Contents

1 数据结构

1.1 点分治

```
void Size(int x,int fa)
{
int pt,next;
    size[x]=1;
    for (pt=first[x];pt;pt=e[pt].next)
    {
        next=e[pt].to;
        if (next==fa||vis[next]) continue;
        Size(next,x);
        size[x]+=size[next];
    }
}
int Center(int x,int fa,int ori)
{
int pt,next;
    for (pt=first[x];pt;pt=e[pt].next)
    {
        next=e[pt].to;
        if (next==fa||vis[next]||2*size[next]<=size[ori]) continue;
        return Center(next,x,ori);
    }
    return x;
}
void DFS(int x)
{
int pt,next,i,j,low,high;
    Size(x,x);
    x=Center(x,x,x);
    vis[x]=true;
    for (pt=first[x];pt;pt=e[pt].next)
    {
        next=e[pt].to;
        if (vis[next]) continue;
        DFS(next);
    }
    vis[x]=false;
}
```

1.2 AC自动机

```
void Build_AC()
{
int low,high,i;
Node *temp;
    low=0;
    high=-1;
    for (i=0;i<26;i++)
    if (root->son[i]) que[++high]=root->son[i];
    for (;low<=high;low++)
        for (i=0;i<26;i++)
        if (que[low]->son[i])
        {
            que[++high]=que[low]->son[i];
            for (temp=que[low];temp!=root;temp=temp->fail)
            if (temp->fail->son[i])
            {
                que[high]->fail=temp->fail->son[i];
                break;
            }
        }
}
```

1.3 后缀数组

```
bool Same(int *st,int a,int b,int len)
{
	return st[a]==st[b]&&st[a+len]==st[b+len];
}
void DA(int m=1000)
{
int	cnt,i,len,*x=arr1,*y=arr2;
	memset(arr1,127,sizeof(arr1));
	memset(arr2,127,sizeof(arr2));
	for (i=0;i<=m;i++) sum[i]=0;
	for (i=1;i<=n;i++) sum[x[i]=st[i]]++;
	for (i=1;i<=m;i++) sum[i]+=sum[i-1];
	for (i=n;i>=1;i--) sa[sum[x[i]]--]=i;
	for (cnt=0,len=1;cnt<n;len<<=1,m=cnt)
	{
		for (cnt=0,i=n-len+1;i<=n;i++) y[++cnt]=i;
		for (i=1;i<=n;i++)
		if  (sa[i]>len) y[++cnt]=sa[i]-len;
		for (i=0;i<=m;i++) sum[i]=0;
		for (i=1;i<=n;i++) sum[x[y[i]]]++;
		for (i=1;i<=m;i++) sum[i]+=sum[i-1];
		for (i=n;i>=1;i--) sa[sum[x[y[i]]]--]=y[i];
		swap(x,y);
		for (cnt=1,x[sa[1]]=1,i=2;i<=n;i++)
		if  (Same(y,sa[i-1],sa[i],len)) x[sa[i]]=cnt; else x[sa[i]]=++cnt;
	}
	for (i=1;i<=n;i++) rank[sa[i]]=i;
}
void Height()
{
int	len,i;
	for (len=0,i=1;i<=n;height[rank[i++]]=len)
	{
		if  (len) len--;
		if  (rank[i]==1) continue;
		for (;st[i+len]==st[sa[rank[i]-1]+len];len++);
	}
}
```

2 数学

2.1 线性基

```
#include<bits/stdc++.h>
```

```
using namespace std;
#define B 30
#define N 10050

const int allset=(1<<B)-1;

int a[N];

struct LB
{
        int mat[B],cnt;
        multiset<int> st;
        LB(){}
        void clear()
        {
            st.clear();
            cnt=0;
            memset(mat,0,sizeof(mat));
        }
        void add(int x)
        {
        int i,j;
            for (i=B-1;i>=0;i--)
            if ((x>>i)&1)
            {
                if (mat[i]) x^=mat[i]; else
                {
                    cnt++;
                    mat[i]=x;
                    break;
                }
            }
        }
        void fix()
        {
        int i,j;
            for (i=0;i<B;i++)
            if (mat[i])
            {
                for (j=i+1;j<B;j++)
                if ((mat[j]>>i)&1) mat[j]^=mat[i];
            }
        }
        void preset()//正确性待定
        {
        int i;
            fix();
            for (i=0;i<B;i++)
            if (mat[i]) st.insert(mat[i]);
        }
        int kth(int k)//正确性待定
        {
        int i,ans;
        multiset<int>::iterator it;
```

```
            if (k<=0||k>(1<<cnt)-1) return 0;//无解
            for (ans=i=0,it=st.begin();it!=st.end();it++,i++)
            if ((k>>i)&1) ans^=(*it);
            return ans;
        }
        int getmax()
        {
        int i,ans;
            fix();
            ans=0;
            for (i=B-1;i>=0;i--)
            if (ans^mat[i]>ans) ans^=mat[i];
            return ans;
        }
}       tree[N*10];
```

