**1002大数相加**

#include <stdio.h>

#include <string.h>

#define M 1000

void ADD(char str1[],char str2[])

{

int num1[M]={0};

int num2[M]={0};

int num3[M]={0};

int i,j;

for(i=strlen(str1)-1,j=0;i>=0;i--)

{

num1[j++]=str1[i]-'0';

}

for(i=strlen(str2)-1,j=0;i>=0;i--)

{

num2[j++]=str2[i]-'0';

}

for(i=0;i<M;i++)

{

num3[i]+=num1[i]+num2[i];

if(num3[i]>9)

{

num3[i+1]=num3[i]/10;

num3[i]=num3[i]%10;

//num3[i]-=10;

//num3[i+1]++;

}

}

for(i=M-1;num3[i]==0 && i>=0;i--);

printf("%s + %s = ",str1,str2);

if(i>=0)

{

i++;

while(i--)

{

printf("%d",num3[i]);

}

}

else

printf("0");

}

int main()

{

int T,k=1;

char str1[M]={0};

char str2[M]={0};

scanf("%d",&T);

if(T>=1 && T<=20)

{

while(T--)

{

scanf("%s%s",str1,str2);

printf("Case %d:\n",k++);

ADD(str1,str2);

if(T>0)

{

printf("\n\n");

}

else

printf("\n");

}

}

}

**1042阶乘**

#include <stdio.h>

#include <string.h>

#define M 100000

int main()

{

int n,i,j,temp,c,count;

int a[M];

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

{

c=0;

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

a[0]=1;

count=1;

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

{

for(j=0;j<count;j++)

{

temp=i\*a[j]+c;

a[j]=temp%10;

c=temp/10;

}

while(c)

{

a[count++]=c%10;

c/=10;

}

}

j=M-1;

while(a[j]==0)

{

j--;

}

while(j>=0)

{

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

}

printf("\n");

}

return 0;

}

**1124求N！末尾的零**

#include <stdio.h>

int main()

{

int T,n,count;

scanf("%d",&T);

while(T--)

{

count=0;

scanf("%d",&n);

while(n)

{

count+=n/5;

n/=5;

}

printf("%d\n",count);

}

return 0;

}

**1715大菲波数**

#include <stdio.h>

#include <string.h>

int a[1005][1010];

int main()

{

int i,j,s,c,n,m;

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

a[1][0]=1;

a[2][0]=1;

for(i=3;i<1005;i++)

{

c=0;

for(j=0;j<=1000;j++)

{

s=a[i-2][j]+a[i-1][j]+c;

a[i][j]=s%10;

c=s/10;

}

}

scanf("%d",&n);

while(n--)

{

scanf("%d",&m);

for(i=1010;a[m][i]==0 && i>=0;i--);

i++;

while(i--)

{

printf("%d\n",a[m][i]);

}

}

return 0;

}

**2058等差数列**

#include <stdio.h>

#include <string.h>

int a[1005][1010];

int main()

{

int i,j,s,c,n,m;

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

a[1][0]=1;

a[2][0]=1;

for(i=3;i<1005;i++)

{

c=0;

for(j=0;j<=1000;j++)

{

s=a[i-2][j]+a[i-1][j]+c;

a[i][j]=s%10;

c=s/10;

}

}

scanf("%d",&n);

while(n--)

{

scanf("%d",&m);

for(i=1010;a[m][i]==0 && i>=0;i--);

i++;

while(i--)

{

printf("%d\n",a[m][i]);

}

}

return 0;

}

**错排公式+递推 2048**

/\*三种思路：

1. N张字条的所有可能排列自然是N!（分母）。

现在的问题就是求N张字条的错排数f(N)（分子）。

首先我们考虑，如果前面N-1个人拿的都不是自己的票，即前N-1个人满足错排，现在又来了一个人，他手里拿的是自己的票。只要他把自己的票与其他N-1个人中的任意一个交换，就可以满足N个人的错排。这时有(N-1)\*f(N-1)种方法。

Besides，我们考虑，如果前N-1个人不满足错排，而第N个人把自己的票与其中一个人交换后恰好满足错排。

这种情况发生在原先N-1人中，N-2个人满足错排，有且仅有一个人拿的是自己的票，而第N个人恰好与他做了交换，这时候就满足了错排。

又因为前N-1个人中，每个人都有机会拿着自己的票。所以有N-1种交换的可能。故这时有(N-1)\*f(N-2)种方法。

综上所述：f(N)=(N-1)\*[f(N-1)+f(N-2)]

2. 另一种推导思路是先通过容斥定理直接计算出错排数f(n)=n!(1/2!-1/3!+…..+((-1)^n)/n!)

