**斐波那契数列公式：**

**F(n) = (((1+Sqrt(5))/2)^n -** **((1-Sqrt(5))/2)^n)\*1/Sqrt(5)**

**略去（（(1-Sqrt(5))/2)^n)\*1/Sqrt(5)**

**F(n)≈((1+Sqrt(5))/2)^n/sqrt(5);** **bool Interact(struct node a,struct node b,struct node c,struct node d)//叉乘法判断c,d为端点的线段是否在a,b线段所在的直线上**

**{ //若要判断 (a,b)线段与(c,d)线段是否相交，则调用**

**double x1,y1,x2,y2,x3,y3,t; // Interact(a,b,c,d)&&Interact(c,d,a,b).**

**x1=a.x-b.x;**

**y1=a.y-b.y;**

**x2=c.x-b.x;**

**y2=c.y-b.y;**

**x3=d.x-b.x;**

**y3=d.y-b.y;**

**t=(x1\*y2-x2\*y1)\*(x1\*y3-x3\*y1);**

**if(t<0)**

**return true;**

**if((x1\*y2-x2\*y1)==0)**

**{**

**if(c.x>min(a.x,b.x) && c.x<max(a.x,b.x))**

**return true;**

**}**

**if((x1\*y3-x3\*y1)==0)**

**{**

**if(d.x>min(a.x,b.x) && d.x<max(a.x,b.x))**

**return true;**

**}**

**return false;**

**}**

**矩阵+二分求幂：**

#include<stdio.h>

#define MOD 10000

struct matrix

{

int a,b,c,d;

}mat[31];

struct matrix matrix\_mul(struct matrix x,struct matrix y) //矩阵乘法

{

struct matrix ans;

ans.a=x.a\*y.a%MOD+x.b\*y.c%MOD;ans.a%=MOD;

ans.b=x.a\*y.b%MOD+x.b\*y.d%MOD;ans.b%=MOD;

ans.c=x.c\*y.a%MOD+x.d\*y.c%MOD;ans.c%=MOD;

ans.d=x.c\*y.b%MOD+x.d\*y.d%MOD;ans.d%=MOD;

return ans;

}

int main()

{

int i,n;

mat[0].a=0;mat[0].b=mat[0].c=mat[0].d=1; mat[i]表示矩阵的^i

for(i=1;i<=30;i++)

{

mat[i].a=mat[i-1].a\*mat[i-1].a%MOD+mat[i-1].b\*mat[i-1].c%MOD;mat[i].a%=MOD;

mat[i].b=mat[i-1].a\*mat[i-1].b%MOD+mat[i-1].b\*mat[i-1].d%MOD;mat[i].b%=MOD;

mat[i].c=mat[i-1].c\*mat[i-1].a%MOD+mat[i-1].d\*mat[i-1].c%MOD;mat[i].c%=MOD;

mat[i].d=mat[i-1].c\*mat[i-1].b%MOD+mat[i-1].d\*mat[i-1].d%MOD;mat[i].d%=MOD;

}

while(scanf("%d",&n) && n!=-1)

{

if(n==0)

{

printf("0\n");

continue;

}

n-=1;

i=0;

struct matrix ans;

ans.a=ans.d=1;ans.b=ans.c=0;

while(n>0) // 二分快速幂乘

{

if(n&1)

{

ans=matrix\_mul(ans,mat[i]);

}

n>>=1;

++i;

}

printf("%d\n",ans.d);

}

return 0;