2.2 类欧几里得

$\sum_{x=0}^{n} x^{k1}(\lfloor\frac{ax+b}{c}\rfloor)^{k2}$

```
#include<bits/stdc++.h>
using namespace std;
#define pb push_back
typedef long long ll;
const int MOD=1e9+7;
inline ll qp(ll a,ll b)
{
    ll x=1; a%=MOD;
    while(b)
    {
        if(b&1) x=x*a%MOD;
        a=a*a%MOD; b>>=1;
    }
    return x;
}
namespace Lagrange {
ll x[23333],y[23333],a[23333],g[23333],h[23333],p[23333]; int N;
void work()
{
    for(int i=0;i<N;++i) a[i]=0;
    g[0]=1;
    for(int i=0;i<N;++i)
    {
        for(int _=0;_<=i;++_)
            h[_+1]=g[_]; h[0]=0;
        for(int _=0;_<=i;++_)
            h[_]=(h[_]-g[_]*(ll)x[i])%MOD;
        for(int _=0;_<=i+1;++_) g[_]=h[_];
    }
    for(int i=0;i<N;++i)
    {
        for(int j=0;j<=N;++j) p[j]=g[j];
        for(int j=N;j;--j)
```

```
            p[j-1]=(p[j-1]+p[j]*(ll)x[i])%MOD;
        ll s=1;
        for(int j=0;j<N;++j) if(i!=j)
            s=s*(x[i]-x[j])%MOD;
        s=y[i]*qp(s,MOD-2)%MOD;
        for(int _=0;_<N;++_)
            a[_]=(a[_]+p[_+1]*s)%MOD;
    }
}
vector<int> feed(vector<int> v)
{
    N=v.size();
    for(int i=0;i<N;++i) x[i]=i,y[i]=v[i];
    work(); v.clear();
    for(int i=0;i<N;++i) v.pb(a[i]);
    while(v.size()&&!v.back()) v.pop_back();
    return v;
}
ll calc(vector<int>&v,ll xx)
{
    ll s=0,gg=1; xx%=MOD;
    for(int i=0;i<N;++i)
        s=(s+gg*v[i])%MOD,gg=gg*xx%MOD;
    return s;
}
}
using Lagrange::feed;
using Lagrange::calc;
//ps[k]=\sum_{i=0}^x i^k
vector<int> ps[2333];
//rs[k]=\sum_{i=0}^x ((i+1)^k-i^k)
vector<int> rs[2333];
struct arr{ll p[11][11];};
ll C[233][233];
arr calc(ll a,ll b,ll c,ll n)
{
    arr w;
    if(n==0) a=0;
    if(a==0||a*n+b<c)
    {
        for(int i=0;i<=10;++i)
        {
            ll t=calc(ps[i],n),s=b/c;
            for(int j=0;i+j<=10;++j)
                w.p[i][j]=t,t=t*s%MOD;
        }
        return w;
    }
    for(int i=0;i<=10;++i)
        w.p[i][0]=calc(ps[i],n);
    if(a>=c||b>=c)
    {
        arr t=calc(a%c,b%c,c,n);
        ll p=a/c,q=b/c;
        for(int i=0;i<=10;++i)
            for(int j=1;i+j<=10;++j)
            {
                ll s=0,px=1;
                for(int x=0;x<=j;++x,px=px*p%MOD)
                {
                    ll qy=1;
                    for(int y=0;x+y<=j;++y,qy=qy*q%MOD)
                    {
                        //x^(i) (px)^x q^y ??^(j-x-y)
                        s+=px*qy%MOD*C[j][x]%MOD*C[j-x][y]
                        %MOD*t.p[i+x][j-x-y]; s%=MOD;
                    }
                }
                w.p[i][j]=s;
            }
        return w;
    }
    ll m=(a*n+b)/c;
    arr t=calc(c,c-b-1,a,m-1);
    for(int i=0;i<=10;++i)
        for(int j=1;i+j<=10;++j)
        {
            ll s=calc(rs[j],m-1)*calc(ps[i],n)%MOD;
            for(int p=0;p<j;++p)
            {
                for(unsigned q=0;q<ps[i].size();++q)
                {
                    ll v=C[j][p]*ps[i][q]%MOD;
                    //v*t^p*((tc+c-b-1)/a)^q
                    s-=t.p[p][q]*v; s%=MOD;
                }
            }
            w.p[i][j]=s%MOD;
        }
    return w;
}
int T,n,a,b,c,k1,k2;
int main()
{
    freopen("a.in","r",stdin);
    freopen("a.out","w",stdout);
    for(int i=0;i<=230;++i)
    {
        C[i][0]=1;
        for(int j=1;j<=i;++j)
            C[i][j]=(C[i-1][j-1]+C[i-1][j])%MOD;
    }
    for(int i=0;i<=10;++i)
    {
        ll sp=0,sr=0; vector<int> p,r;
        for(int j=0;j<=20;++j)
            sp+=qp(j,i),sr+=qp(j+1,i)-qp(j,i),
            sp%=MOD,sr%=MOD,p.pb(sp),r.pb(sr);
        ps[i]=feed(p); rs[i]=feed(r);
    }
    scanf(”%d”,&T);
    while(T--)
    {
        scanf(”%d%d%d%d%d%d”,
        &n,&a,&b,&c,&k1,&k2);
        arr s=calc(a,b,c,n);
        int p=s.p[k1][k2];
        p=(p%MOD+MOD)%MOD;
        printf(”%d\n”,p);
    }
}
```

2.3 高斯消元法实数方程

```
void Gauss(int n,int m)
{
int i,j,k,t;
double mul;
    for (i=j=1;i<=n&&j<=m;i++,j++)
    {
        for (k=i+1;k<=n;k++)
        if (abs(mat[k][j])>abs(mat[i][j]))
            for (t=1;t<=m+1;t++) swap(mat[i][t],mat[k][t]);
        if (abs(mat[i][j])<eps)
        {
            i--;
            continue;
        }
        for (k=i+1;k<=n;k++)
        {
            mul=mat[k][j]/mat[i][j];
            for (t=1;t<=m+1;t++) mat[k][t]-=mat[i][t]*mul;
        }
    }
    for (i=n;i>=1;i--)//表示那个变量是否确定solved
    {
        for (j=1;j<=m;j++)
        if (abs(mat[i][j])>eps) break;
        if (j>m) continue;
        solved[j]=true;
        ans[j]=mat[i][m+1];
        for (k=j+1;k<=m;k++)
        if (abs(mat[i][k])>eps&&!solved[k]) solved[j]=false;
        for (k=j+1;k<=m;k++) ans[j]-=ans[k]*mat[i][k];
        ans[j]/=mat[i][j];
    }
}
```

2.4 高斯消元法模方程

```
long long Pow(long long x,long long y)
{
      if (y==0) return 1;
long long t=Pow(x,y/2);
      if (y&1) return t*t%mod*x%mod;
      return t*t%mod;
}
void Gauss(long long n,long long m)
{
long long i,j,k,t,lcm,muli,mulk;
      for (i=j=1;i<=n&&j<=m;i++,j++)
      {
           for (k=i;k<=n;k++)
           if (mat[k][j])
           {
                for (t=1;t<=m+1;t++) swap(mat[k][t],mat[i][t]);
                break;
           }
           if (mat[i][j]==0)
           {
                i--;
                continue;
           }
           for (k=i+1;k<=n;k++)
           if (mat[k][j])
           {
                lcm=mat[k][j]*mat[i][j]/Gcd(mat[k][j],mat[i][j]);
                muli=lcm/mat[i][j];
                mulk=lcm/mat[k][j];
                for (t=1;t<=m+1;t++)
                {
                     mat[k][t]=mat[k][t]*mulk-mat[i][t]*muli;
                     mat[k][t]=(mat[k][t]%mod+mod)%mod;
                }
           }
      }
      for (i=n;i>=1;i--)
      {
           for (j=1;j<=m;j++)
           if (mat[i][j]) break;
           if (j>m) continue;
           ans[j]=mat[i][m+1];
           for (k=j+1;k<=m;k++) ans[j]-=ans[k]*mat[i][k];
           ans[j]=(ans[j]*Pow(mat[i][j],mod-2)%mod+mod)%mod;
      }
}
```

