大跟堆

```
long dn=0;                    /*大跟堆*/
void jia(long key)
{ long i,j,k,m;
   a[++dn]=key;
   i=dn/2; j=dn;
   while(i>=1)
   { if(a[j]>a[i]) swap(&a[j],&a[i]);
      else break;
      j=i; i/=2;
   }
}
void del()
{ long i,j,k,m;
   swap(&a[1],&a[dn]);
   dn--;
   i=1;
   while(i<=dn/2)
   { j=i*2;
      if(j+1<=dn&&a[j]<a[j+1]) j++;
      if(a[i]<a[j]) swap(&a[i],&a[j]);
      else break;
      i=j;
   }
}
```

高精度

高乘高

```
void gaochen()
{ long i,j,k,m,n;
   for(i=1;i<=a[0];i++)
     for(j=1;j<=b[0];j++)
     { ans[i+j-1]+=a[i]*b[j];
        if(ans[i+j-1]>=10)
        { ans[i+j]+=ans[i+j-1]/10;
           ans[i+j-1]%=10;
        }
     }
   i=1;
   while(i<=a[0]+b[0]-1||ans[i]>0)
     i++;
   ans[0]=i-1;
```

高减高

```
void gaojian(long a[],long b[])
{ long i,j,k,h;
   for(i=1;i<=b[0];i++)
     ans[i]=a[i]-b[i];
   for(;i<=a[0];i++)
     ans[i]=a[i];
   ans[0]=a[0];
   i=1;
   while(i<=ans[0])
   { if(ans[i]<0)
      { ans[i+1]--;
         ans[i]+=10;
      }
      i++;
   }
   while(ans[0]>=1&&ans[ans[0]]==0) ans[0]--;
   if(ans[0]==0) ans[0]=1;
}
long pan(long a[],long b[])
{ long i;
   if(a[0]>b[0]) return 1;
   else if(a[0]<b[0]) return 0;
   else
   { for(i=a[0];i>=1;i--)
        if(a[i]>b[i]) return 1;
        else if(a[i]<b[i]) return 0;
   }
   return 0;
}
int main()
{ long i,j,k;
   init();
   if(pan(a,b)==1)
     gaojian(a,b);
   else
   { fprintf(output,"-");
      gaojian(b,a);
   }
   print();
```

高除单

```
 /*读入时，按顺序读入。/
void chu()
{ long i,j,k,g;
   g=0;
   for(i=1;i<=a[0];i++)
   { g=g*10+a[i];
      ans[a[0]-i+1]=g/b;
      g=g%b;
   }
   ans[0]=a[0]; i=1;
   while(i<=ans[0]&&ans[ans[0]]==0) ans[0]--;
   if(ans[0]==0) ans[0]=1;
}
```

Rmq

```
long a[41]={0},n;
long f[10000][100];
void init()
{ long i,j,k,m,h;
   h=log(n)/log(2);
   for(i=1;i<=n;i++)
     f[i][0]=a[i];
   for(j=1;j<=h;j++)
     for(i=1;i<=n-(1<<j)+1;i++)
       if(f[i][j-1]>f[i+(1<<(j-1))][j-1])    f[i][j]=f[i][j-1];
       else f[i][j]=f[i+(1<<(j-1))][j-1];
}
long find(long a,long b)
{ long i,j,k,m;
   m=log(b-a+1)/log(2);
   if(f[a][m]>f[b-(1<<m)+1][m]) return f[a][m];
   else return f[b-(1<<m)+1][m];
}
```

最长公共上升序列

```
for(i=1;i<=n;i++)
{ max=0;
   for(j=1;j<=n;j++)[1]
   { f[i][j]=f[i-1][j];
      if(a[i]>b[j]&&f[i-1][j]>f[i-1][max]) max=j;
      if(a[i]==b[j]&&f[i][j]<f[i-1][max]+1)
      { f[i][j]=f[i-1][max]+1;
         if(maxn<f[i][j])
           maxn=f[i][j];
      }
   }
```

