似于ans=Bigint(233)的办法\*/  
 int m;  
 cin>>m;  
 for(int i=1;i<=m;i++){  
 fac=fac\*i;//模拟题意   
 ans=ans+fac;  
 }  
 ans.print();//输出答案   
}