2.5 扩展欧几里得

求一组整数解且防爆

```
void GCD_EX(long long A, long long &x, long long B, long long &y,
    long long C)
```

```
{
long long xx, yy, temp, gcd;
        if  (B == 0)
        {
            x = C;
            y = 0;
        }   else
        {
            GCD_EX(B, xx, A % B, yy, C);
            x = yy;
            temp = 1 - x / C * A;
            gcd = GCD(temp, B);
            temp /= gcd;
            B /= gcd;
            y = C / B * temp;
        }
        return;
}
```

2.6 FWT异或，与，或

```
void fwtXor(int* a, int len) {
    if(len == 1) return;
    int h = len >> 1;
    fwtXor(a, h);
    fwtXor(a + h, h);
    for(int i = 0; i < h; ++i) {
        int x1 = a[i];
        int x2 = a[i + h];
        a[i] = (x1 + x2) % mod;
        a[i + h] = (x1 - x2 + mod) % mod;
    }
}
void ifwtXor(int* a, int len) {
    if(len == 1) return;
    int h = len >> 1;
    for(int i = 0; i < h; ++i) {
        int y1 = a[i];
        int y2 = a[i + h];
        a[i] = 1ll*(y1 + y2) * div2 % mod;
        a[i + h] = 1ll*(y1 - y2 + mod) * div2 % mod;
    }
    ifwtXor(a, h);
    ifwtXor(a + h, h);
}
void fwtAnd(int* a, int len) {
    if(len == 1) return;
    int h = len >> 1;
    fwtAnd(a, h);
    fwtAnd(a + h, h);
    for(int i = 0; i < h; ++i) {
        int x1 = a[i];
        int x2 = a[i + h];
        a[i] = (x1 + x2) % mod;
        a[i + h] = x2;
    }
}
void ifwtAnd(int* a, int len) {
    if(len == 1) return;
    int h = len >> 1;
    for(int i = 0; i < h; ++i) {
        int y1 = a[i];
        int y2 = a[i + h];
        a[i] = (y1 - y2 + mod) % mod;
        a[i + h] = y2;
    }
    ifwtAnd(a, h);
    ifwtAnd(a + h, h);
}
void fwtOr(int* a, int len) {
    if(len == 1) return;
    int h = len >> 1;
    fwtOr(a, h);
    fwtOr(a + h, h);
    for(int i = 0; i < h; ++i) {
        int x1 = a[i];
        int x2 = a[i + h];
        a[i] = x1;
        a[i + h] = (x1 + x2) % mod;
    }
}
void ifwtOr(int* a, int len) {
    if(len == 1) return;
    int h = len >> 1;
    for(int i = 0; i < h; ++i) {
        int y1 = a[i];
        int y2 = a[i + h];
        a[i] = y1;
        a[i + h] = (y2 - y1 + mod) % mod;
    }
    ifwtOr(a, h);
    ifwtOr(a + h, h);
}
```

3 图论

3.1 tarjan

3.1.1 有向图强连通分量

```
void DFS(int x)
{
int     pt,next;
        dfn[x]=low[x]=++times;
        sk[++tp]=x;
        instack[x]=true;
        for (pt=first[x];pt;pt=e[pt].next)
        {
                next=e[pt].to;
                if (!dfn[next])
        {
            DFS(next);
            low[x]=min(low[x],low[next]);
        }   else
        if (instack[next]) low[x]=min(low[x],dfn[next]);
        }
        if (low[x]==dfn[x])
        {
                tot++;
                for (;tp;)
                {
                    instack[sk[tp]]=false;
                        belong[sk[tp--]]=tot;
                        if (sk[tp+1]==x) break;
                }
        }
}
for (i=1;i<=n;i++)
if (!dfn[i]) DFS(i);
```

3.1.2 点双联通分量

```
void DFS(int x,int fa)
{
int  pt,next;
     vis[x]=true;
     dfn[x]=low[x]=++times;
     sk[++tp]=x;
     for (pt=first[x];pt;pt=e[pt].next)
     {
         next=e[pt].to;
         if (e[pt].id==fa) continue;
         if (!vis[next])
         {
             DFS(next,e[pt].id);
             low[x]=min(low[x],low[next]);
             if (low[next]>=dfn[x])
             {
                 tot++;
                 vec[tot].clear();
                 for (;tp;)
                 {
                     vec[tot].push_back(sk[tp]);
                     tp--;
                     if (sk[tp+1]==next) break;
                 }
                 vec[tot].push_back(x);
             }
             }   else
             if  (dfn[next]>last) low[x]=min(low[x],dfn[next]);
       }
}
for (i=1;i<=n;i++)
if  (!vis[i])
{
    DFS(i,0);
    last=times;
    if  (tp)
    {
          tot++;
          vec[tot].clear();
          for (i=1;i<=tp;i++) vec[tot].push_back(sk[i]);
          tp=0;
    }
}
```

3.1.3 边双联通分量

```
void DFS(int x,int fa)
{
int   pt,next;
       vis[x]=true;
       dfn[x]=low[x]=++times;
       sk[++tp]=x;
       for (pt=first[x];pt;pt=e[pt].next)
       {
             next=e[pt].to;
             if  (e[pt].id==fa) continue;
             if  (!vis[next])
             {
                  DFS(next,e[pt].id);
                  low[x]=min(low[x],low[next]);
                  if  (low[next]>dfn[x])
                  {
                        tot++;
                        vec[tot].clear();
                        for (;tp;)
                        {
                              vec[tot].push_back(sk[tp]);
                              tp--;
                              if  (sk[tp+1]==next) break;
                        }
                  }
             }   else
             if  (dfn[next]>last) low[x]=min(low[x],dfn[next]);
       }
}
for (i=1;i<=n;i++)
if  (!vis[i])
{
    DFS(i,0);
    last=times;
    if (tp)
    {
        tot++;
        vec[tot].clear();
        for (i=1;i<=tp;i++) vec[tot].push_back(sk[i]);
        tp=0;
    }
}
```