}

千进制

**int main()**

**{**

**char s[1002];**

**int a[200],r,t,i;**

**while(scanf("%s%d",s,&r))//输入大整数，求余r**

**{**

**//千进制转化**

**memset(a,0,sizeof(a));**

**int len=strlen(s);**

**for(i=0;i<len;i++)**

**a[(len-i+2)/3-1]=a[(len-i+2)/3-1]\*10+s[i]-'0';**

**len=(len+2)/3;**

**/\* for(i=len-1;i>=0;i--)**

**{**

**cout<<setfill('0')<<setw(3)<<a[i]<<",";**

**}**

**\*/**

**//求余**

**int ans=0;**

**for(i=len-1;i>=0;i--)**

**{**

**ans=(ans\*1000+a[i])%r;**

**}**

**cout<<ans<<endl;**

**}**

**return 0;**

**}**

**double fsimp(double a,double b,double eps) /\*辛普森算法\*/**

**/\*a为积分下限，b为积分上限，eps是希望达到的精度\*/**

**{**

**int n,k;**

**double h,t1,t2,s1,s2,ep,p,x;**

**n=1; h=b-a;**

**t1=h\*(f(a)+f(b))/2.0; /\*用梯形公式求出一个大概的估值\*/**

**s1=t1;**

**ep=eps+1.0;**

**while (ep>=eps)**

**{**

**/\*用梯形法则计算\*/**

**p=0.0;**

**for (k=0;k<=n-1;k++)**

**{**

**x=a+(k+0.5)\*h;**

**p=p+f(x);**

**}**

**t2=(t1+h\*p)/2.0;**

**/\*用辛普森公式求精\*/**

**s2=(4.0\*t2-t1)/3.0;**

**ep=fabs(s2-s1);**

**t1=t2; s1=s2; n=n+n; h=h/2.0;**

**}**

**return(s2);**

**}**

**#include<iostream>**

**#include<string>**

**using namespace std;**

**int c[1000][1000];**

**int z[1000][1000];//记录状态**

**string a,b;**

**/\***

**int lcs(int i,int j)**

**{**

**if(i==0 || j==0)**

**return 0;**

**if(a[i-1]==b[j-1])**

**return c[i-1][j-1]+1;**

**return c[i][j-1]>c[i-1][j]?c[i][j-1]:c[i-1][j];**

**}**

**\*/**

**int main()**

**{**

**int n,m,i,j;**

**while(cin>>a>>b)**

**{**

**n=a.size();**

**m=b.size();**

**//dp非递归版**

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

**c[i][0]=0;**

**for(j=0;j<=m;j++)**

**c[0][j]=0;**

**for(i=1;i<=n;i++)**

**for(j=1;j<=m;j++)**

**{**

**if(a[i-1]==b[j-1])**

**{**

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

**z[i][j]=0;**

**}**

**else**

**if(c[i][j-1]>c[i-1][j])**

**{**

**c[i][j]=c[i][j-1];**

**z[i][j]=1;**

**}**

**else**

**{**

**c[i][j]=c[i-1][j];**

**z[i][j]=-1;**

**}**

**}**

**cout<<c[n][m]<<endl;**

**// 逆向求最大公共子序列**

**string lsb;**

**i=n;j=m;**

**while(1)**

**{**

**if(i==0 || j==0)**

**break;**

**if(z[i][j]==0)**

**{**

**lsb+=a[i-1];**

**i--;j--;**

**}**

**if(z[i][j]==1)**

**j--;**

**if(z[i][j]==-1)**

**i--;**

**}**

**for(i=lsb.size()-1;i>=0;i--)**

**cout<<lsb[i];**

**cout<<endl;**

**/\***

**//dp递归版**

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

**for(j=0;j<=m;j++)**

**c[i][j]=lcs(i,j);**

**cout<<c[n][m]<<endl;**

**\*/**

**}**

**return 0;**

**}**

**线段树**

**#include<iostream>**

**using namespace std;**

**int ans; //记录不同颜色数**

**bool flag[33];//颜色是否已出现过**

**struct node**

**{**

**int l,r,color;**

**}a[300002];**

**void build(int i,int left,int right)//建树**

**{**

**a[i].color=1; //初始每个区间的颜色都为1**

**a[i].l=left;**

**a[i].r=right;**

**if(left==right)**

**return ;**

**int mid=(left+right)>>1;**

**build((i<<1),left,mid);**

**build((i<<1)+1,mid+1,right);**

**}**

**void paint(int i,int left,int right,int color) //给区间上色**

**{**

**if(left==a[i].l && a[i].r==right) //找到要上色区间则上色，返回。**

**{a[i].color=color;return;}**

**if(a[i].color!=color && a[i].color) //还没找到，若现在所在的这个区间是单色的，且和要上色的颜色不一样，**

**//则令它的两个子区间和该区间的颜色值一样。若不执行此操作，wa，不解。**

**{**

**a[(i<<1)].color=a[(i<<1)+1].color=a[i].color;**

**a[i].color=0;**

**}**

**if(color==a[i].color) //无论区间在哪个范围，只要当前所在区间是单色，且颜色和所要涂的颜色相同，则已涂好，返回。**

**return;**

**int mid=(a[i].l+a[i].r)>>1;**

**if(right<=mid)**

**paint((i<<1),left,right,color);**

**else**

**if(left>mid)**

**paint((i<<1)+1,left,right,color);**

**else**

**{**

**paint((i<<1),left,mid,color);**

**paint((i<<1)+1,mid+1,right,color);**

**}**

**}**

**void find(int i,int left,int right) //[left,right]区间内 查找不同颜色。