算法分析与设计

1. 现代密码学需要对超过100位的十进制大整数作乘法运算。分治法可用于求解这个问题。例如两位十进制数 a=a1a0 和b=b1b0 可用如下方式进行计算：c=a\*b=c210^2 +c1\*10+c0,其中， c2=a1\*b1,c0=a0\*b0,c1=(a1+a0)(b1+b0)-(c2+c0). (1)请设计一个分治算法，计算任意两个n位二进制数的乘法，假定n是2的幂；（2）列出递推式，计算算法的时间复杂度。
2. 设两个二进制数x,y。将x，y各分成两端，每段n/2位

x=A\*2^(n/2)+B y=C\*2^(n/2)+D

xy=AC\*2^(n)+(AD+BC)\*2^(n/2)+BD

源程序代码：

#include<string>

#include<iostream>

#include<math.h>

using namespace std;

#define M 100

#define N 100

int number(int a[]) //计算二进制位数

{

int i=0;

while(a[i]==1||a[i]==0)

i++;

return i;

}

int \*add(int \*a,int \*b) //二进制数相加

{

int ab[M];

int bb=0;//bb表示进位

int bn=number(b);

int an=number(a);

for(int i=0;i<an;i++)

{

ab[i]=b[i]+a[i]+bb;

if(b[i]&&a[i]==1);

else bb=0;

if(ab[i]>=2){ //两数之和大于2则进位

bb=ab[i]/2;

ab[i]=ab[i]%2;

}

}

for(int i=an;i<bn;i++){

ab[i]=b[i]+bb;

bb=0;

if(ab[i]>=2){

bb=ab[i]/2;

ab[i]=ab[i]%2;

}

}

int nab=number(ab);

if(bb==1){

ab[nab]=bb;

}

return(ab);

}

int \*Mult(int n,int \*a,int \*b) //二进制数相乘

{ int mid,i,j;

int aa[M],bb[M]; //第一个二进制数高位和低位

int cc[M],dd[M]; //第二个二进制数高位和低位

int \*a1,\*b1,\*c1,\*d1;

int ac[M],ad[M],bc[M],bd[M]; //aa\*cc aa\*dd bb\*cc bb\*dd结果

int fac[M],fad[M],fbc[M],fbd[M]; //aa\*cc的倒置

int result[M];

int k=0;

if(n==1)//一位二进制数

{

result[k]=a[0]\*b[0];

k++;

return(result);

}

else

{

mid=(n+1)/2; //将两个二进制数分为高位和低位

for(i=0;i<mid;i++){

aa[i]=a[i];

bb[i]=a[i+mid];

cc[i]=b[i];

dd[i]=b[i+mid];

}

//两个二进制数的高低位分别相乘

i=0;

a1=Mult(mid,aa,cc);

while(\*a1==1||\*a1==0){ //ac

ac[i]=\*a1; //A[]存放第一个高位与第二个高位相乘结果

\*a1++;

i++;

}

i=0;

b1=Mult(n/2,aa,dd);

while(\*b1==1||\*b1==0){

ad[i]=\*b1;

\*b1++;

i++;

}

i=0;

c1=Mult(n/2,bb,cc);

while(\*c1==1||\*c1==0){

bc[i]=\*c1;

\*c1++;

i++;

}

i=0;

d1=Mult(n/2,bb,dd);

while(\*d1==1||\*d1==0){

bd[i]=\*d1;

\*d1++;

i++;

}

//获取相乘结果的倒置

j=0;

int numbera=number(ac); //aa\*cc结果倒置

for(i=numbera-1;i>=0;i--){

fac[j]=ac[i];

j++;

}

j=0;

int numberb=number(ad);

for(i=numberb-1;i>=0;i--){

fad[j]=ad[i];

j++;

}

j=0;

int numberc=number(bc);

for(i=numberc-1;i>=0;i--){

fbc[j]=bc[i];

j++;