3.2 网络流Dinic

```
struct Edge
{
      int to,flow,next;
}     e[1000050];
void Add(int x,int y,int z)
{
    e[++now].to=y;
    e[now].flow=z;
    e[now].next=first[x];
    first[x]=now;
}
bool Level()
{
int   low,high,pt,next;
    memset(level,-1,sizeof(level));
    q[low=high=0]=S;
    level[S]=0;
    for (;low<=high;low++)
        for (pt=first[q[low]];pt!=-1;pt=e[pt].next)
        {
            next=e[pt].to;
            if (level[next]!=-1||e[pt].flow<=0) continue;
            level[next]=level[q[low]]+1;
            q[++high]=next;
            if (next==T) return true;
        }
    return false;
}
int   Find(int x,int delta)
{
int   pt,next,temp,res=0;
    if (x==T||delta<=0) return delta;
    for (pt=last[x];pt!=-1;last[x]=pt=e[pt].next)
    {
        next=e[pt].to;
        if (level[next]!=level[x]+1) continue;
        temp=Find(next,min(delta,e[pt].flow));
        delta-=temp;
        res+=temp;
        e[pt].flow-=temp;
```

```
            e[pt^1].flow+=temp;
            if  (delta<=0) return res;
        }
        return res;
}
```

3.3 费用流

```
struct Edge
{
           int to,flow,cost,next;
}          e[1000050];
void Add(int x,int y,int flow,int cost)
{
        e[++now].to=y;
        e[now].flow=flow;
        e[now].cost=cost;
        e[now].next=first[x];
        first[x]=now;
}
bool Find()
{
int     low,high,pt,next;
        memset(inq,false,sizeof(inq));
        memset(f,127,sizeof(f));
        q[low=high=0]=S;
        f[S]=0;
        inq[S]=true;
        for  (;low<=high;inq[q[low++]]=false)
             for  (pt=first[q[low]];pt!=-1;pt=e[pt].next)
             {
                   next=e[pt].to;
                   if  (f[next]<=f[q[low]]+e[pt].cost||e[pt].flow<=0||q[low]==T||next==S) continue;
                   f[next]=f[q[low]]+e[pt].cost;
                   path[next]=q[low];
                   number[next]=pt;
                   if  (!inq[next])
                   {
                         inq[next]=true;
                         q[++high]=next;
                   }
             }
        return f[T]<999999999;
}
void Addflow()
{
int     t=999999999,i;
        for  (i=T;i!=S;i=path[i])  t=min(t,e[number[i]].flow);
        mincost+=t*f[T];
        maxflow+=t;
        for  (i=T;i!=S;i=path[i])
        {
            e[number[i]].flow-=t;
            e[number[i]^1].flow+=t;
        }
}
```

4 小算法

4.1 跨平台大随机数

```
int Rand()
{
    int ra = rand() % 32768;
    int rb = rand() % 32768;
    return (ra << 15) | rb;
}
```

4.2 读入优化

```
char *st,*nd,ch[1000050];
inline char Get()
{
        if  (st==nd)
        {
            st=ch;
            nd=st+fread(ch,1,1000007,stdin);
        }
        return *(st++);
}
inline int Read()
{
register int     t=0;
register char c=Get();
register bool fu=false;
      for (;c!='-'&&(c<'0'||'9'<c);c=Get());
      if  (c=='-')
      {
          fu=true;
          c=Get();
      }
      for (;'0'<=c&&c<='9';c=Get()) t=10*t+c-'0';
      if  (fu) return -t;
      return t;
}
```

4.3 KMP算法

```
          len = strlen(st);
          memset(prep, -1, sizeof(prep));
          memset(number, 0, sizeof(number));
          for (i = 1; i < len; i++)
          {
              for (j = i - 1; j != -1 && st[prep[j] + 1] != st[i]; j = prep[j]);
              prep[i] = j != -1 ? prep[j] + 1 : -1;
          }
```

4.4 扩展KMP

```
        extend[1] = number;
        for (far = 1; far < number; far++)
        if (tested[far] != tested[far + 1])
        {
            break;
        }
        extend[source = 2] = far - 1;
        for (j = 3; j <= number; j++)
        {
            temp = extend[j - source + 1];
            if (j + temp - 1 < far)
            {
                extend[j] = temp;
            } else
            {
                for (k = max(far + 1, j); k <= number; k++)
                if (tested[k] != tested[k - j + 1])
                {
                    break;
                }
                far = k - 1;
                source = j;
                extend[j] = far - source + 1;
            }
        }
```

5 计算几何

5.1 凸包

```
bool cmp(const Point &a,const Point &b)
{
     return F(a.x-b.x)<0||F(a.x-b.x)==0&&a.y<b.y;
}
void Gram(int id[],int n)
{
int i,mid;
     sort(id,id+n,cmp);
     tp=0;
```

```
    //凸包从最小的点出发，逆时针方向x
    for (i=0;i<n;i++)
    {
        for (;tp>=2&&Cross(p[sk[tp-1]]-p[sk[tp-2]],p[id[i]]-p[sk[tp-1]])<=0;tp--);
        //有重点必须使用小于等于不留共线点，无重点使用小于等于不留共线点，无重点使用小于留共线点
        sk[tp++]=id[i];
    }
    mid=tp;
    for (i=n-2;i>=0;i--)
    {
        for (;tp>mid&&Cross(p[sk[tp-1]]-p[sk[tp-2]],p[id[i]]-p[sk[tp-1]])<=0;tp--);
        //有重点必须使用小于等于不留共线点，无重点使用小于等于不留共线点，无重点使用小于留共线点
        sk[tp++]=id[i];
    }
    if (n>1) tp--;
}
```

