

Contents

1	数据结构	1
1.1	点分治	1
1.2	AC自动机	2
1.3	后缀数组	3
2	数学	3
2.1	线性基	3
2.2	类欧几里得	5
2.3	高斯消元法实数方程	8
2.4	高斯消元法模方程	8
2.5	扩展欧几里得	9
2.6	FWT异或, 与, 或	10
3	图论	11
3.1	tarjan	11
3.1.1	有向图强连通分量	11
3.1.2	点双联通分量	12
3.1.3	边双联通分量	13
3.2	网络流Dinic	14
3.3	费用流	15
4	小算法	16
4.1	跨平台大随机数	16
4.2	读入优化	16
4.3	KMP算法	16
4.4	扩展KMP	17
5	计算几何	17
5.1	凸包	17
5.2	定义	18
5.3	半平面交	22
5.4	圆与多边形交集	23
5.5	三角形面积并	23
5.6	K圆并	26
6	三维计算几何	29
6.1	基本定义	29
6.2	一些补充	29
7	bitset用法	30

1 数据结构

1.1 点分治

```
1 void Size(int x,int fa)
2 {
3     int pt,next;
4     size[x]=1;
5     for (pt=first[x];pt;pt=e[pt].next)
6     {
7         next=e[pt].to;
```

```

8         if (next==fa || vis[next]) continue;
9         Size(next,x);
10        size[x]+=size[next];
11    }
12 }
13 int Center(int x,int fa,int ori)
14 {
15     int pt,next;
16     for (pt=first[x];pt;pt=e[pt].next)
17     {
18         next=e[pt].to;
19         if (next==fa || vis[next] || 2*size[next]<=size[ori]) continue;
20         return Center(next,x,ori);
21     }
22     return x;
23 }
24 void DFS(int x)
25 {
26     int pt,next,i,j,low,high;
27     Size(x,x);
28     x=Center(x,x,x);
29     vis[x]=true;
30     for (pt=first[x];pt;pt=e[pt].next)
31     {
32         next=e[pt].to;
33         if (vis[next]) continue;
34         DFS(next);
35     }
36     vis[x]=false;
37 }

```

1.2 AC自动机

```

1 void Build_AC()
2 {
3     int low,high,i;
4     Node *temp;
5     low=0;
6     high=-1;
7     for (i=0;i<26;i++)
8     if (root->son[i]) que[++high]=root->son[i];
9     for (;low<=high;low++)
10    for (i=0;i<26;i++)
11    if (que[low]->son[i])
12    {
13        que[++high]=que[low]->son[i];
14        for (temp=que[low];temp!=root;temp=temp->fail)
15        if (temp->fail->son[i])
16        {
17            que[high]->fail=temp->fail->son[i];
18            break;
19        }

```

```

20     }
21 }

```

1.3 后缀数组

```

1  bool Same(int *st, int a, int b, int len)
2  {
3      return st[a]==st[b]&&st[a+len]==st[b+len];
4  }
5  void DA(int m=1000)
6  {
7      int cnt, i, len, *x=arr1, *y=arr2;
8      memset(arr1, 127, sizeof(arr1));
9      memset(arr2, 127, sizeof(arr2));
10     for (i=0; i<=m; i++) sum[i]=0;
11     for (i=1; i<=n; i++) sum[x[i]=st[i]]++;
12     for (i=1; i<=m; i++) sum[i]+=sum[i-1];
13     for (i=n; i>=1; i--) sa[sum[x[i]]-1]=i;
14     for (cnt=0, len=1; cnt<n; len<=1, m=cnt)
15     {
16         for (cnt=0, i=n-len+1; i<=n; i++) y[++cnt]=i;
17         for (i=1; i<=n; i++)
18             if (sa[i]>len) y[++cnt]=sa[i]-len;
19         for (i=0; i<=m; i++) sum[i]=0;
20         for (i=1; i<=n; i++) sum[x[y[i]]]++;
21         for (i=1; i<=m; i++) sum[i]+=sum[i-1];
22         for (i=n; i>=1; i--) sa[sum[x[y[i]]]-1]=y[i];
23         swap(x, y);
24         for (cnt=1, x[sa[1]]=1, i=2; i<=n; i++)
25             if (Same(y, sa[i-1], sa[i], len)) x[sa[i]]=cnt; else x[sa[i]]
                ]=++cnt;
26     }
27     for (i=1; i<=n; i++) rank[sa[i]]=i;
28 }
29 void Height()
30 {
31     int len, i;
32     for (len=0, i=1; i<=n; height[rank[i++]]=len)
33     {
34         if (len) len--;
35         if (rank[i]==1) continue;
36         for (; st[i+len]==st[sa[rank[i]-1]+len]; len++);
37     }
38 }

```

2 数学

2.1 线性基

```

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

```

```

2  using namespace std;
3  #define B 30
4  #define N 10050
5
6  const int allset=(1<<B)-1;
7
8  int a[N];
9
10 struct LB
11 {
12     int mat[B],cnt;
13     multiset<int> st;
14     LB(){}
15     void clear()
16     {
17         st.clear();
18         cnt=0;
19         memset(mat,0,sizeof(mat));
20     }
21     void add(int x)
22     {
23         int i,j;
24         for (i=B-1;i>=0;i--)
25             if ((x>>i)&1)
26             {
27                 if (mat[i] x^=mat[i]; else
28                 {
29                     cnt++;
30                     mat[i]=x;
31                     break;
32                 }
33             }
34     }
35     void fix()
36     {
37         int i,j;
38         for (i=0;i<B;i++)
39             if (mat[i])
40             {
41                 for (j=i+1;j<B;j++)
42                     if ((mat[j]>>i)&1) mat[j]^=mat[i];
43             }
44     }
45     void preset()//正确性待定
46     {
47         int i;
48         fix();
49         for (i=0;i<B;i++)
50             if (mat[i]) st.insert(mat[i]);
51     }
52     int kth(int k)//正确性待定
53     {
54         int i,ans;
55         multiset<int>::iterator it;