所以f(n)/n!-f(n-1)/(n-1)!=((-1)^n)/n!

两边同时乘上n!有f(n)-n\*f(n-1)=(-1)^n

所以f(n-1)-(n-1)\*f(n-2)=(-1)^(n-1)=(-1)\*[f(n)-n\*f(n-1)]

化简得f(n)=(n-1)\*[f(n-1)+f(n-2)] \*/

#include <stdio.h>

#define M 25

int main()

{

int c,n,i;

double factorial;

\_int64 a[M];

a[1]=0;

a[2]=1;

for(i=3;i<M;i++)

{

a[i]=(i-1)\*(a[i-1]+a[i-2]);

}

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

{

while(c--)

{

scanf("%d",&n);

factorial=1.0;

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

factorial\*=i;

printf("%.2lf%%\n",a[n]/factorial\*100);

}

}

return 0;

}

递推错排2049

#include <stdio.h>

#define M 25

\_int64 cal(int n,int m)

{

int i;

\_int64 s1,s2,s3;

s1=s2=s3=1;

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

s1\*=i;

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

s2\*=i;

for(i=1;i<=(n-m);i++)

s3\*=i;

return s1/s2/s3;

}

int main()

{

int n,m,c;

int i;

\_int64 a[M],k;

a[1]=0;

a[2]=1;

for(i=3;i<M;i++)

{

a[i]=(i-1)\*(a[i-1]+a[i-2]);

}

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

{

while(c--)

{

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

k=cal(n,m);

printf("%I64d\n",k\*a[m]);

}

}

return 0;

}

**递归2041**

/\*

当你到达第n阶的时候有两种到达方式。 在n-1处上 1个楼梯。在n-2处上2个楼梯

所以上N阶楼梯的情况总数=上n-1的总数+上n-2的总数

这样递推公式就出来了

f（n）=f（n-1）+f（n-2）

\*/

#include <stdio.h>

int main()

{

int n,a[45],m,i;

a[1]=1;

a[2]=1;

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

{

for(i=3;i<=45;i++)

a[i]=a[i-1]+a[i-2];

while(n--)

{

scanf("%d",&m);

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

}

}

return 0;

}

**递推2044**

#include <stdio.h>

#define M 55

int main()

{

int n,a,b,i;

\_int64 f[M];

f[1]=1;

f[2]=1;

for(i=3;i<M;i++)

f[i]=f[i-1]+f[i-2];

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

{

while(n--)

{

scanf("%d%d",&a,&b);

printf("%I64d\n",f[b-a+1]);

}

}

return 0;

}

//斐波那契数列的第47项就已超出int型的范围，第50项开始就为11位数，故int型无法满足，而VC中无法识别long long类型，故用\_int64类型

**递推2045**

#include <stdio.h>

#define M 55

int main()

{

int i,n;

\_int64 a[M];

a[1]=3;

a[2]=6;

a[3]=6;

for(i=4;i<M;i++)

a[i]=a[i-1]+a[i-2]\*2;

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

{

printf("%I64d\n",a[n]);

}

return 0;

}

由此类推到有n个格子，假设前n-1个格子有f(n-1)种情况，当第n-1个格子和第一个格子不同时，

则 第n个格子只有一种选择，所以一共有f(n-1)\*1种情况；当第n-2个格子和第一个不同，第n-1个格子和第一个格子相同时，

那么第 n 个格子可以有两种选择，这样一来，共有f(n-2)\*2种情况；把这两种情况加起来一共有f(n-1)\*1+f(n-2)\*2

//每个基本元素都是满足题目中的条件，即第一个格子和最后一个格子颜色不同

**递推2046**

#include <stdio.h>

#define M 55

int main()

{

int n,i;