5.2 定义

```
struct Point
{
        double x,y;
        Point(){}
        Point(double _x,double _y):x(_x),y(_y){}
};
struct Seg
{
    Point a,b;
    Seg(){}
    Seg(Point _a,Point _b):a(_a),b(_b){}
};
struct Circle
{
        double x,y,r;
        Point pt() {return Point(x,y);}
        double Area() {return pi*r*r;}
};
Point operator +(const Point &a,const Point &b)
{
    return Point(a.x+b.x,a.y+b.y);
}
Point operator -(const Point &a,const Point &b)
{
    return Point(a.x-b.x,a.y-b.y);
}
Point operator *(const Point &a,double b)
{
    return Point(a.x*b,a.y*b);
}
```

```
Point operator /(const Point &a,double b)
{
    return Point(a.x/b,a.y/b);
}
int F(double x)
{
    if (x>eps) return 1;
    if (x<-eps) return -1;
    return 0;
}
bool operator ==(const Point &a,const Point &b)
{
    return F(a.x-b.x)==0&&F(a.y-b.y)==0;
}
double Dist(const Point &a)
{
    return sqrt(a.x*a.x+a.y*a.y);
}
double Dot(const Point &a,const Point &b)
{
    return a.x*b.x+a.y*b.y;
}
double Cross(const Point &a,const Point &b)
{
    return a.x*b.y-a.y*b.x;
}
Point Rotate(const Point &p,double a) // 逆时针旋转
{
    return Point(p.x*cos(a)-p.y*sin(a),p.x*sin(a)+p.y*cos(a));
}
Point Inter(Seg a,Seg b)   // 两线段相交（前提有交点）
{
double s=Cross(a.b-a.a,b.a-a.a),t=Cross(a.b-a.a,b.b-a.a);
       return b.a+(b.b-b.a)*s/(s-t);
}
vector<Point> SegCir(Seg seg,Point pt,double r)//线圆
{
vector<Point> ans;
double mul;
Point vec,mid;
        ans.clear();
        vec=Rotate(seg.b-seg.a,pi/2);
        mid=Inter(seg,Seg(pt,pt+vec));
        if  (F(Dist(pt-mid)-r)>0) return ans;
        if  (F(Dist(pt-mid)-r)==0)
        {
                ans.push_back(mid);
                ans.push_back(mid);
                return ans;
        }
        vec=seg.b-seg.a;
        mul=sqrt(r*r-Dist2(mid-pt))/Dist(vec);
        ans.push_back(mid+vec*mul);
        ans.push_back(mid-vec*mul);
```

```
          return ans;
}
vector<Point> Circir(Circle a,Circle b)//圆圆相交
{
vector<Point> ans;
double dis,dis2,alpha;
Point pa,pb,vec;
        ans.clear();
        if (a.r<b.r) swap(a,b);
        pa=a.pt();
        pb=b.pt();
        vec=pb-pa;
        dis=Dist(vec);
        dis2=Dist2(vec);
        if (F(dis-(a.r+b.r))>0||F(dis-(a.r-b.r))<0) return ans;
        if (F(dis-(a.r+b.r))==0)
        {
            ans.push_back(pa+vec*a.r/(a.r+b.r));
            return ans;
        }
        if (F(dis-(a.r-b.r))==0)
        {
            ans.push_back(pa+vec*a.r/(a.r-b.r));
            return ans;
        }
        alpha=acos((a.r*a.r+dis2-b.r*b.r)/2/a.r/dis);
        ans.push_back(pa+Rotate(vec,alpha)*a.r/dis);
        ans.push_back(pa+Rotate(vec,-alpha)*a.r/dis);
        return ans;
}
double Bing(double ra,double rb,double dis)
{
double alpha,beta;
         if (ra<rb) swap(ra,rb);
         if (F(dis-(ra-rb))<=0) return pi*ra*ra;
         if (F(dis-(ra+rb))>=0) return pi*ra*ra+pi*rb*rb;
         alpha=acos((ra*ra+dis*dis-rb*rb)/2/dis/ra);
         beta=acos((rb*rb+dis*dis-ra*ra)/2/dis/rb);
         return (pi-alpha)*ra*ra+(pi-beta)*rb*rb+ra*dis*sin(alpha);
}

double Jiao(double ra,double rb,double dis)
{
         return pi*ra*ra+pi*rb*rb-Bing(ra,rb,dis);
}

Point Gongmid(Circle a,Circle b)//正确性待定
{
Point pa=a.pt(),pb=b.pt();
        return pa+(pb-pa)*a.r/(a.r+b.r);
}

Point Gongright(Circle a,Circle b)
{
```

```
Point pa=a.pt(),pb=b.pt();
      return pa+(pb-pa)*a.r/(a.r-b.r);
}

int Ptinpol(Point pt)
{
int i,k,d1,d2,wn=0;
    for(i=0;i<n;i++)
    {
        if(Ins(pt,Seg(p[i],p[(i+1)%n]))) return 2;
        k=F(Cross(p[(i+1)%n]-p[i],pt-p[i]));
        d1=F(p[i].y-pt.y);
        d2=F(p[(i+1)%n].y-pt.y);
        if(k>0&&d1<=0&&d2>0)wn++;
        if(k<0&&d2<=0&&d1>0)wn--;
    }
    return wn!=0;
}
bool Cirinpol(Point pt)//需要点在多边形内的前提
{
int i;
double nearest;
     nearest=1e+100;
     for (i=0;i<n;i++)
     {
         nearest=min(nearest,Dist(p[i]-pt));
         if (F(Dot(pt-p[i],p[(i+1)%n]-p[i]))>0&&
             F(Dot(pt-p[(i+1)%n],p[i]-p[(i+1)%n]))>0)
             nearest=min(nearest,abs(Cross(p[i]-pt,p[(i+1)%n]-pt))/dis[i]);
     }
     return F(nearest-r)>=0;
}
bool Ins(const Point &p,const Seg &s)
{
     return F(Cross(s.a-p,s.b-p))==0&&
            F(p.x-min(s.a.x,s.b.x))>=0&&
            F(p.x-max(s.a.x,s.b.x))<=0&&
            F(p.y-min(s.a.y,s.b.y))>=0&&
            F(p.y-max(s.a.y,s.b.y))<=0;
}
double PS(const Point &p,const Seg &s) 点到线段最短距离
{
        if (F(Dot(p-s.a,s.b-s.a))<0||F(Dot(p-s.b,s.a-s.b))<0) return min(Dist(p-s.a),Dist(p-s.b));
        return abs(Cross(s.a-p,s.b-p))/Dist(s.a-s.b);
}
double SS(const Seg &a,const Seg &b) 线段到线段最短距离
{
       return min(min(PS(a.a,b),PS(a.b,b)),min(PS(b.a,a),PS(b.b,a)));
}
double Alpha(Point a,Point b)
{
double ans;
```

```
        ans=atan2(b.y,b.x)-atan2(a.y,a.x);
        if  (ans<0) ans=-ans;
        if  (ans>pi) ans=2*pi-ans;
        return ans;
}
double Shan(Circle c,double a)
{
        return c.r*c.r*a/2;
}
```