```

```

56         if (k<=0||k>(1<<cnt)-1) return 0;//无解
57         for (ans=i=0,it=st.begin();it!=st.end();it++,i++)
58             if ((k>>i)&1) ans^=(*it);
59         return ans;
60     }
61     int getmax()
62     {
63         int i,ans;
64         fix();
65         ans=0;
66         for (i=B-1;i>=0;i--)
67             if (ans^mat[i]>ans) ans^=mat[i];
68         return ans;
69     }
70 } tree[N*10];

```

2.2 类欧几里得

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

```

1  #include<bits/stdc++.h>
2  using namespace std;
3  #define pb push_back
4  typedef long long ll;
5  const int MOD=1e9+7;
6  inline ll qp(ll a,ll b)
7  {
8      ll x=1; a%=MOD;
9      while(b)
10     {
11         if(b&1) x=x*a%MOD;
12         a=a*a%MOD; b>>=1;
13     }
14     return x;
15 }
16 namespace Lagrange {
17     ll x[23333],y[23333],a[23333],g[23333],h[23333],p[23333]; int N;
18     void work()
19     {
20         for(int i=0;i<N;++i) a[i]=0;
21         g[0]=1;
22         for(int i=0;i<N;++i)
23         {
24             for(int _=0;_<=i;+_ )
25                 h[_+1]=g[_]; h[0]=0;
26             for(int _=0;_<=i;+_ )
27                 h[_]=(h[_]-g[_]*(ll)x[i])%MOD;
28             for(int _=0;_<=i+1;+_ ) g[_]=h[_];
29         }
30         for(int i=0;i<N;++i)
31         {
32             for(int j=0;j<=N;++j) p[j]=g[j];
33             for(int j=N;j-->0)

```

```

34         p[j-1]=(p[j-1]+p[j]*(ll)x[i])%MOD;
35         ll s=1;
36         for(int j=0;j<N;++j) if(i!=j)
37             s=s*(x[i]-x[j])%MOD;
38         s=y[i]*qp(s,MOD-2)%MOD;
39         for(int _=0;_<N;++_)
40             a[_]=(a[_]+p[_+1]*s)%MOD;
41     }
42 }
43 vector<int> feed(vector<int> v)
44 {
45     N=v.size();
46     for(int i=0;i<N;++i) x[i]=i,y[i]=v[i];
47     work(); v.clear();
48     for(int i=0;i<N;++i) v.pb(a[i]);
49     while(v.size()&&!v.back()) v.pop_back();
50     return v;
51 }
52 ll calc(vector<int>&v, ll xx)
53 {
54     ll s=0,gg=1; xx%=MOD;
55     for(int i=0;i<N;++i)
56         s=(s+gg*v[i])%MOD, gg=gg*xx%MOD;
57     return s;
58 }
59 }
60 using Lagrange::feed;
61 using Lagrange::calc;
62 //ps[k]=\sum_{i=0}^x i^k
63 vector<int> ps[2333];
64 //rs[k]=\sum_{i=0}^x ((i+1)^k-i^k)
65 vector<int> rs[2333];
66 struct arr{ll p[11][11];};
67 ll C[233][233];
68 arr calc(ll a, ll b, ll c, ll n)
69 {
70     arr w;
71     if(n==0) a=0;
72     if(a==0||a*n+b<c)
73     {
74         for(int i=0;i<=10;++i)
75         {
76             ll t=calc(ps[i],n),s=b/c;
77             for(int j=0;i+j<=10;++j)
78                 w.p[i][j]=t,t=t*s%MOD;
79         }
80         return w;
81     }
82     for(int i=0;i<=10;++i)
83         w.p[i][0]=calc(ps[i],n);
84     if(a>=c||b>=c)
85     {
86         arr t=calc(a%c,b%c,c,n);
87         ll p=a/c,q=b/c;