线段树（矩形）

```
#include"stdio.h"
#define NEWS (tree *)malloc(sizeof(tree))
typedef struct tree
{ long cx,cy,cc,ccx,ccy;
   struct tree *l1,*l2,*l3,*l4;
}tree;
long lx[1001],ly[1001],llx[1001],lly[1001],llc[1001];
tree *t;
long a[3001]={0};
long n,m,nx,ny;
long min(long a,long b)
{ if(a<b) return a;
   else       return b;
}
long max(long a,long b)
{ if(a>b) return a;
   else       return b;
}
tree *jia(tree *p,long a1,long b1,long a2,long b2,long x1,long y1,long x2,long y2,long c)
{ long mid1,mid2,i,j,r=0,w=0;
   if(x1<a1||y1<b1||x2>a2||y2>b2||x2<x1||y2<y1) return p;
   if(p==NULL)
   { p=NEWS;
      p->cc=-1;
      p->cx=a1;    p->cy=b1;
      p->ccx=a2; p->ccy=b2;
      p->l1=p->l2=p->l3=p->l4=NULL;
   }
   mid1=(a1+a2)/2;
   mid2=(b1+b2)/2;
   if(p->cc!=-1) return p;

if(a1==x1&&y1==b1&&x2==a2&&y2==b2&&p->l1==NULL&&p->l2==NULL&&p->l3==NU
LL&&p->l4==NULL) p->cc=c;
   else

{  p->l1=jia(p->l1,a1       ,b1       ,mid1,mid2,max(a1,x1),max(b1,y1),min(mid1,x2),min(mid2,y2)
,c);

p->l2=jia(p->l2,mid1+1,b1       ,a2    ,mid2,max(mid1+1,x1),max(b1,y1),min(a2,x2),min(mid2,y2
),c);
```

```
p->l3=jia(p->l3,a1        ,mid2+1,mid1,b2    ,max(a1,x1),max(y1,mid2+1),min(mid1,x2),min(b2,y2
),c);

p->l4=jia(p->l4,mid1+1,mid2+1,a2    ,b2    ,max(mid1+1,x1),max(mid2+1,y1),min(a2,x2),min(b2,
y2),c);
   }
   return p;
}
void fun(tree *p)
{
   if(p->cc==-1)
   { if(p->l1!=NULL) fun(p->l1);
      if(p->l2!=NULL) fun(p->l2);
      if(p->l3!=NULL) fun(p->l3);
      if(p->l4!=NULL) fun(p->l4);
   }
   else a[p->cc]+=(p->ccx-p->cx+1)*(p->ccy-p->cy+1);
}
int main()
{ FILE *input,*output;
   long i,j,k,r,w;
   input=fopen("rect1.in","r");
   output=fopen("rect1.out","w");
   fscanf(input,"%ld %ld %ld",&nx,&ny,&n);
   for(i=1;i<=n;i++)
   { fscanf(input,"%ld%ld%ld%ld%ld",&lx[i],&ly[i],&llx[i],&lly[i],&llc[i]);
      lx[i]++; ly[i]++;
   }
   t=NULL;
   for(i=n;i>=1;i--)
   { t=jia(t,1,1,nx,ny,lx[i],ly[i],llx[i],lly[i],llc[i]);
   }


   fun(t);
   a[1]=nx*ny;
   for(i=2;i<=2500;i++) a[1]-=a[i];
   for(i=1;i<=2500;i++)
    if(a[i]!=0)
      fprintf(output,"%ld %ld\n",i,a[i]);
   fclose(input);
   fclose(output);
   return 0;
}
```