5.3 半平面交

```

bool Cmphp(Seg a,Seg b)
{
Point va,vb;
double dega,degb;
      va=a.b-a.a;
      vb=b.b-b.a;
      dega=atan2(va.y,va.x);
      degb=atan2(vb.y,vb.x);
      return F(dega-degb)<0||F(dega-degb)==0&&Cross(a.b-a.a,b.a-a.a)<0;
}
void HalfPlane(Seg hp[],int n,Point pol[],int &pols)
{
int  tp,i,low,high;
Point mid;
      hp[n++]=Seg(Point(-oo,-oo),Point(oo,-oo));
      hp[n++]=Seg(Point(oo,-oo),Point(oo,oo));
      hp[n++]=Seg(Point(oo,oo),Point(-oo,oo));
      hp[n++]=Seg(Point(-oo,oo),Point(-oo,-oo));
      sort(hp,hp+n,Cmphp);
      tp=0;//sk 0~tp-1
      low=0;
      high=-1;
      for (i=0;i<n;i++)
      if  (high-low+1==0||F(Cross(sk[high].b-sk[high].a,hp[i].b-hp[i].a)))
      {
           for (;low<high;high--)
           {
                mid=Inter(sk[high],sk[high-1]);
                if  (F(Cross(hp[i].b-hp[i].a,mid-hp[i].a))>0) break;
           }
           for (;low<high;low++)
           {
                mid=Inter(sk[low],sk[low+1]);
                if  (F(Cross(hp[i].b-hp[i].a,mid-hp[i].a))>0) break;
           }
           sk[++high]=hp[i];
      }
```

```
    for (;low<high;high--)
    {
        mid=Inter(sk[high],sk[high-1]);
        if (Cross(sk[low].b-sk[low].a,mid-sk[low].a)>0) break;
    }
    tp=high-low+1;
    for (i=0;i<tp;i++) sk[i]=sk[low+i];
    pols=0;
    if (tp<=2) return;
    for (i=0;i<tp;i++) pol[pols++]=Inter(sk[i],sk[(i+1)%tp]);
}
```

5.4 圆与多边形交集

```
double CT(Circle c,Point a,Point b) 圆与三角形交（多边形）
{
double da,db;
Seg     s;
vector <Point> temp;
        da=Dist(a-c.pt());
        db=Dist(b-c.pt());
        if (da>db)
        {
            swap(a,b);
            swap(da,db);
        }
        s=Seg(a,b);
        temp=CS(c,s);
        if (F(db-c.r)<=0) return 0.5*abs(Cross(a-c.pt(),b-c.pt()));
        if (F(da-c.r)<0)
        {
            if (F(Dot(a-temp[1],b-temp[1]))<0) swap(temp[0],temp[1]);
            return Shan(c,Alpha(temp[0]-c.pt(),b-c.pt()))+0.5*abs(
                Cross(a-c.pt(),temp[0]-c.pt()));
        }
        if (!temp.size()) return Shan(c,Alpha(a-c.pt(),b-c.pt()));
        if (Ins(temp[1],s)&&Dist2(a-temp[1])<Dist2(a-temp[0])) swap(
            temp[0],temp[1]);
        if (Ins(temp[0],s)&&Ins(temp[1],s))
        {
            return Shan(c,Alpha(a-c.pt(),temp[0]-c.pt()))+
                   Shan(c,Alpha(b-c.pt(),temp[1]-c.pt()))+
                   0.5*abs(Cross(temp[0]-c.pt(),temp[1]-c.pt()));
        }
        return Shan(c,Alpha(a-c.pt(),b-c.pt()));
}
```

5.5 三角形面积并

```
#include<math.h>
#include<stdio.h>
```

```
#include<string.h>
#include<algorithm>
#define N 333
#define pr pair<ld,ld>
using namespace std;
typedef long double ld;
const ld EPS=1e-8;
const ld INF=1e100;
struct Point
{
    ld x,y;
    Point(){}
    Point(ld _,ld __):x(_),y(__){}
    void read()
    {
        double _,__;
        scanf("%lf%lf",&_,&__);
        x=_,y=__;
    }
    friend bool operator <(Point a,Point b)
    {
        if(fabs(a.x-b.x)<EPS)
        return a.y<b.y;
        return a.x<b.x;
    }
    friend Point operator +(Point a,Point b)
    {
        return Point(a.x+b.x,a.y+b.y);
    }
    friend Point operator -(Point a,Point b)
    {
        return Point(a.x-b.x,a.y-b.y);
    }
    friend Point operator *(ld a,Point b)
    {
        return Point(a*b.x,a*b.y);
    }
    friend ld operator *(Point a,Point b)
    {
        return a.x*b.x+a.y*b.y;
    }
    friend ld operator ^(Point a,Point b)
    {
        return a.x*b.y-a.y*b.x;
    }
}a[N][3],Poi[N*N];
struct Line
{
    Point p,v;
    Line(){}
    Line(Point _,Point __){p=_,v=__-_;}
    Point operator [](int k)
    {
        if(k)    return p+v;
        else    return p;
    }
    friend bool Cross(Line a,Line b)
    {
        return (a.v^b[0]-a.p)*(a.v^b[1]-a.p)<-EPS&&(b.v^a[0]-b.p)*(b.v^a[1]-b.p)<-EPS;
    }
    friend Point getP(Line a,Line b)
    {
        Point u=a.p-b.p;
        ld temp=(b.v^u)/(a.v^b.v);
        return a.p+temp*a.v;
    }
}l[N][3],T;
pr p[N];
int main()
{
    int n,m,i,j,k,x,y,cnt,tot;
    ld ans,last,A,B,sum;
    scanf(”%d”,&n);
    for(i=1,tot=0;i<=n;i++)
    {
        a[i][0].read(),a[i][1].read(),a[i][2].read();
        Poi[++tot]=a[i][0],Poi[++tot]=a[i][1],Poi[++tot]=a[i][2];
        sort(a[i],a[i]+3);
        if((a[i][2]-a[i][0]^a[i][1]-a[i][0])>EPS)
            l[i][0]=Line(a[i][0],a[i][2]),l[i][1]=Line(a[i][2],a[i][1]),l[i][2]=Line(a[i][1],a[i][0]);
        else
            l[i][0]=Line(a[i][2],a[i][0]),l[i][1]=Line(a[i][1],a[i][2]),l[i][2]=Line(a[i][0],a[i][1]);
    }
    for(i=1;i<=n;i++)
    {
        for(j=1;j<i;j++)
        {
            for(x=0;x<3;x++)
                for(y=0;y<3;y++)
                {
                    if(Cross(l[i][x],l[j][y]))
                        Poi[++tot]=getP(l[i][x],l[j][y]);
                }
        }
    }
    sort(Poi+1,Poi+tot+1);
    ans=0,last=Poi[1].x;
    T=Line(Point(0,-INF),Point(0,INF));
    for(i=2;i<=tot;i++)
    {
        T.p.x=(last+Poi[i].x)/2;
        for(j=1,cnt=0;j<=n;j++)
        {
            if(Cross(l[j][0],T))
            {
                A=getP(l[j][0],T).y;
                if(Cross(l[j][1],T))
                    B=getP(l[j][1],T).y;
                else
                    B=getP(l[j][2],T).y;
                if(A>B)    swap(A,B);
                p[++cnt]=pr(A,B);
            }
        }
        sort(p+1,p+cnt+1);
        for(j=1,sum=0,A=-INF;j<=cnt;j++)
        {
            if(p[j].first>A)
            {
                sum+=p[j].second-p[j].first;
                A=p[j].second;
            }
            else
            {
                if(p[j].second>A)
                    sum+=p[j].second-A,A=p[j].second;
            }
        }
        ans+=(Poi[i].x-last)*sum;
        last=Poi[i].x;
    }
    printf("%.2lf\n",(double)ans);
    return 0;
}
```