```

```

88         for (int i=0; i<=10; ++i)
89             for (int j=1; i+j<=10; ++j)
90                 {
91                     ll s=0, px=1;
92                     for (int x=0; x<=j; ++x, px=px*p%MOD)
93                         {
94                             ll qy=1;
95                             for (int y=0; x+y<=j; ++y, qy=qy*q%MOD)
96                                 {
97                                     //  $x^i (px)^x q^y \dots^{j-x-y}$ 
98                                     s+=px*qy%MOD*C[j][x]%MOD*C[j-x][y]
99                                     %MOD*t.p[i+x][j-x-y]; s%=MOD;
100                                 }
101                             }
102                     w.p[i][j]=s;
103                 }
104         return w;
105     }
106     ll m=(a*n+b)/c;
107     arr t=calc(c, c-b-1, a, m-1);
108     for (int i=0; i<=10; ++i)
109         for (int j=1; i+j<=10; ++j)
110             {
111                 ll s=calc(rs[j], m-1)*calc(ps[i], n)%MOD;
112                 for (int p=0; p<j; ++p)
113                     {
114                         for (unsigned q=0; q<ps[i].size(); ++q)
115                             {
116                                 ll v=C[j][p]*ps[i][q]%MOD;
117                                 //  $v * t^p * ((tc+c-b-1)/a)^q$ 
118                                 s-=t.p[p][q]*v; s%=MOD;
119                             }
120                     }
121                 w.p[i][j]=s%MOD;
122             }
123     return w;
124 }
125 int T, n, a, b, c, k1, k2;
126 int main()
127 {
128     freopen("a.in", "r", stdin);
129     freopen("a.out", "w", stdout);
130     for (int i=0; i<=230; ++i)
131         {
132             C[i][0]=1;
133             for (int j=1; j<=i; ++j)
134                 C[i][j]=(C[i-1][j-1]+C[i-1][j])%MOD;
135         }
136     for (int i=0; i<=10; ++i)
137         {
138             ll sp=0, sr=0; vector<int> p, r;
139             for (int j=0; j<=20; ++j)
140                 sp+=qp(j, i), sr+=qp(j+1, i)-qp(j, i),
141                 sp%=MOD, sr%=MOD, p.pb(sp), r.pb(sr);

```

```

142         ps[i]=feed(p); rs[i]=feed(r);
143     }
144     scanf("%d",&T);
145     while(T--)
146     {
147         scanf("%d%d%d%d%d%d",
148             &n,&a,&b,&c,&k1,&k2);
149         arr s=calc(a,b,c,n);
150         int p=s.p[k1][k2];
151         p=(p%MOD+MOD)%MOD;
152         printf("%d\n",p);
153     }
154 }

```

2.3 高斯消元法实数方程

```

1 void Gauss(int n,int m)
2 {
3     int i,j,k,t;
4     double mul;
5     for (i=j=1;i<=n&&j<=m;i++,j++)
6     {
7         for (k=i+1;k<=n;k++)
8             if (abs(mat[k][j])>abs(mat[i][j]))
9                 for (t=1;t<=m+1;t++) swap(mat[i][t],mat[k][t]);
10            if (abs(mat[i][j])<eps)
11            {
12                i--;
13                continue;
14            }
15            for (k=i+1;k<=n;k++)
16            {
17                mul=mat[k][j]/mat[i][j];
18                for (t=1;t<=m+1;t++) mat[k][t]-=mat[i][t]*mul;
19            }
20        }
21    for (i=n;i>=1;i--)//表示那个变量是否确定 solved
22    {
23        for (j=1;j<=m;j++)
24            if (abs(mat[i][j])>eps) break;
25        if (j>m) continue;
26        solved[j]=true;
27        ans[j]=mat[i][m+1];
28        for (k=j+1;k<=m;k++)
29            if (abs(mat[i][k])>eps&&!solved[k]) solved[j]=false;
30        for (k=j+1;k<=m;k++) ans[j]-=ans[k]*mat[i][k];
31        ans[j]/=mat[i][j];
32    }
33 }

```

2.4 高斯消元法模方程


```

1 long long Pow(long long x,long long y)
2 {
3     if (y==0) return 1;
4     long long t=Pow(x,y/2);
5     if (y&1) return t*t%mod*x%mod;
6     return t*t%mod;
7 }
8 void Gauss(long long n,long long m)
9 {
10 long long i,j,k,t,lcm,muli,mulk;
11 for (i=j=1;i<=n&&j<=m;i++,j++)
12 {
13     for (k=i;k<=n;k++)
14     if (mat[k][j])
15     {
16         for (t=1;t<=m+1;t++) swap(mat[k][t],mat[i][t]);
17         break;
18     }
19     if (mat[i][j]==0)
20     {
21         i--;
22         continue;
23     }
24     for (k=i+1;k<=n;k++)
25     if (mat[k][j])
26     {
27         lcm=mat[k][j]*mat[i][j]/Gcd(mat[k][j],mat[i][j]);
28         muli=lcm/mat[i][j];
29         mulk=lcm/mat[k][j];
30         for (t=1;t<=m+1;t++)
31         {
32             mat[k][t]=mat[k][t]*mulk-mat[i][t]*muli;
33             mat[k][t]=(mat[k][t]%mod+mod)%mod;
34         }
35     }
36 }
37 for (i=n;i>=1;i--)
38 {
39     for (j=1;j<=m;j++)
40     if (mat[i][j]) break;
41     if (j>m) continue;
42     ans[j]=mat[i][m+1];
43     for (k=j+1;k<=m;k++) ans[j]-=ans[k]*mat[i][k];
44     ans[j]=(ans[j]*Pow(mat[i][j],mod-2)%mod+mod)%mod;
45 }
46 }

```

2.5 扩展欧几里得

求一组整数解且防爆

```

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

```

```

2 {
3 long long xx, yy, temp, gcd;
4     if (B == 0)
5     {
6         x = C;
7         y = 0;
8     } else
9     {
10        GCDEX(B, xx, A % B, yy, C);
11        x = yy;
12        temp = 1 - x / C * A;
13        gcd = GCD(temp, B);
14        temp /= gcd;
15        B /= gcd;
16        y = C / B * temp;
17    }
18    return;
19 }