伸展树

```
/* 左小右大，自顶向下 */
#include"stdio.h"
#define NEWS (splaytree *)malloc(sizeof(splaytree))
typedef struct splaytree
{ struct splaytree *lc,*rc;
   long data;
}splaytree;
splaytree *nullnode;
FILE *input,*output;
splaytree *singleft(splaytree *t)
{ splaytree *p;
   p=t->lc;
   t->lc=p->rc;
   p->rc=t;
   return p;
}
splaytree *singright(splaytree *t)
{ splaytree *p;
   p=t->rc;
   t->rc=p->lc;
   p->lc=t;
   return p;
}
splaytree *splay(splaytree *t,long key)
{ splaytree head;
   splaytree *lcmax,*rcmin;
   head.lc=head.rc=nullnode;
   lcmax=rcmin=&head;
   nullnode->data=key;

   while(t->data!=key)
   { if(key<t->data)
      { if(key<t->lc->data) t=singleft(t);
         if(t->lc==nullnode) break;
         rcmin->lc=t;
         rcmin=t;
         t=t->lc;
      }
      if(key>t->data)
      { if(key>t->rc->data) t=singright(t);
```

```
            if(t->rc==nullnode) break;
            lcmax->rc=t;
            lcmax=t;
            t=t->rc;
        }
    }
    lcmax->rc=t->lc;
    rcmin->lc=t->rc;
    t->lc=head.rc;
    t->rc=head.lc;
    return t;
}
splaytree *del(splaytree *t,long key)
{ splaytree *newtree;
    if(t!=nullnode)
    {
        t=splay(t,key);
        if(t->lc==nullnode) newtree=t->rc;
        else
        { newtree=t->lc;
            newtree=splay(newtree,key);
            newtree->rc=t->rc;
        }
        free(t);
        t=newtree;
    }
    return t;
}
splaytree *insert(splaytree *t,long key)
{ splaytree *newnode;
    newnode=NEWS;
    newnode->lc=newnode->rc=nullnode;
    newnode->data=key;
    if(t==nullnode) t=newnode;
    else
    { t=splay(t,key);
        if(key<t->data)
        { newnode->rc=t;
            newnode->lc=t->lc;
            t->lc=nullnode;
            t=newnode;
        }
        else
        { newnode->lc=t;
```

```
            newnode->rc=t->rc;
            t->rc=nullnode;
            t=newnode;
        }
    }
    newnode=NULL;
    return t;
}
int main()
{ splaytree *t;
    long i,j,k,r,w,n,m;
    input=fopen("tree.in","r");
    output=fopen("tree.out","w");
    nullnode=NEWS;
    nullnode->lc=nullnode->rc=nullnode;
    t=nullnode;
    fscanf(input,"%ld",&n);
    for(i=1;i<=n;i++)
    { fscanf(input,"%ld",&k);
       t=insert(t,k);
    }
    fscanf(input,"%ld",&m);
    for(i=1;i<=m;i++)
    { fscanf(input,"%ld",&k);
       del(t,k);
    }
    fclose(input);
    fclose(output);
    return 0;
}
```

平衡树

```
/* 小的在左，大的在右。 */
#include"stdio.h"
#define NEWS (avltree *)malloc(sizeof(avltree))
typedef struct avltree
{ struct avltree *rc,*lc;
   long height,data,h,gao;
}avltree;
   FILE *input,*output;
long max(long a,long b) { if(a>b) return a; else return b;}
long min(long a,long b) { if(a<b) return a; else return b;}
long mheight(avltree *t){ if(t==NULL) return 0; else return t->height;    }
avltree *singleft(avltree *t)
{ avltree *a;
   a=t->lc;
   t->lc=a->rc;
   a->rc=t;

   t->height=max(mheight(t->lc),mheight(t->rc))+1;
   a->height=max(mheight(a->lc),mheight(a->rc))+1;
   return a;
}
avltree *singright(avltree *t)
{ avltree *p;
   p=t->rc;
   t->rc=p->lc;
   p->lc=t;

   t->height=max(mheight(t->lc),mheight(t->rc))+1;
   p->height=max(mheight(p->lc),mheight(p->rc))+1;
   return p;
}
avltree *douleft(avltree *t)
{ t->lc=singright(t->lc);
   t=singleft(t);
   return t;
}
avltree *douright(avltree *t)
{ t->rc=singleft(t->rc);
   t=singright(t);
   return t;
```

```
}
avltree *insert(avltree *t,long key)
{ long i,j,k,r,w;
   avltree *p;
   if(t==NULL)
   { p=NEWS;
      p->height=1;
      p->data=key;
      p->lc=p->rc=NULL;
      return p;
   }
   if(key>t->data)
   { t->rc=insert(t->rc,key);
      if(mheight(t->rc) - mheight(t->lc) ==2)
      { if(key>t->rc->data) t=singright(t);
         else t=douright(t);
      }
   }
   else if(key<t->data)
   { t->lc=insert(t->lc,key);
      if(mheight(t->lc) - mheight(t->rc) ==2)
      { if(key<t->lc->data) t=singleft(t);
         else t=douleft(t);
      }
   }
   t->height=max(mheight(t->lc),mheight(t->rc))+1;
   return t;
}

int main()
{ long i,j,k,r,w,n;
   avltree *t=NULL;
   FILE *input,*output;
   input=fopen("avl.in","r");
   output=fopen("avl.out","w");
   fscanf(input,"%ld",&n);
   for(i=1;i<=n;i++)
   { fscanf(input,"%ld",&r);
      t=insert(t,r);
   }
   fclose(input);
   fclose(output);
   return 0;
}
```