\_int64 a[M];

a[1]=1;

a[2]=2;

for(i=3;i<M;i++)

a[i]=a[i-1]+a[i-2];

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

{

printf("%I64d\n",a[n]);

}

}

//第n个，即2\*n时，可由第n-1个的竖直排列再加一个，和第n-2个中横着排两个（因为竖着的一个、横着的两个是最基本元素，当n>=3时，所有情况都由这两种组成）

**动态规划2059**

/\*

用dp[i]表示 到第i个充电站的最短时间；

这里我们先求到第3个充电站用的最短时间来分析：

1.在第二个充电站时 ， 有充电或不充电 这两种方式到达 第3个充电站

两个充电站的距离 len=p[3]-p[2];

不充电用时 T1=len/VT2;

充电用时 --有两种情况

C<len : T2=C/VT1+(len-C)/VT2;

C>-len : T2=Len/VT2;

所以 dp[3]=dp[2]+min{T1,T2};

那么这就是到达第3个充电站的最短时间吗？不一定，因为有可能从第一个充电站直接到第三个充电站的时间更少

2.故我们还要比较从dp[1]到dp[3] 的情况；从dp[2]到dp[3] 的情况， 从这三者中选出时间最少的。

综上分析，我们求第i个充电站的最短时间，就要把从第0个到第i-1个充电站全部扫描一遍，每一趟都包含上述两种情况，再从这i-1趟中

选择最小的。这也是下面代码中为什么用了两个for循环。

\*/

#include <stdio.h>

int L;

int N,C,T;

int VR,VT1,VT2;

int p[110];

double dp[110];

double min(double a,double b)

{

return a>b?b:a;

}

int main()

{

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

{

scanf("%d%d%d",&N,&C,&T);

scanf("%d%d%d",&VR,&VT1,&VT2);

for(int i=1;i<=N;i++)

{

scanf("%d",p+i); //p1,p2.....pn分别表示各个充电站离跑道起点的距离

}

p[0]=0;

p[N+1]=L;

for(int i=0;i<=N+1;i++)

{

dp[i]=0xfffff;

}

dp[0]=0.0;

for(int i=1;i<=N+1;i++)

{

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

{

int len=p[i]-p[j];

double time2=1.0\*len/VT2; //从第j个充电站到第i个充电站 选择 用脚瞪

double time1;

if(len > C) //从第j个充电站到第i个充电站 选择 充电--则有两种情况，从中选小。

time1=1.0\*C/VT1 + 1.0\*(len-C)/VT2 + T;

else

time1=1.0\*len/VT1 + T;

if(j==0)

time1-=T; //第一次出发不需要充电

double time=min(time1,time2);

dp[i]=min(dp[i],dp[j]+time);

}

}

double rab\_t=1.0\*L/VR;

if(rab\_t>dp[N+1])

{

printf("What a pity rabbit!\n");

}

else

{

printf("Good job,rabbit!\n");

}

}

return 0;

}

**多边形面积**

#include <stdio.h>

int main()

{

int n,x[3],y[3];

double s;

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

{

s=0;

scanf("%d%d",&x[0],&y[0]);

x[2]=x[0];

y[2]=y[0];

while(--n)

{

scanf("%d%d",&x[1],&y[1]);

s+=x[0]\*y[1]-x[1]\*y[0];

x[0]=x[1];

y[0]=y[1];

}

s+=x[0]\*y[2]-x[2]\*y[0];

printf("%.1lf\n",s/2);

}

return 0;

}

**进制转换**

#include <stdio.h>

#include <math.h>

#define M 1000

int main()

{

int n,r,a[M],i,m;

int b[]={'A','B','C','D','E','F'};

while(scanf("%d%d",&m,&r) != EOF)

{

i=0;

n=m;

while(n)

{

a[i++]=abs(n%r);

n=n/r;

}

if(m<0)

printf("-");

while(i--)

{

if(abs(a[i]) >= 10)

printf("%c",b[a[i]-10]);

else

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

}

if(m==0)

printf("0");

printf("\n");

}

return 0;

}

**两个长方形相交求重叠面积2056**

#include <stdio.h>

void swap(double \*a,double \*b)

{

double t;

t=\*a;

\*a=\*b;

\*b=t;

}

int main()