```

2.6 FWT异或, 与, 或

```

1 void fwtXor(int* a, int len) {
2     if(len == 1) return;
3     int h = len >> 1;
4     fwtXor(a, h);
5     fwtXor(a + h, h);
6     for(int i = 0; i < h; ++i) {
7         int x1 = a[i];
8         int x2 = a[i + h];
9         a[i] = (x1 + x2) % mod;
10        a[i + h] = (x1 - x2 + mod) % mod;
11    }
12 }
13 void ifwtXor(int* a, int len) {
14     if(len == 1) return;
15     int h = len >> 1;
16     for(int i = 0; i < h; ++i) {
17         int y1 = a[i];
18         int y2 = a[i + h];
19         a[i] = 1ll*(y1 + y2) * div2 % mod;
20         a[i + h] = 1ll*(y1 - y2 + mod) * div2 % mod;
21    }
22     ifwtXor(a, h);
23     ifwtXor(a + h, h);
24 }
25 void fwtAnd(int* a, int len) {
26     if(len == 1) return;
27     int h = len >> 1;
28     fwtAnd(a, h);
29     fwtAnd(a + h, h);
30     for(int i = 0; i < h; ++i) {
31         int x1 = a[i];

```

```

32         int x2 = a[i + h];
33         a[i] = (x1 + x2) % mod;
34         a[i + h] = x2;
35     }
36 }
37 void ifwtAnd(int* a, int len) {
38     if(len == 1) return;
39     int h = len >> 1;
40     for(int i = 0; i < h; ++i) {
41         int y1 = a[i];
42         int y2 = a[i + h];
43         a[i] = (y1 - y2 + mod) % mod;
44         a[i + h] = y2;
45     }
46     ifwtAnd(a, h);
47     ifwtAnd(a + h, h);
48 }
49 void fwtOr(int* a, int len) {
50     if(len == 1) return;
51     int h = len >> 1;
52     fwtOr(a, h);
53     fwtOr(a + h, h);
54     for(int i = 0; i < h; ++i) {
55         int x1 = a[i];
56         int x2 = a[i + h];
57         a[i] = x1;
58         a[i + h] = (x1 + x2) % mod;
59     }
60 }
61 void ifwtOr(int* a, int len) {
62     if(len == 1) return;
63     int h = len >> 1;
64     for(int i = 0; i < h; ++i) {
65         int y1 = a[i];
66         int y2 = a[i + h];
67         a[i] = y1;
68         a[i + h] = (y2 - y1 + mod) % mod;
69     }
70     ifwtOr(a, h);
71     ifwtOr(a + h, h);
72 }

```

3 图论

3.1 tarjan

3.1.1 有向图强连通分量

```

1 void DFS(int x)
2 {
3     int pt, next;
4     dfn[x] = low[x] = ++times;
5     sk[++tp] = x;

```

```

6         instack[x]=true;
7         for (pt=first[x]; pt; pt=e[pt].next)
8         {
9             next=e[pt].to;
10            if (!dfn[next])
11            {
12                DFS(next);
13                low[x]=min(low[x], low[next]);
14            } else
15            if (instack[next]) low[x]=min(low[x], dfn[next]);
16        }
17        if (low[x]==dfn[x])
18        {
19            tot++;
20            for (; tp;)
21            {
22                instack[sk[tp]]=false;
23                belong[sk[tp--]]=tot;
24                if (sk[tp+1]==x) break;
25            }
26        }
27    }
28    for (i=1; i<=n; i++)
29    if (!dfn[i]) DFS(i);

```

3.1.2 点双联通分量

```

1 void DFS(int x, int fa)
2 {
3     int pt, next;
4     vis[x]=true;
5     dfn[x]=low[x]=++times;
6     sk[++tp]=x;
7     for (pt=first[x]; pt; pt=e[pt].next)
8     {
9         next=e[pt].to;
10        if (e[pt].id==fa) continue;
11        if (!vis[next])
12        {
13            DFS(next, e[pt].id);
14            low[x]=min(low[x], low[next]);
15            if (low[next]>=dfn[x])
16            {
17                tot++;
18                vec[tot].clear();
19                for (; tp;)
20                {
21                    vec[tot].push_back(sk[tp]);
22                    tp--;
23                    if (sk[tp+1]==next) break;
24                }
25                vec[tot].push_back(x);
26            }

```

```

27         }    else
28         if   (dfn[next]>last) low[x]=min(low[x],dfn[next]);
29     }
30 }
31 for (i=1;i<=n;i++)
32 if   (!vis[i])
33 {
34     DFS(i,0);
35     last=times;
36     if   (tp)
37     {
38         tot++;
39         vec[tot].clear();
40         for (i=1;i<=tp;i++) vec[tot].push_back(sk[i]);
41         tp=0;
42     }
43 }

```

3.1.3 边双联通分量

```

1 void DFS(int x,int fa)
2 {
3     int pt,next;
4     vis[x]=true;
5     dfn[x]=low[x]=++times;
6     sk[++tp]=x;
7     for (pt=first[x];pt;pt=e[pt].next)
8     {
9         next=e[pt].to;
10        if (e[pt].id==fa) continue;
11        if (!vis[next])
12        {
13            DFS(next,e[pt].id);
14            low[x]=min(low[x],low[next]);
15            if (low[next]>dfn[x])
16            {
17                tot++;
18                vec[tot].clear();
19                for (;tp;)
20                {
21                    vec[tot].push_back(sk[tp]);
22                    tp--;
23                    if (sk[tp+1]==next) break;
24                }
25            }
26        }    else
27        if   (dfn[next]>last) low[x]=min(low[x],dfn[next]);
28    }
29 }
30 for (i=1;i<=n;i++)
31 if   (!vis[i])
32 {
33     DFS(i,0);