**

**{**

**if(a[i].color) //如果当前区间是单色的则判断该颜色是否出现过，若未出现过则计数加1，返回，不再往下。**

**{**

**if(!flag[a[i].color])**

**{ ans++;flag[a[i].color]=true; }**

**return;**

**}**

**int mid=(a[i].l+a[i].r)>>1;**

**if(right<=mid)**

**find((i<<1),left,right);**

**else**

**if(left>mid)**

**find((i<<1)+1,left,right);**

**else**

**{**

**find((i<<1),left,mid);**

**find((i<<1)+1,mid+1,right);**

**}**

**}**

**int main()**

**{**

**int n,t,o,p,q,c;**

**char x;**

**scanf("%d%d%d",&n,&t,&o);**

**getchar();**

**build(1,1,n);**

**while(o--)**

**{**

**x=getchar();**

**if(x=='C')**

**{**

**scanf("%d%d%d",&p,&q,&c);**

**if(p>q)**

**{int tmp=p;p=q;q=tmp;}**

**paint(1,p,q,c);**

**}**

**if(x=='P')**

**{**

**scanf("%d%d",&p,&q);**

**if(p>q)**

**{int tmp=p;p=q;q=tmp;}**

**ans=0;**

**memset(flag,0,sizeof(flag));**

**find(1,p,q);**

**printf("%d\n",ans);**

**}**

**getchar();**

**}**

**return 0;**

**}**

**树状数组**

**#include<iostream>**

**#include<cstdio>**

**#include<string>**

**#include<cmath>**

**#include<vector>**

**#include<deque>**

**#include<cstdlib>**

**using namespace std;**

**int C[32003];**

**int level[15001];**

**int n;**

**int N=32001;**

**int Lowbit(int x)**

**{return x&-x;}**

**void Modify(int i,int x)**

**{**

**while(i<=N)**

**{**

**C[i]+=x;**

**i+=Lowbit(i);**

**}**

**}**

**int Sum(int i)**

**{**

**int sum=0;**

**while(i>0)**

**{**

**sum+=C[i];**

**i-=Lowbit(i);**

**}**

**return sum;**

**}**

**int main()**

**{**

**int i,x;**

**while(scanf("%d",&n)!=EOF)**

**{**

**for(i=1;i<=n;i++)**

**{**

**scanf("%d%\*d",&x);**

**x++;**

**Modify(x,1);**

**level[Sum(x)-1]++;**

**}**

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

**printf("%d\n",level[i]);**

**}**

**return 0;**

**}**

**多重背包**

**#include<stdio.h> //多重背包**

**#define N 202**

**int f[N],c[N],w[N],num[N],n,m;**

**void ZeroOnePack(int cost,int weight)**

**{**

**for(int j=m;j>=cost;j--)**

**if(f[j]<f[j-cost]+weight)**

**f[j]=f[j-cost]+weight;**

**}**

**void CompletePack(int cost,int weight)**

**{**

**for(int j=cost;j<=m;j++)**

**if(f[j]<f[j-cost]+weight)**

**f[j]=f[j-cost]+weight;**

**}**

**void MultiplePack(int cost,int weight,int number)**

**{**

**if(number==1)**

**{**

**ZeroOnePack(cost,weight);**

**return;**

**}**

**if(cost\*number>=m)**

**{**

**CompletePack(cost,weight);**

**return ;**

**}**

**int k=1;**

**while(k<number)**

**{**

**ZeroOnePack(cost\*k,weight\*k);**

**number-=k;**

**k<<=1;**

**}**

**ZeroOnePack(cost\*number,weight\*number);**

**}**

**int main()**

**{**

**int i,j,T;**

**scanf("%d",&T);**

**while(T--)**

**{**

**scanf("%d%d",&m,&n);**

**for(i=1;i<=n;i++)**

**scanf("%d%d%d",c+i,w+i,num+i);**

**for(i=0;i<=m;i++)**

**f[i]=0;**

**for(i=1;i<=n;i++)**

**MultiplePack(c[i],w[i],num[i]);**

**printf("%d\n",f[m]);**

**}**

**return 0;**

**}**

**最长递增子序列**

**#include<iostream>**

**using namespace std;**

**#define N 10002**

**int f[N],a[N];// f 最长递增子序列。**

**int main()**

**{**

**int i,j,n,max;**

**while(scanf("%d",&n)!=EOF)**

**{**

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

**{**

**scanf("%d",a+i);**

**f[i]=1;**

**}**

**max=1;**

**for(i=1;i<n;i++)**

**{**

**for(j=0;j<i;j++)**

**if(a[i]>a[j] && f[i]<f[j]+1)**

**{**

**f[i]=f[j]+1;**

**}**

**if(max<f[i])**

**max=f[i];**

**}**

**//for(i=0;i<n;i++)**

**// printf("%d ",f[i]);**

**//printf("\n");**

**printf("%d\n",max);**

**}**

**return 0;**

**}**