最小费用流（SPFA）

```
void jin(long k)
{ list[++rear]=k; rear%=MAXN; }
void chu(long *k)
{ *k=list[++front]; front%=MAXN; }
void zdl_spfa()
{ long i,j,k,r,w;
   node *p;
   do
   { for(i=s;i<=t;i++) dis[i] = MAXNUM;
      memset(fa,0,sizeof(fa));
      dis[s] = 0;
      rear = front = 0;
      jin(s); v[s] = 1;
      while(rear!=front)
      { chu(&i); v[i] = 0;
         for(p=g[i];p;p=p->next)
          if(dis[p->adj] > p->cost + dis[i] && p->lost > 0)
          { dis[p->adj] = p->cost + dis[i];
             ft[p->adj] = p;
             fa[p->adj] = i;
             if(v[p->adj] == 0) { jin(p->adj); v[p->adj] = 1; }
          }
      }
      if(dis[t] != MAXNUM)
      { w = MAXNUM;
         for(i=t;i!=s;i=fa[i])
          w = min(w,ft[i]->lost);
         for(i=t;i!=s;i=fa[i])
         { ans += ft[i]->cost*w;
            ft[i]->lost -= w;
            ft[i]->op->lost += w;

         }
      }
   }while(dis[t] != MAXNUM);
}
```

DINIC

```
int lv[maxn], zh[maxn], st[maxn], zhan[maxn];

int re, fr, top;
void jin(int w){
	zh[++top] = w;
}
bool level(){
	memset(lv, -1, sizeof(lv));
	re = fr = 0;
	zh[++re] = s;
	lv[s] = 0;
	while(re != fr){
		int k = zh[++fr];
		for(int p = g[k]; p; p = e[p].ne)
			if(e[p].fow != 0 && lv[ e[p].ad ] == -1){
				lv[ e[p].ad ] = lv[k] + 1;
				zh[++re] = e[p].ad;
				if(e[p].ad == t)
					return true;
			}
	}
	return false;
}
void dinic(){
	top = 0;
	jin(s);
	for(int i = 0; i <= t; i++) st[i] = g[i];
	while(top != 0){
		int k = zh[top];
		if(k != t){
			for(; st[k]; st[k] = e[ st[k] ].ne)
				if(lv[ e[ st[k] ].ad ] == lv[k] + 1 && e[ st[k] ].fow != 0)
					break;
			if(st[k] == 0) { top--; lv[k] = -1;}
			else{
				jin( e[ st[k] ].ad);
				zhan[top] = st[k];
			}
		}
		else{
```

```
                int w = maxint;
                for(int i = top; i >= 2; i--) w = min(w, e[ zhan[i] ].fow);
                for(int i = top; i >= 2; i--){
                    e[ zhan[i] ].fow -= w;
                    e[ zhan[i] ^ 1].fow += w;
                    if( e[ zhan[i] ].fow == 0) top = i - 1;
                }
                ans += w;
            }
        }
}
void gao(){
    ans = 0;
    while(level())
        dinic();
}
```

tarjan

```
tarjan(u)
{
    DFN[u]=Low[u]=++Index                               // 为节点 u 设定次序编号和 Low 初值
        Stack.push(u)                                   // 将节点 u 压入栈中
        for each (u, v) in E                            // 枚举每一条边
            if (v is not visted)                        // 如果节点 v 未被访问过
                tarjan(v)                               // 继续向下找
                Low[u] = min(Low[u], Low[v])
            else if (v in S)                            // 如果节点 u 还在栈内
                Low[u] = min(Low[u], DFN[v])
        if (DFN[u] == Low[u])                           // 如果节点 u 是强连通分量的根
            repeat
                v = S.pop                               // 将 v 退栈，为该强连通分量中一个顶点
                print v
            until (u== v)
}
```