```

```

34     last=times;
35     if (tp)
36     {
37         tot++;
38         vec[tot].clear();
39         for (i=1;i<=tp;i++) vec[tot].push_back(sk[i]);
40         tp=0;
41     }
42 }

```

3.2 网络流Dinic

```

1  struct Edge
2  {
3      int to, flow, next;
4  } e[1000050];
5  void Add(int x, int y, int z)
6  {
7      e[++now].to=y;
8      e[now].flow=z;
9      e[now].next=first[x];
10     first[x]=now;
11 }
12 bool Level()
13 {
14     int low, high, pt, next;
15     memset(level, -1, sizeof(level));
16     q[low=high=0]=S;
17     level[S]=0;
18     for (; low<=high; low++)
19         for (pt=first[q[low]]; pt!=-1; pt=e[pt].next)
20             {
21                 next=e[pt].to;
22                 if (level[next]!=-1||e[pt].flow<=0) continue;
23                 level[next]=level[q[low]]+1;
24                 q[++high]=next;
25                 if (next==T) return true;
26             }
27     return false;
28 }
29 int Find(int x, int delta)
30 {
31     int pt, next, temp, res=0;
32     if (x==T||delta<=0) return delta;
33     for (pt=last[x]; pt!=-1; last[x]=pt=pt.next)
34         {
35             next=e[pt].to;
36             if (level[next]!=level[x]+1) continue;
37             temp=Find(next, min(delta, e[pt].flow));
38             delta-=temp;
39             res+=temp;
40             e[pt].flow-=temp;

```

```

41         e[pt ^ 1].flow+=temp;
42         if (delta<=0) return res;
43     }
44     return res;
45 }

```

3.3 费用流

```

1  struct Edge
2  {
3      int to,flow , cost , next;
4  } e[1000050];
5  void Add(int x,int y,int flow ,int cost)
6  {
7      e[++now].to=y;
8      e[now].flow=flow;
9      e[now].cost=cost;
10     e[now].next=first [x];
11     first [x]=now;
12 }
13 bool Find()
14 {
15     int low , high , pt , next;
16     memset(inq , false , sizeof(inq));
17     memset(f , 127 , sizeof(f));
18     q[low=high=0]=S;
19     f[S]=0;
20     inq[S]=true;
21     for (;low<=high;inq[q[low++]]=false)
22         for (pt=first [q[low]];pt!=-1;pt=e[pt].next)
23             {
24                 next=e[pt].to;
25                 if (f[next]<=f[q[low]]+e[pt].cost || e[pt].flow <=0||q[low]
26                     ]==T||next==S) continue;
27                 f[next]=f[q[low]]+e[pt].cost;
28                 path[next]=q[low];
29                 number[next]=pt;
30                 if (!inq[next])
31                     {
32                         inq[next]=true;
33                         q[++high]=next;
34                     }
35             }
36     return f[T]<999999999;
37 }
38 void Addflow()
39 {
40     int t=999999999,i;
41     for (i=T;i!=S;i=path[i]) t=min(t,e[number[i]].flow);
42     mincost+=t*f[T];
43     maxflow+=t;
44     for (i=T;i!=S;i=path[i])

```

```

44     {
45         e[number[i]].flow-=t;
46         e[number[i]^1].flow+=t;
47     }
48 }

```

4 小算法

4.1 跨平台大随机数

```

1  int Rand()
2  {
3      int ra = rand() % 32768;
4      int rb = rand() % 32768;
5      return (ra << 15) | rb;
6  }

```

4.2 读入优化

```

1  char *st,*nd,ch[1000050];
2  inline char Get()
3  {
4      if (st==nd)
5      {
6          st=ch;
7          nd=st+fread(ch,1,1000007,stdin);
8      }
9      return *(st++);
10 }
11 inline int Read()
12 {
13     register int t=0;
14     register char c=Get();
15     register bool fu=false;
16     for (;c!='-'&&(c<'0' || '9'<c);c=Get());
17     if (c=='-')
18     {
19         fu=true;
20         c=Get();
21     }
22     for (;'0'<=c&&c<='9';c=Get()) t=10*t+c-'0';
23     if (fu) return -t;
24     return t;
25 }

```

4.3 KMP算法

```

1      len = strlen(st);

```



```

2      memset(prepare, -1, sizeof(prepare));
3      memset(number, 0, sizeof(number));
4      for (i = 1; i < len; i++)
5      {
6          for (j = i - 1; j != -1 && st[prepare[j] + 1] != st[i]; j
              = prepare[j]);
7          prepare[i] = j != -1 ? prepare[j] + 1 : -1;
8      }

```

4.4 扩展KMP

```

1      extend[1] = number;
2      for (far = 1; far < number; far++)
3      if (tested[far] != tested[far + 1])
4      {
5          break;
6      }
7      extend[source = 2] = far - 1;
8      for (j = 3; j <= number; j++)
9      {
10         temp = extend[j - source + 1];
11         if (j + temp - 1 < far)
12         {
13             extend[j] = temp;
14         } else
15         {
16             for (k = max(far + 1, j); k <= number; k++)
17             if (tested[k] != tested[k - j + 1])
18             {
19                 break;
20             }
21             far = k - 1;
22             source = j;
23             extend[j] = far - source + 1;
24         }
25     }