}

j=0;

int numberd=number(bd);

for(i=numberd-1;i>=0;i--){

fbd[j]=bd[i];

j++;

}

int c=number(fac); //将fac右移动n位

for(i=c-1;i>=0;i--){

fac[i+n]=fac[i];

}

for(i=0;i<n;i++){

fac[i]=0;

}

int \*r1,r11[M];

r1=add(fad,fbc); //r1=fad+fbc

j=0;

while(\*r1==1||\*r1==0){

r11[j]=\*r1;

\*r1++;

j++;

}

int numberr1=number(r11); ///将fad+fbc 右移n/2

for(i=numberr1-1;i>=0;i--){

r11[i+n/2]=r11[i];

}

for(i=0;i<n/2;i++){

r11[i]=0;

}

int \*r2,r22[M];

r2=add(r11,fac); // fac\*2^n+(fad+fbc)\*2^(n/2)

j=0;

while(\*r2==1||\*r2==0){

r22[j]=\*r2;

\*r2++;

j++;

}

int \*r3,r33[M];

r3=add(fbd,r22); //fac\*2^n+(fad+fbc)\*2^(n/2)+fbd

j=0;

while(\*r3==1||\*r3==0){

r33[j]=\*r3;

\*r3++;

j++;

}

int RN=number(r33); //再次倒置

int sum[M];

j=0;

for(i=RN-1;i>=0;i--){

sum[j]=r33[i];

j++;

}

return (sum);

}

}

int main()

{

int i=0,n;

int \*mult;

char a[M],b[M];//两个二进制数

int A[M],B[M],m[M];

cout<<"输入第一个二进制数："<<endl;

cin.getline(a,N); //读取字符

while(a[i]=='1'||a[i]=='0')

{

A[i]=(int)a[i]-48; //字符1转化成数字1 ,字符行‘1’为49

i++;

}

i=0;

cout<<"输入第二个二进制数："<<endl;

cin.getline(b,N);

while(a[i]=='1'||a[i]=='0')

{

B[i]=(int)b[i]-48;

i++;

}

n=number(A);

mult=Mult(n,A,B);

i=0;

while(\*mult==1||\*mult==0)

{

m[i]=\*mult;

\*mult++;

i++;

}

cout<<"相乘结果："<<endl;

i=0;

while(m[i]==1||m[i]==0)

{

cout<<m[i];

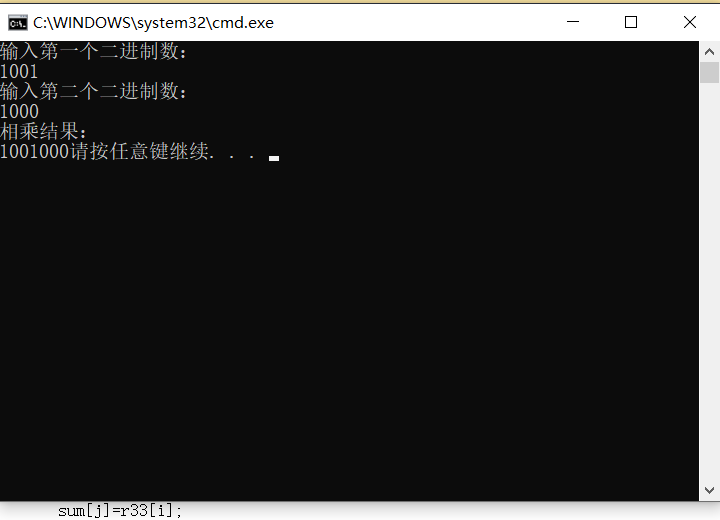
i++;