凸包

```
    bool operator < (const point &p) const{
        if(sgn(x - p.x) != 0) return x < p.x;
        else return y < p.y;
    }
void convex(vector <point> a, vector <point> &tu){ //顺时针
    point hu[maxn], hd[maxn];
    int n = a.size(), un, dn;
    sort(a.begin(), a.end());
    hu[0] = hd[0] = a[0];
    hu[1] = hd[1] = a[1];
    un = dn = 1;
    for(int i = 2; i < n; i++){
        for(; un > 0 && sgn( (hu[un] - hu[un - 1]) * (a[i] - hu[un] )) >=
0; un--);
        for(; dn > 0 && sgn( (hd[dn] - hd[dn - 1]) * (a[i] - hd[dn] ))
<= 0; dn--);
        hu[++un] = a[i];
        hd[++dn] = a[i];
    }
    tu.clear();
    for(int i = 0; i <= un - 1; i++) tu.push_back(hu[i]);
    for(int i = dn; i >= 1; i--) tu.push_back(hd[i]);
}
```

判线段相交，求交点

```
bool jiaodian(point a,point b,point c,point d,point &e)
{
    double d1 = (b-a) * (c-a), d2 = (b-a) * (d-a),
           d3 = (d-c) * (a-c), d4 = (d-c) * (b-c);
    if(sgn(d1)*sgn(d2) > 0)
        return false;
    e = point( (c.x*d2 - d.x*d1) / (d2-d1) ,
               (c.y*d2 - d.y*d1) / (d2-d1) );
    return true;
}
```

N*log(n) Dijkstra(迪杰斯特拉)

```
long long v[MAXN],dis[MAXN],dui[MAXN],rear,front,dn,b[MAXN];
void up(long long x)
{
	long long i,j,k;
	i = x/2; j = x;
	while(i >= 1)
	{
	    if(dis[ dui[j] ] < dis[ dui[i] ] ) { swap(&dui[j],&dui[i]);
swap(&b[ dui[j] ],&b[ dui[i] ]);  }
	    else break;
	    j = i;
	    i /= 2;
	}
}
void jin(long long a)
{
	dui[++dn] = a;
	b[a] = dn;
	up(dn);
}
void chu(long long *a)
{
	long long i,j,k;
	*a = dui[1];
	swap(&dui[1],&dui[dn]);
	swap(&b[ dui[1] ],&b[ dui[dn] ]);
	dn--;
	i = 1;
	while(i<=dn/2)
	{   j = i*2;
	    if(j+1<=dn && dis[ dui[j] ] > dis[ dui[j+1] ]) j++;
	    if(dis[ dui[i] ] > dis[ dui[j] ]) { swap(&dui[i],&dui[j]); swap(&b[ dui[i] ],&b[ dui[j] ]); }
```

```
            else break;
            i = j;
        }
}
void dij(long long w)
{
        long long i,j,k,r;
        node *p;
        memset(v,0,sizeof(v));
        memset(dui,10,sizeof(dui));
        /*for(i=1;i<=s4;i++) dis[i] = MAXNUM;*/
        dn = 0;
        dis[w] = 0;
        for(i=1;i<=s4;i++) jin(i);
        for(i=1;i<=(n-1)*(n-1)+3;i++)
        {   chu(&k);    /*printf("%I64d:%I64d\n",k,dis[k]);*/
            for(p=g[k];p;p=p->next)
               if(dis[p->adj] > dis[k] + p->road)
               {   dis[p->adj] = dis[k] + p->road;
                   up(b[p->adj]);
               }
        }
}
```