5.6 K圆并

```
#include <cstdio>
#include <cstdlib>
#include <climits>
#include <iostream>
#include <algorithm>
#include <cstring>
#include <string>
#include <queue>
#include <map>
#include <vector>
#include <bitset>
#include <cmath>
#include <set>
#include <utility>
#include <ctime>
#define sqr(x) ((x)*(x))
using namespace std;

const int N = 1010;
const double eps = 1e-8;
const double pi = acos(-1.0);
double area[N];
int n;

int dcmp(double x) {
    if (x < -eps) return -1; else return x > eps;
}

struct cp {
    double x, y, r, angle;
    int d;
    cp(){}
    cp(double xx, double yy, double ang = 0, int t = 0) {
        x = xx;  y = yy;  angle = ang;  d = t;
    }
    void get() {
        scanf("%lf%lf%lf", &x, &y, &r);
        d = 1;
    }
}cir[N], tp[N * 2];

double dis(cp a, cp b) {
    return sqrt(sqr(a.x - b.x) + sqr(a.y - b.y));
}

double cross(cp p0, cp p1, cp p2) {
    return (p1.x - p0.x) * (p2.y - p0.y) - (p1.y - p0.y) * (p2.x - p0.x);
}

int CirCrossCir(cp p1, double r1, cp p2, double r2, cp &cp1, cp &cp2) {
    double mx = p2.x - p1.x, sx = p2.x + p1.x, mx2 = mx * mx;
    double my = p2.y - p1.y, sy = p2.y + p1.y, my2 = my * my;
    double sq = mx2 + my2, d = -(sq - sqr(r1 - r2)) * (sq - sqr(r1 + r2));
    if (d + eps < 0) return 0; if (d < eps) d = 0; else d = sqrt(d);
    double x = mx * ((r1 + r2) * (r1 - r2) + mx * sx) + sx * my2;
    double y = my * ((r1 + r2) * (r1 - r2) + my * sy) + sy * mx2;
    double dx = mx * d, dy = my * d; sq *= 2;
    cp1.x = (x - dy) / sq; cp1.y = (y + dx) / sq;
    cp2.x = (x + dy) / sq; cp2.y = (y - dx) / sq;
    if (d > eps) return 2; else return 1;
}

bool circmp(const cp& u, const cp& v) {
    return dcmp(u.r - v.r) < 0;
}

bool cmp(const cp& u, const cp& v) {
    if (dcmp(u.angle - v.angle)) return u.angle < v.angle;
    return u.d > v.d;
}

```

```
double calc(cp cir, cp cp1, cp cp2) {
    double ans = (cp2.angle - cp1.angle) * sqr(cir.r)
        - cross(cir, cp1, cp2) + cross(cp(0, 0), cp1, cp2);
    return ans / 2;
}

void CirUnion(cp cir[], int n) {
    cp cp1, cp2;
    sort(cir, cir + n, circmp);
    for (int i = 0; i < n; ++i)
        for (int j = i + 1; j < n; ++j)
            if (dcmp(dis(cir[i], cir[j]) + cir[i].r - cir[j].r) <= 0)
                cir[i].d++;
    for (int i = 0; i < n; ++i) {
        int tn = 0, cnt = 0;
        for (int j = 0; j < n; ++j) {
            if (i == j) continue;
            if (CirCrossCir(cir[i], cir[i].r, cir[j], cir[j].r,
                cp2, cp1) < 2) continue;
            cp1.angle = atan2(cp1.y - cir[i].y, cp1.x - cir[i].x);
            cp2.angle = atan2(cp2.y - cir[i].y, cp2.x - cir[i].x);
            cp1.d = 1;     tp[tn++] = cp1;
            cp2.d = -1;    tp[tn++] = cp2;
            if (dcmp(cp1.angle - cp2.angle) > 0) cnt++;
        }
        tp[tn++] = cp(cir[i].x - cir[i].r, cir[i].y, pi, -cnt);
        tp[tn++] = cp(cir[i].x - cir[i].r, cir[i].y, -pi, cnt);
        sort(tp, tp + tn, cmp);
        int p, s = cir[i].d + tp[0].d;
        for (int j = 1; j < tn; ++j) {
            p = s;  s += tp[j].d;
            area[p] += calc(cir[i], tp[j - 1], tp[j]);
        }
    }
}

void solve() {
    for (int i = 0; i < n; ++i)
        cir[i].get();
    memset(area, 0, sizeof(area));
    CirUnion(cir, n);
    for (int i = 1; i <= n; ++i) {
        area[i] -= area[i + 1];
        printf("[%d]␣=␣%.3lf\n", i, area[i]);
    }
}

int main() {
    freopen("a.in", "r", stdin);
    while (scanf("%d", &n) != EOF) {
        solve();
    }
    return 0;
}
```

6 三维计算几何

6.1 基本定义

```
Point Cross(Point a,Point b)
{
      return Point(a.y*b.z-a.z*b.y,a.z*b.x-a.x*b.z,a.x*b.y-a.y*b.x);
}
double Crossxy(Point a,Point b)
{
       return a.x*b.y-a.y*b.x;
}
vector<Point> SegPlane(Seg seg,Plane p)
{
double        s,t;
Point         fa;
vector<Point> ans;
              ans.clear();
              fa=Cross(p.b-p.a,p.c-p.a);
              if  (F(Dot(fa,seg.b-seg.a))==0) return ans;
              s=Dot(p.a-seg.a,fa)/Dist(fa);
              t=Dot(p.a-seg.b,fa)/Dist(fa);
              ans.push_back(seg.a+(seg.b-seg.a)*s/(s-t));
              return ans;
}
```