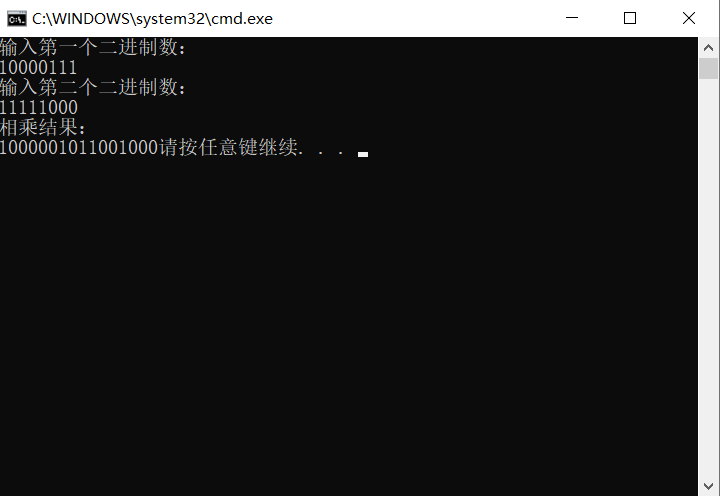
}

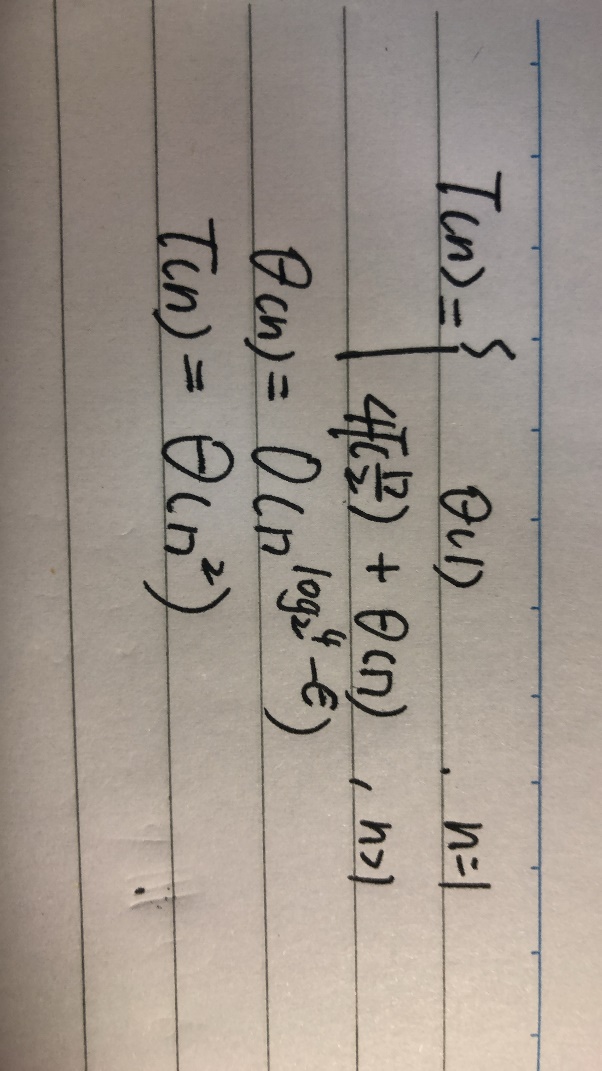
return 0;

}

算法结果：





1. 

2.机器人一次可以走1m，2m或3m。编写一个动态规划算法求机器人走n米有多少种走法 ？

源程序代码：

#include<iostream>

using namespace std;

#define N 100

int m[100];

int walk(int n)

{

m[0]=1;

m[1]=1;

m[2]=2;

if(n==1)

return 1;

else if(n==2)

return 2;

else

{

int i=3;

for(i;i<=n;i++)

{

m[i]=m[i-1]+m[i-2]+m[i-3];

}

i=i-1;5

return m[i];

}

}

int main()

{

int n;

cout<<"请输入机器人的总路程:"<<endl;

cin>>n;

cout<<"走法共:"<<walk(n)<<"种"<<endl;

return 0;

}

算法结果：