```

5 计算几何

5.1 凸包

```

1  bool cmp(const Point &a, const Point &b)
2  {
3      return F(a.x-b.x) < 0 || (F(a.x-b.x) == 0 && a.y < b.y);
4  }
5  void Gram(int id[], int n)
6  {
7      int i, mid;
8      sort(id, id+n, cmp);
9      tp=0;

```

```

10 //凸包从最小的点出发, 逆时针方向x
11 for (i=0;i<n;i++)
12 {
13     for (;tp>=2&&Cross(p[sk[tp-1]]-p[sk[tp-2]],p[id[i]]-p[sk[tp-1]])<=0;tp--);
14     //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
15     sk[tp++]=id[i];
16 }
17 mid=tp;
18 for (i=n-2;i>=0;i--)
19 {
20     for (;tp>mid&&Cross(p[sk[tp-1]]-p[sk[tp-2]],p[id[i]]-p[sk[tp-1]])<=0;tp--);
21     //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
22     sk[tp++]=id[i];
23 }
24 if (n>1) tp--;
25 }

```

5.2 定义

```

1 struct Point
2 {
3     double x,y;
4     Point() {}
5     Point(double _x,double _y):x(_x),y(_y) {}
6 };
7 struct Seg
8 {
9     Point a,b;
10    Seg() {}
11    Seg(Point _a,Point _b):a(_a),b(_b) {}
12 };
13 struct Circle
14 {
15     double x,y,r;
16     Point pt() {return Point(x,y);}
17     double Area() {return pi*r*r;}
18 };
19 Point operator +(const Point &a,const Point &b)
20 {
21     return Point(a.x+b.x,a.y+b.y);
22 }
23 Point operator -(const Point &a,const Point &b)
24 {
25     return Point(a.x-b.x,a.y-b.y);
26 }
27 Point operator *(const Point &a,double b)
28 {
29     return Point(a.x*b,a.y*b);
30 }

```

```

31 Point operator /(const Point &a, double b)
32 {
33     return Point(a.x/b, a.y/b);
34 }
35 int F(double x)
36 {
37     if (x>eps) return 1;
38     if (x<-eps) return -1;
39     return 0;
40 }
41 bool operator ==(const Point &a, const Point &b)
42 {
43     return F(a.x-b.x)==0&&F(a.y-b.y)==0;
44 }
45 double Dist(const Point &a)
46 {
47     return sqrt(a.x*a.x+a.y*a.y);
48 }
49 double Dot(const Point &a, const Point &b)
50 {
51     return a.x*b.x+a.y*b.y;
52 }
53 double Cross(const Point &a, const Point &b)
54 {
55     return a.x*b.y-a.y*b.x;
56 }
57 Point Rotate(const Point &p, double a) // 逆时针旋转
58 {
59     return Point(p.x*cos(a)-p.y*sin(a), p.x*sin(a)+p.y*cos(a));
60 }
61 Point Inter(Seg a, Seg b) // 两线段相交（前提有交点）
62 {
63     double s=Cross(a.b-a.a, b.a-a.a), t=Cross(a.b-a.a, b.b-a.a);
64     return b.a+(b.b-b.a)*s/(s-t);
65 }
66 vector<Point> SegCir(Seg seg, Point pt, double r) // 线圆
67 {
68     vector<Point> ans;
69     double mul;
70     Point vec, mid;
71     ans.clear();
72     vec=Rotate(seg.b-seg.a, pi/2);
73     mid=Inter(seg, Seg(pt, pt+vec));
74     if (F(Dist(pt-mid)-r)>0) return ans;
75     if (F(Dist(pt-mid)-r)==0)
76     {
77         ans.push_back(mid);
78         ans.push_back(mid);
79         return ans;
80     }
81     vec=seg.b-seg.a;
82     mul=sqrt(r*r-Dist2(mid-pt))/Dist(vec);
83     ans.push_back(mid+vec*mul);
84     ans.push_back(mid-vec*mul);

```

```

85         return ans;
86     }
87     vector<Point> Circir(Circle a,Circle b)//圆圆相交
88     {
89         vector<Point> ans;
90         double dis,dis2,alpha;
91         Point pa,pb,vec;
92         ans.clear();
93         if (a.r<b.r) swap(a,b);
94         pa=a.pt();
95         pb=b.pt();
96         vec=pb-pa;
97         dis=Dist(vec);
98         dis2=Dist2(vec);
99         if (F(dis-(a.r+b.r))>0||F(dis-(a.r-b.r))<0) return ans;
100        if (F(dis-(a.r+b.r))==0)
101        {
102            ans.push_back(pa+vec*a.r/(a.r+b.r));
103            return ans;
104        }
105        if (F(dis-(a.r-b.r))==0)
106        {
107            ans.push_back(pa+vec*a.r/(a.r-b.r));
108            return ans;
109        }
110        alpha=acos((a.r*a.r+dis2-b.r*b.r)/2/a.r/dis);
111        ans.push_back(pa+Rotate(vec,alpha)*a.r/dis);
112        ans.push_back(pa+Rotate(vec,-alpha)*a.r/dis);
113        return ans;
114    }
115    double Bing(double ra,double rb,double dis)
116    {
117        double alpha,beta;
118        if (ra<rb) swap(ra,rb);
119        if (F(dis-(ra-rb))<=0) return pi*ra*ra;
120        if (F(dis-(ra+rb))>=0) return pi*ra*ra+pi*rb*rb;
121        alpha=acos((ra*ra+dis*dis-rb*rb)/2/dis/ra);
122        beta=acos((rb*rb+dis*dis-ra*ra)/2/dis/rb);
123        return (pi-alpha)*ra*ra+(pi-beta)*rb*rb+ra*dis*sin(alpha);
124    }
125
126    double Jiao(double ra,double rb,double dis)
127    {
128        return pi*ra*ra+pi*rb*rb-Bing(ra,rb,dis);
129    }
130
131    Point Gongmid(Circle a,Circle b)//正确性待定
132    {
133        Point pa=a.pt(),pb=b.pt();
134        return pa+(pb-pa)*a.r/(a.r+b.r);
135    }
136
137    Point Gongright(Circle a,Circle b)
138    {