O(N)求回文串

http://blog.csdn.net/linulysses/article/details/5634104

```
void getff()
{
    long i,j,k,r,w,id,am,mx;
    long p;
    memset(s,0,sizeof(s));
    memset(ff,0,sizeof(ff));
    n = strlen(b);
    s[0] = '#';
    for(i=1;i<=2*n;i++)
      if(i%2 == 1) s[i] = b[i/2];
      else s[i] = '#';
    m = 2*n;    w = j = id = am= mx = 0;
    p = 1;
    while(p < m)
    {   if(mx > p) { ff[p] = min( ff[ id-(p-id) ] , ff[id] - (p-id));}
        else ff[p] = 1;

        for(;s[p + ff[p]] == s[p - ff[p]]; ff[p]++);
```

```
      if(ff[p] + p > mx)
      {   mx = ff[p] + p;
        id = p;
      }

      p++;
    }
    for(i=1;i<=m;i++) ff[i]--;
}
```

```
 功能          |        示例          |   位运算
---------------------+--------------------------+-------------------
去掉最后一位        | (101101->10110)          | x shr 1
在最后加一个 0       | (101101->1011010)        | x shl 1
在最后加一个 1       | (101101->1011011)        | x shl 1+1
把最后一位变成 1      | (101100->101101)         | x or 1
把最后一位变成 0      | (101101->101100)         | x or 1-1
最后一位取反        | (101101->101100)         | x xor 1
把右数第 k 位变成 1    | (101001->101101,k=3)     | x or (1 shl (k-1))
把右数第 k 位变成 0    | (101101->101001,k=3)     | x and not (1 shl (k-1))
右数第 k 位取反       | (101001->101101,k=3)     | x xor (1 shl (k-1))
取末三位           | (1101101->101)           | x and 7
取末 k 位           | (1101101->1101,k=5)      | x and (1 shl k-1)
取右数第 k 位         | (1101101->1,k=4)         | x shr (k-1) and 1
把末 k 位变成 1       | (101001->101111,k=4)     | x or (1 shl k-1)
末 k 位取反          | (101001->100110,k=4)     | x xor (1 shl k-1)
把右边连续的 1 变成 0   | (100101111->100100000)   | x and (x+1)
把右起第一个 0 变成 1   | (100101111->100111111)   | x or (x+1)
把右边连续的 0 变成 1   | (11011000->11011111)     | x or (x-1)
取右边连续的 1        | (100101111->1111)        | (x xor (x+1)) shr 1
去掉右起第一个 1 的左边 | (100101000->1000)        | x and (x xor (x-1))
```

求 I 所有子集：

```
for(int j = i&(i-1); j; j = i&(j-1))
```

二分查找 32 位整数的前导 0 个数

这里用的 C 语言，我直接 Copy 的 Hacker's Delight 上的代码。这段代码写成 C 要好看些，写成 Pascal 的话会出现很多 begin 和 end，搞得代码很难看。程序思想是二分查找，应该很

简单，我就不细说了。

```
int nlz(unsigned x)
{
      int n;

      if (x == 0) return(32);
      n = 1;
      if ((x >> 16) == 0) {n = n +16; x = x <<16;}
      if ((x >> 24) == 0) {n = n + 8; x = x << 8;}
      if ((x >> 28) == 0) {n = n + 4; x = x << 4;}
      if ((x >> 30) == 0) {n = n + 2; x = x << 2;}
      n = n - (x >> 31);
      return n;
}
```

AC 自动机

```
typedef struct node
{
        struct node *fail,*next[26];
        int count;
        node()
        {   fail = NULL;
            count = 0;
            memset(next,NULL,sizeof(next));
        }

}node;

node *dui[500010];
char ss[MAXN];
int rear,front,n;
void build(char str[],node *t)
{
      node *p;
      int i=0,index;
      p = t;
      while(str[i])
      {   index = str[i]-'a';
          if(p->next[index] == NULL) p->next[index] = new node();
          p = p->next[index];
```

```
            i++;
        }
        p->count++;
}
void getac(node *t)
{
        int i;
        rear = front = 0;
        t->fail = NULL;
        dui[rear++] = t; rear %= MAXN;
        while(rear != front)
        {   node *temp = dui[front++];    front %= MAXN;
            node *p=NULL;
            for(i=0;i<26;i++)
            {   if(temp->next[i] != NULL)
                {   if(temp == t) temp->next[i]->fail = t;
                    else
                    {   p = temp->fail;
                        while(p != NULL)
                        {   if(p->next[i] != NULL)
                            {   temp->next[i]->fail = p->next[i];
                                break;
                            }
                            p=p->fail;
                        }
                        if(p == NULL) temp->next[i]->fail = t;
                    }
                    dui[rear++] = temp->next[i];    rear %= MAXN;
                }
            }
        }
}
void query(node *t)
{
        int i,ans=0,index,len=strlen(ss);
        node *p=t;
        i = 0;
        while(ss[i])
        {   index = ss[i]-'a';
            while(p->next[index]==NULL && p != t) p=p->fail;
            p = p->next[index];
            if(p == NULL) p = t;
            node *temp = p;
            while(temp != t && temp->count != -1)
```

```
            {    ans += temp->count;
                 temp->count = -1; //若只求在待匹配串中出现一次则加上这句。
                  temp = temp->fail;
            }
            i++;
        }
        printf("%d\n",ans);
}
```