{

double x1,y1,x2,y2,x3,y3,x4,y4;

double a,c,b,d; //重叠部分的长方形的左下方和右上方坐标（a,c）(b,d)

while(scanf("%lf%lf%lf%lf%lf%lf%lf%lf",&x1,&y1,&x2,&y2,&x3,&y3,&x4,&y4) != EOF)

{

if(x1>x2)

swap(&x1,&x2);

if(y1>y2)

swap(&y1,&y2);

if(x3>x4)

swap(&x3,&x4);

if(y3>y4)

swap(&y3,&y4); //将长方形的对角线坐标统一为：左下方和右上方

if(x3<x2 && y3<y2 && x1<x4 && y1<y4) //两个长方形相交（或者包含）的条件： 第3个点在第2个点的左下方；第1个点在第4个点的左下方 可画图先将一个长方形固定，然后将另一个长方形旋转，即可发现规律

{

a=(x1>x3)?x1:x3;

c=(y1>y3)?y1:y3; //观察图形可知，（a,c）这个坐标是在第1个点和第3个点中取横纵坐标大的值

b=(x2<x4)?x2:x4;

d=(y2<y4)?y2:y4; //观察图形可知，（b,d）这个坐标是在第2个点和第4个点中取横纵坐标大的值

printf("%.2lf\n",(b-a)\*(d-c));

}

else

printf("0.00\n");

}

return 0;

}

**去掉前导0、后导0方法（strchr）2054**

#include <stdio.h>

#include <string.h>

#define M 100000

char a[M],b[M];

void ridlastz(char \*s)

{

int len=strlen(s);

char \*p=s+len-1;

if(strchr(s,'.'))

{

while(\*p=='0')

{

\*p--=NULL;

}

}

if(\*p=='.')

\*p=NULL;

}

int main()

{

while(scanf("%s%s",a,b) !=EOF)

{

char \*pa=a;

char \*pb=b;

while(\*pa=='0')

pa++;

while(\*pb=='0')

pb++;

ridlastz(pa);

ridlastz(pb);

if(strcmp(pa,pb)==0)

printf("YES\n");

else

printf("NO\n");

}

return 0;

}

**十六进制数的A+B**

#include <stdio.h>

int main()

{

\_int64 a,b;

while(scanf("%I64x%I64x",&a,&b) != EOF) //由于a,b值可能会很大，用\_int64类型，输入输出有十六进制的格式，是无符号数

{

if(a+b>=0)

printf("%I64X\n",a+b);

else

printf("-%I64X\n",-(a+b)); //X为大写时，表示输出的十六进制数用大写字母表示。

}

return 0;

}

**贪心算法2021**

#include <stdio.h>

int main()

{

int wage,i,temp,n,sum,count;

int a[6]={100,50,10,5,2,1};

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

{

sum=0;

while(n--)

{

scanf("%d",&wage);

for(i=0;i<6;i++)

{

count=wage/a[i];

temp=wage%a[i];

sum+=count;

if(temp==0)

break;

wage=temp;

}

}

printf("%d\n",sum);

}

return 0;

}

#include <stdio.h>

#include <stdlib.h>

#define M 105

typedef struct

{

int s;

int e;

}time;

time t[M];

int cmp(const void \*a,const void \*b)

{

return (\*(time \*)a).e-(\*(time \*)b).e;

}

**贪心算法2037**

int main()

{

int i,end,count,n;

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

{

count=0;

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

scanf("%d%d",&t[i].s,&t[i].e);

qsort(t,n,sizeof(t[0]),cmp);

count++;

end=t[0].e;

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

{

if(t[i].s >= end)

{

count++;

end=t[i].e;

}

}

printf("%d\n",count);

}

return 0;

}

**折线分平面2050**

#include <stdio.h>

#define M 10005

int main()

{

int c,n,i;

\_int64 a[M];

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

{

a[i]=2\*i\*i-i+1;

}

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

{

while(c--)

{

scanf("%d",&n);

printf("%I64d\n",a[n]);

}

}

return 0;