```

```

139 Point pa=a.pt(),pb=b.pt();
140     return pa+(pb-pa)*a.r/(a.r-b.r);
141 }
142
143 int Ptinpol(Point pt)
144 {
145     int i,k,d1,d2,wn=0;
146     for(i=0;i<n;i++)
147     {
148         if(Ins(pt,Seg(p[i],p[(i+1)%n]))) return 2;
149         k=F(Cross(p[(i+1)%n]-p[i],pt-p[i]));
150         d1=F(p[i].y-pt.y);
151         d2=F(p[(i+1)%n].y-pt.y);
152         if(k>0&&d1<=0&&d2>0)wn++;
153         if(k<0&&d2<=0&&d1>0)wn--;
154     }
155     return wn!=0;
156 }
157 bool Cirinpol(Point pt)//需要点在多边形内的前提
158 {
159     int i;
160     double nearest;
161     nearest=1e+100;
162     for(i=0;i<n;i++)
163     {
164         nearest=min(nearest,Dist(p[i]-pt));
165         if(F(Dot(pt-p[i],p[(i+1)%n]-p[i]))>0&&
166            F(Dot(pt-p[(i+1)%n],p[i]-p[(i+1)%n]))>0)
167             nearest=min(nearest,abs(Cross(p[i]-pt,p[(i+1)%n]-pt))/
168                           dis[i]);
169     }
170     return F(nearest-r)>=0;
171 }
172 bool Ins(const Point &p,const Seg &s)
173 {
174     return F(Cross(s.a-p,s.b-p))==0&&
175            F(p.x-min(s.a.x,s.b.x))>=0&&
176            F(p.x-max(s.a.x,s.b.x))<=0&&
177            F(p.y-min(s.a.y,s.b.y))>=0&&
178            F(p.y-max(s.a.y,s.b.y))<=0;
179 }
180 double PS(const Point &p,const Seg &s) 点到线段最短距离
181 {
182     if(F(Dot(p-s.a,s.b-s.a))<0||F(Dot(p-s.b,s.a-s.b))<0) return
183         min(Dist(p-s.a),Dist(p-s.b));
184     return abs(Cross(s.a-p,s.b-p))/Dist(s.a-s.b);
185 }
186 double SS(const Seg &a,const Seg &b) 线段到线段最短距离
187 {
188     return min(min(PS(a.a,b),PS(a.b,b)),min(PS(b.a,a),PS(b.b,a)));
189 }
190 double Alpha(Point a,Point b)
191 {
192     double ans;

```

```

191     ans=atan2(b.y,b.x)-atan2(a.y,a.x);
192     if (ans<0) ans=-ans;
193     if (ans>pi) ans=2*pi-ans;
194     return ans;
195 }
196 double Shan(Circle c,double a)
197 {
198     return c.r*c.r*a/2;
199 }

```

5.3 半平面交

```

1
2 bool Cmphp(Seg a,Seg b)
3 {
4     Point va,vb;
5     double dega,degb;
6     va=a.b-a.a;
7     vb=b.b-b.a;
8     dega=atan2(va.y,va.x);
9     degb=atan2(vb.y,vb.x);
10    return F(dega-degb)<0||F(dega-degb)==0&&Cross(a.b-a.a,b.a-a.a)
        <0;
11 }
12 void HalfPlane(Seg hp[],int n,Point pol[],int &pols)
13 {
14     int tp,i,low,high;
15     Point mid;
16     hp[n++]=Seg(Point(-oo,-oo),Point(oo,-oo));
17     hp[n++]=Seg(Point(oo,-oo),Point(oo,oo));
18     hp[n++]=Seg(Point(oo,oo),Point(-oo,oo));
19     hp[n++]=Seg(Point(-oo,oo),Point(-oo,-oo));
20     sort(hp,hp+n,Cmphp);
21     tp=0;//sk 0~tp-1
22     low=0;
23     high=-1;
24     for (i=0;i<n;i++)
25     if (high-low+1==0||F(Cross(sk[high].b-sk[high].a,hp[i].b-hp[i].
        a)))
26     {
27         for (;low<high;high--)
28         {
29             mid=Inter(sk[high],sk[high-1]);
30             if (F(Cross(hp[i].b-hp[i].a,mid-hp[i].a))>0) break;
31         }
32         for (;low<high;low++)
33         {
34             mid=Inter(sk[low],sk[low+1]);
35             if (F(Cross(hp[i].b-hp[i].a,mid-hp[i].a))>0) break;
36         }
37         sk[++high]=hp[i];
38     }