6.2 一些补充

```
\\ mixed product
double Mix(Point3 a,Point3 b,Point3 c)
{
        return Dot(Cross(a,b),c);
}
\\ distance from point to plane
double PP(Point3 pt,Plane pl)
{
Point3 fa=Cross(pl.b-pl.a,pl.c-pl.a);
        return abs(Dot(fa,pt-pl.a))/Dist(fa);
}
\\ get the center point from 3D(need plane well prepared)
Point3 Getcenter(Point3 p[],int n,Plane pp[],int nn)
{
int i;
double sumv,tempv;
Point3 sum;
        sumv=0;
        sum=Point3(0,0,0);
        for (i=0;i<nn;i++)
        {
                tempv=Mix(pp[i].b-pp[i].a,pp[i].c-pp[i].a,Point3
                    (0,0,0)-pp[i].a);
                sum=sum+(pp[i].a+pp[i].b+pp[i].c)*tempv/4.0;
                sumv+=tempv;
        }
        return sum/sumv;
}
```

7 bitset用法

```
C++ bitset 用法


C的++ bitset 在 bitset 头文件中，它是一种类似数组的结构，它的每一个元素只能是0或1，每个元素仅用1空间。bit下面是具体用法构造函数常用构造函数有四种，如下




 bitset复制代码


    bitset<4> bitset1    ;//无参构造，长度为4，默认每一位为0

    bitset<8> bitset2(12)    ;//长度为8，二进制保存，前面用0补充

    string s = "100101";
    bitset<10> bitset3(s)    ;//长度为，前面用0补充10

    char s2[] = "10101";
    bitset<13> bitset4(s2)    ;//长度为，前面用0补充13

    cout << bitset1 << endl    ;//0000
    cout << bitset2 << endl    ;//00001100
    cout << bitset3 << endl    ;//0000100101
    cout << bitset4 << endl    ;//0000000010101复制代码注意：用字符串构造时，字符串只能包含





  '0' 或 '1' ，否则会抛出异常。构造时，需在中表明

<>bitset 的大小即(size)。在进行有参构造时，若参数的二进制表示比的小，则在前面用0补充

 bitsetsize如上面的栗子()；若比大，参数为整数时取后面部分，参数为字符串时取前面部分bitsize如下面栗子()：复制代码


    bitset<2> bitset1(12)    ;//的二进制为（长度为4），但121100的bitset1size，只取后面部分，即=200

    string s = "100101"    ;
    bitset<4> bitset2(s)    ;//的ssize，而=6的bitsetsize，只取前面部分，即=41001

    char s2[] = "11101";
    bitset<4> bitset3(s2)    ;//与同理，只取前面部分，即bitset21110

    cout << bitset1 << endl    ;//00
    cout << bitset2 << endl    ;//1001
    cout << bitset3 << endl    ;//1110复制代码可用的操作符对于二进制有位操作符，具体如下




bitset复制代码


    bitset<4> foo (string("1001"));
    bitset<4> bar (string("0011"));

    cout << (foo^=bar) << endl;        // 1010 (对按位异或后赋值给foobarfoo)
    cout << (foo&=bar) << endl;        // 0010 按位与后赋值给(foo)
    cout << (foo|=bar) << endl;        // 0011 按位或后赋值给(foo)

    cout << (foo<<=2) << endl;         // 1100 左移2位，低位补0，有自身赋值()
    cout << (foo>>=1) << endl;         // 0110 右移1位，高位补0，有自身赋值()

    cout << (~bar) << endl;            // 1100 按位取反()
    cout << (bar<<1) << endl;          // 0110 左移，不赋值()
    cout << (bar>>1) << endl;          // 0001 右移，不赋值()

    cout << (foo==bar) << endl;        // false 为(0110==0011false)
    cout << (foo!=bar) << endl;        // true 为 (0110!=0011true)

    cout << (foo&bar) << endl;         // 0010 按位与，不赋值()
    cout << (foo|bar) << endl;         // 0111 按位或，不赋值()
    cout << (foo^bar) << endl;         // 0101 按位异或，不赋值()复制代码此外，可以通过

 [ ] 访问元素类似数组()，注意最低位下标为0，如下：

    bitset<4> foo ("1011");

    cout << foo[0] << endl    ;//1
    cout << foo[1] << endl    ;//1
    cout << foo[2] << endl    ;//0当然，通过这种方式对某一位元素赋值也是可以的，栗子就不放了。可用函数还支持一些有意思的函数，比如：





```

```
bitset复制代码



    bitset<8> foo ("10011011");

    cout << foo.count() << endl    ;//    （5函数用来求中的位数，中共有 5 个 1 countbitset1foo
    cout << foo.size() << endl    ; //    （8函数用来求的大小，一共有 8 位sizebitset

    cout << foo.test(0) << endl    ;//    （函数用来查下标处的元素是 0 还是 1，并返回或，此处truetestfalsetruefoo为 1，返回[0]true
    cout << foo.test(2) << endl    ;//    （同理，falsefoo为 0，返回[2]false

    cout << foo.any() << endl    ;//    （函数检查中是否有 1 trueanybitset
    cout << foo.none() << endl    ;//    （函数检查中是否没有 1 falsenonebitset
    cout << foo.all() << endl    ;//    （函数检查中是否全部为 1 falseallbitset复制代码补充说明一下：函数会对下标越界作出检查，而通过


test [ ] 访问元素却不会经过下标检查，所以，在两种方式通用的情况下，选择函数更安全一些test另外，含有一些函数：   复制代码






    bitset<8> foo ("10011011");

    cout << foo.flip(2) << endl    ;//    （10011111函数传参数时，用于将参数位取反，本行代码将下标 2 处flipfoo反转，即 0 变 1， 1 变 0 ””
    cout << foo.flip() << endl    ; //    （01100000函数不指定参数时，将每一位全部取反flipbitset

    cout << foo.set() << endl        ;//    （11111111函数不指定参数时，将的每一位全部置为 1 setbitset
    cout << foo.set(3,0) << endl    ;//    （11110111函数指定两位参数时，将第一参数位的元素置为第二参数的值，本行对的操作相当于setfoofoo[3]=0
    cout << foo.set(3) << endl    ;  //    （11111111函数只有一个参数时，将参数下标处置为 1 set

    cout << foo.reset(4) << endl    ;//    （11101111函数传一个参数时将参数下标处置为 0 reset
    cout << foo.reset() << endl    ; //    （00000000函数不传参数时将的每一位全部置为 0 resetbitset复制代码同样，它们也都会检查下标是否越界，如果越界就会抛出异常最后，还有一些类型转换的函数，如下：复制代码






    bitset<8> foo ("10011011");

    string s = foo.to_string() ;//将转换成类型bitsetstring
    unsigned long a = foo.to_ulong() ;//将转换成bitsetunsigned 类型long
    unsigned long long b = foo.to_ullong() ;//将转换成bitsetunsigned long 类型long

    cout << s << endl ;//10011011
    cout << a << endl ;//155
    cout << b << endl ;//155复制代码
```