}

**直线、折线、封闭曲线、平面分割空间问题**

(1) n条直线最多分平面问题

题目大致如:n条直线，最多可以把平面分为多少个区域。

析:可能你以前就见过这题目，这充其量是一道初中的思考题。但一个类型的题目还是从简单的入手，才容易发现规律。

当有n-1条直线时，平面最多被分成了f（n-1）个区域。则第n条直线要是切成的区域数最多，就必须与每条直线相交且不能有同一交点。

这样就会得到n-1个交点。这些交点将第n条直线分为2条射线和n-2条线断。而每条射线和线断将以有的区域一分为二。

这样就多出了2+（n-2）个区域。

故：f(n)=f(n-1)+n

=f(n-2)+(n-1)+n

……

=f(1)+1+2+……+n

=n(n+1)/2+1

(2) 折线分平面（hdu2050）

根据直线分平面可知，由交点决定了射线和线段的条数，进而决定了新增的区域数。当n-1条折线时，区域数为f（n-1）。

为了使增加的区域最多，则折线的两边的线段要和n-1条折线的边，即2\*（n-1）条线段相交。那么新增的线段数为4\*（n-1），

射线数为2。但要注意的是，折线本身相邻的两线段只能增加一个区域。

故：f(n)=f(n-1)+4(n-1)+2-1

=f(n-1)+4(n-1)+1

=f(n-2)+4(n-2)+4(n-1)+2

……

=f(1)+4+4\*2+……+4(n-1)+(n-1)

=2n^2-n+1

// 或者可以这样思考：

n条直线能将平面分割成多少块。答案是1+（1+n）\*n/2.

我们可以把n条折线看成2\*n条直线，只不过每2条直线和一条折线相比都多分割了2个平面。

（可以将折线看为两条交叉直线去掉顶点外的另一部分，画图可知变为折线后少了2个平面）

所以不仅把前一个公示的n变成2n，还要再减去2n，所以公式成了1+（1+2n）\*2n/2-2n.

化简后，为2n^2-n-1.

(3) 封闭曲线分平面问题

题目大致如设有n条封闭曲线画在平面上，而任何两条封闭曲线恰好相交于两点，且任何三条封闭曲线不相交于同一点，

问这些封闭曲线把平面分割成的区域个数。

析：当n-1个圆时，区域数为f(n-1).那么第n个圆就必须与前n-1个圆相交，则第n个圆被分为2（n-1）段线段，

增加了2（n-1）个区域。

故： f(n)=f(n-1)+2(n-1)

=f(1)+2+4+……+2(n-1)

=n^2-n+2

(4)平面分割空间问题（hdu1290）

由二维的分割问题可知，平面分割与线之间的交点有关，即交点决定射线和线段的条数，从而决定新增的区域数。

试想在三维中则是否与平面的交线有关呢？当有n-1个平面时，分割的空间数为f（n-1）。

要有最多的空间数，则第n个平面需与前n-1个平面相交，且不能有共同的交线。即最多有n-1 条交线。

而这n-1条交线把第n个平面最多分割成g（n-1）个区域。（g（n）为（1）中的直线分平面的个数）此平面将原有的空间一分为二，

则最多增加g（n-1）个空间。

故：f=f(n-1)+g(n-1) ps:g(n)=n(n+1)/2+1

=f(n-2)+g(n-2)+g(n-1)

……

=f(1)+g(1)+g(2)+……+g(n-1)

=2+(1\*2+2\*3+3\*4+……+(n-1)n)/2+（n-1）

=(1+2^2+3^2+4^2+……+n^2-1-2-3-……-n )/2+n+1

=(n^3+5n)/6+1

**最小公倍数（最大公约数）**

#include <stdio.h>

/\*最小公倍数=两整数的乘积/最大公约数

最大公约数算法：辗转相除法

1. a%b得余数c

2. 若c=0,则b即为两数的最大公约数

3. 若c!=0,则a=b,b=c,再回去执行1

\*/

int main()

{

int a,b,c,m,n;

scanf("%d%d",&a,&b);

m=a;

n=b;

while(b)

{

c=a%b;

a=b;

b=c;

}

printf("最大公约数为：%d\n",a);

printf("最大公倍数为：%d\n",m\*n/a);

}

//穷举法求最小公倍数 设a>b

for(i=a;;i++)

{

if(i%a==0 && i%b==0)

break;

printf("最大公倍数为：%d\n",i);

}