读入

```
int  scanf(int  &num)
{
        char  in;
        while((in=getchar())!=EOF  &&  (in>'9'  ||  in<'0'));
        if(in==EOF)  return  0;
        num=in-'0';
        while(in=getchar(),in>='0'  &&  in<='9')  num*=10,num+=in-'0';
        return  1;
}
```

不支持负数

```
long long Gcd(long long a,long long b)
{
    for(long long t=a%b;t; a=b,b=t,t=a%b); return b;
}
long long ExpandGcd(long long a, long long b, long long &d, long long &x, long long &y)
{
    if( b ) { ExpandGcd( b, a%b , d, y, x); y -= a/b * x; }
    else { d = a; x = 1; y = 0; }
}
```

（欧拉公式）如果一个连通的平面图有 n 个点，m 条边和 f 个面，那么 $f=m-n+2$

卡特兰数 ： 1, 1, 2, 5, 14, 42, 132, 429, 1430, 4862, 16796, 58786, 208012, 742900, 2674440, 9694845, 35357670, 129644790, 477638700, 1767263190, 6564120420, 24466267020, 91482563640, 343059613650, 1289904147324, 4861946401452,

令 h(1)=1,h(0)=1，catalan 数满足递归式：

h(n)= h(0)*h(n-1)+h(1)*h(n-2) + ... + h(n-1)h(0) (其中 n>=2)

另类递归式：

h(n)=h(n-1)*(4*n-2)/(n+1);

该递推关系的解为：

h(n)=C(2n,n)/(n+1) (n=1,2,3,...)

```
//列主元 gauss 消去求解 a[][]x[]=b[]
//返回是否有唯一解,若有解在 b[]中
int gauss_cpivot(int n,double a[][MAXN],double b[]){
    int i,j,k,row;
    double maxp,t;
    for (k=0;k<n;k++){
        for (maxp=0,i=k;i<n;i++)
            if (fabs(a[i][k])>fabs(maxp))
                maxp=a[row=i][k];
        if (fabs(maxp)<eps)
            return 0;
        if (row!=k){
            for (j=k;j<n;j++)
                t=a[k][j],a[k][j]=a[row][j],a[row][j]=t;
            t=b[k],b[k]=b[row],b[row]=t;
        }
        for (j=k+1;j<n;j++){
            a[k][j]/=maxp;
            for (i=k+1;i<n;i++)
                a[i][j]-=a[i][k]*a[k][j];
        }
        b[k]/=maxp;
        for (i=k+1;i<n;i++)
            b[i]-=b[k]*a[i][k];
    }
    for (i=n-1;i>=0;i--)
        for (j=i+1;j<n;j++)
            b[i]-=a[i][j]*b[j];
```

```
    return 1;
}

//全主元 gauss 消去解 a[][]x[]=b[]
//返回是否有唯一解,若有解在 b[]中
int gauss_tpivot(int n,double a[][MAXN],double b[]){
    int i,j,k,row,col,index[MAXN];
    double maxp,t;
    for (i=0;i<n;i++)
        index[i]=i;
    for (k=0;k<n;k++){
        for (maxp=0,i=k;i<n;i++)
            for (j=k;j<n;j++)
                if (fabs(a[i][j])>fabs(maxp))
                    maxp=a[row=i][col=j];
        if (fabs(maxp)<eps)
            return 0;
        if (col!=k){
            for (i=0;i<n;i++)
                t=a[i][col],a[i][col]=a[i][k],a[i][k]=t;
            j=index[col],index[col]=index[k],index[k]=j;
        }
        if (row!=k){
            for (j=k;j<n;j++)
                t=a[k][j],a[k][j]=a[row][j],a[row][j]=t;
            t=b[k],b[k]=b[row],b[row]=t;
        }
        for (j=k+1;j<n;j++){
            a[k][j]/=maxp;
            for (i=k+1;i<n;i++)
                a[i][j]-=a[i][k]*a[k][j];
        }
        b[k]/=maxp;
        for (i=k+1;i<n;i++)
            b[i]-=b[k]*a[i][k];
    }
    for (i=n-1;i>=0;i--)
        for (j=i+1;j<n;j++)
            b[i]-=a[i][j]*b[j];
    for (k=0;k<n;k++)
        a[0][index[k]]=b[k];
    for (k=0;k<n;k++)
        b[k]=a[0][k];
    return 1;
```

```
}
```