**1.高精度算法**

**struct hp{//high-precision**

int c[maxl];

hp(){

memset(c,0,sizeof(c));

}

};

**void hout(const hp &z){//高精度输出**

if(!z.c[0]){

printf("0");

return;

}

for(int i=z.c[0];i>=1;i--)printf("%d",z.c[i]);

}

**hp turn(int x){**

hp z;

while(x){

z.c[0]++;

z.c[z.c[0]]=x%10;

x/=10;

}return z;

}

**hp empty(hp &z){**

memset(&z,0,sizeof(z));

}

**hp read(string s){**

hp z;

z.c[0]=s.length();

for(int i=z.c[0];i>=1;i--){

z.c[i]=s[z.c[0]-i]-'0';

}

return z;

}

**bool cmp(const hp &x,const hp &y){**

for(int i=max(x.c[0],y.c[0]);i>=1;i--)//从最高位判定

{

if(x.c[i]>y.c[i])return true;

if(x.c[i]<y.c[i])return false;

}

return true;

}

**hp max(const hp &x,const hp &y){**

if(cmp(x,y))return x;

return y;

}

//----------------------------------------------------------

**hp operator+(const hp& x,const hp& y){**

hp z;

z.c[0]=max(x.c[0],y.c[0]);//1.决定基础长度

for(int i=1;i<=z.c[0];i++){

z.c[i]+=x.c[i]+y.c[i];//2.for有进位，两个相加

z.c[i+1]+=z.c[i]/10;//3.进位

z.c[i]%=10;

}if(z.c[z.c[0]+1])z.c[0]++;//4.最高一位处理

return z;

}

**hp operator-(const hp&x,const hp&y){**

hp z;

z.c[0]=max(x.c[0],y.c[0]);//1.基础长度

for(int i=1;i<=z.c[0];i++){

z.c[i]+=x.c[i]-y.c[i];//2.for相减

if(z.c[i]<0){

z.c[i]+=10;//3.借位

z.c[i+1]--;

}

}while(!z.c[z.c[0]])z.c[0]--;//4.最高一位处理

return z;

}

**hp operator\*(const hp&x,const hp&y){**

hp z;

z.c[0]=x.c[0]+y.c[0];//1.基础长度

for(int i=1;i<=x.c[0];i++){

for(int j=1;j<=y.c[0];j++){

z.c[i+j-1]+=x.c[i]\*y.c[j];//2.双重for制造算式

}

}

for(int i=1;i<z.c[0];i++){

z.c[i+1]+=z.c[i]/10;

z.c[i]%=10;//3.借位

}if(z.c[z.c[0]]==0)z.c[0]--;//4.范围是i+j-1,i+j

return z;

}

**hp operator/(const hp&x,const hp&y){**

hp z,mod;

int len=x.c[0];

while(len){

mod=turn(10)\*mod+turn(x.c[len]); //2.每次都增加一位来运算

while(cmp(mod,y)){

if(z.c[0]==0)z.c[0]=len;//1.从第一个可以除那一位一次性决定位数

mod=mod-y;

z.c[len]++;//3.逐次减到余数不能用为之

}

len--;//4.下一位

}

return z;

}

1. 二分查找

void function(int n){

lower\_bound(c,c+n,123);//>=

int loc= upper\_bound(c,c+n,233)-c;//>

}

//二分找一个>= x的值

int find1(int x){

int l=-10000,r=123456;

while(l<r){

int mid=l+(r-l+1>>1);//向上取 ,让r来变

if(cal1(x,mid))r=mid-1;

//这个值产生的结果满足条件，那么r可以往下一个看看

else l=mid;

//cout<<l<<' '<<mid<<' '<<r<<endl;

}

return r+1;

}

//二分找一个<= x的值

int find2(int x){

int l=-10000,r=123456;

while(l<r){

int mid=l+(r-l>>1);//向下取 ,让l来变

if(cal2(x,mid))l=mid+1;

//这个值产生结果满足，l可以往上看看

else r=mid;

}

return l-1;

}

//二分在数组中找一个>=x的值

int cfind1(int x,int c[],int n){

int l=-1,r=n;//假如数组开n，那么范围就是 0，n-1,左右增减1单位范围，就是-1和n

while(l<r){

int mid=l+(r-l+1>>1);

if(cal1(x,c[mid]))r=mid-1;

else l=mid;

}

if(r==n)return -233;

return r+1;

}

int cfind2(int x,int c[],int n){

int l=-1,r=n;

while(l<r){

int mid=l+(r-l>>1);

if(cal2(x,c[mid]))l=mid+1;

else r=mid;

}

if(l==-1)return -233;

return l-1;

}

1. 差分

修改：

c[l]++;

c[r+1]--;

还原：

c[i]+=c[i-1];

**一维做二维：**

for(int i=1;i<=m;i++){

scanf("%d%d%d%d",&x1,&y1,&x2,&y2);

for(int j=x1;j<=x2;j++){

b[j][y1]++;

b[j][y2+1]--;

}

}

**二维：**

for(int i=1;i<=m;i++){

scanf("%d%d%d%d",&x1,&y1,&x2,&y2);

b[x1][y1]++;

b[x1][y2+1]--;

b[x2+1][y1]--;

b[x2+1][y2+1]++;

}

for(int i=1;i<=n;i++){

for(int j=1;j<=n;j++){

a[i][j]=a[i-1][j]+a[i][j-1]+b[i][j]-a[i-1][j-1];

printf("%d ",a[i][j]);

}printf("\n");

}return 0;

1. ST表

查询：

int query(int l,int r){

int cnt=l2[r-l+1];

return max(d[l][cnt],d[r+1-(1<<cnt)][cnt]);

//mid+2^k-1==r

}

预处理log2数组：

int j=0;

for(int i=1;i<=n;i++){

if(i>(1<<j+1))j++;

//2^j<i<=2^(j+1) 1,2=>0 3,4=>1 5,6,7,8=>2 9,10,11,12,13,14,15,16 =>3

l2[i]=j;

}

合并

for(int i=1;i<=21;i++){//(1<<i)<=n

for(int j=1;j+(1<<i)-1<=n;j++){

//从1（2长度）开始合并0，然后一直合并下去

d[j][i]=max(d[j][i-1],d[j+(1<<i-1)][i-1]) ;

//mid=l+(1<<i-1),1<<i长

}

}