```

```

39     for (; low < high; high--)
40     {
41         mid = Inter(sk[high], sk[high-1]);
42         if (Cross(sk[low].b - sk[low].a, mid - sk[low].a) > 0) break;
43     }
44     tp = high - low + 1;
45     for (i = 0; i < tp; i++) sk[i] = sk[low + i];
46     pols = 0;
47     if (tp <= 2) return;
48     for (i = 0; i < tp; i++) pol[pols++] = Inter(sk[i], sk[(i+1)%tp]);
49 }

```

5.4 圆与多边形交集

```

1  double CT(Circle c, Point a, Point b) 圆与三角形交（多边形）
2  {
3      double da, db;
4      Seg s;
5      vector <Point> temp;
6      da = Dist(a - c.pt());
7      db = Dist(b - c.pt());
8      if (da > db)
9      {
10         swap(a, b);
11         swap(da, db);
12     }
13     s = Seg(a, b);
14     temp = CS(c, s);
15     if (F(db - c.r) <= 0) return 0.5 * abs(Cross(a - c.pt(), b - c.pt()));
16     if (F(da - c.r) < 0)
17     {
18         if (F(Dot(a - temp[1], b - temp[1])) < 0) swap(temp[0], temp[1]);
19         return Shan(c, Alpha(temp[0] - c.pt(), b - c.pt())) + 0.5 * abs(
20             Cross(a - c.pt(), temp[0] - c.pt()));
21     }
22     if (!temp.size()) return Shan(c, Alpha(a - c.pt(), b - c.pt()));
23     if (Ins(temp[1], s) && Dist2(a - temp[1]) < Dist2(a - temp[0])) swap(
24         temp[0], temp[1]);
25     if (Ins(temp[0], s) && Ins(temp[1], s))
26     {
27         return Shan(c, Alpha(a - c.pt(), temp[0] - c.pt())) +
28             Shan(c, Alpha(b - c.pt(), temp[1] - c.pt())) +
29             0.5 * abs(Cross(temp[0] - c.pt(), temp[1] - c.pt()));
30     }
31     return Shan(c, Alpha(a - c.pt(), b - c.pt()));
32 }

```

5.5 三角形面积并

```

1  #include <math.h>
2  #include <stdio.h>

```

```

3  #include<string.h>
4  #include<algorithm>
5  #define N 333
6  #define pr pair<ld,ld>
7  using namespace std;
8  typedef long double ld;
9  const ld EPS=1e-8;
10 const ld INF=1e100;
11 struct Point
12 {
13     ld x,y;
14     Point() {}
15     Point(ld _,ld _):x(_),y(_){}
16     void read()
17     {
18         double _ , --;
19         scanf("%lf%lf",&_,&--);
20         x=_ , y=--;
21     }
22     friend bool operator <(Point a,Point b)
23     {
24         if(fabs(a.x-b.x)<EPS)
25             return a.y<b.y;
26         return a.x<b.x;
27     }
28     friend Point operator +(Point a,Point b)
29     {
30         return Point(a.x+b.x,a.y+b.y);
31     }
32     friend Point operator -(Point a,Point b)
33     {
34         return Point(a.x-b.x,a.y-b.y);
35     }
36     friend Point operator *(ld a,Point b)
37     {
38         return Point(a*b.x,a*b.y);
39     }
40     friend ld operator *(Point a,Point b)
41     {
42         return a.x*b.x+a.y*b.y;
43     }
44     friend ld operator ^(Point a,Point b)
45     {
46         return a.x*b.y-a.y*b.x;
47     }
48 }a[N][3],Poi[N*N];
49 struct Line
50 {
51     Point p,v;
52     Line() {}
53     Line(Point _,Point _){p=_,v=-- --;}
54     Point operator [](int k)
55     {
56         if(k) return p+v;

```



```

57         else    return p;
58     }
59     friend bool Cross(Line a,Line b)
60     {
61         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;
62     }
63     friend Point getP(Line a,Line b)
64     {
65         Point u=a.p-b.p;
66         ld temp=(b.v^u)/(a.v^b.v);
67         return a.p+temp*a.v;
68     }
69 }l[N][3],T;
70 pr p[N];
71 int main()
72 {
73     int n,m,i,j,k,x,y,cnt,tot;
74     ld ans,last,A,B,sum;
75     scanf("%d",&n);
76     for(i=1,tot=0;i<=n;i++)
77     {
78         a[i][0].read(),a[i][1].read(),a[i][2].read();
79         Poi[++tot]=a[i][0],Poi[++tot]=a[i][1],Poi[++tot]=a[i][2];
80         sort(a[i],a[i]+3);
81         if((a[i][2]-a[i][0]^a[i][1]-a[i][0])>EPS)
82             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]);
83         else
84             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]);
85     }
86     for(i=1;i<=n;i++)
87     {
88         for(j=1;j<i;j++)
89         {
90             for(x=0;x<3;x++)
91             for(y=0;y<3;y++)
92             {
93                 if(Cross(l[i][x],l[j][y]))
94                     Poi[++tot]=getP(l[i][x],l[j][y]);
95             }
96         }
97     }
98     sort(Poi+1,Poi+tot+1);
99     ans=0,last=Poi[1].x;
100     T=Line(Point(0,-INF),Point(0,INF));
101     for(i=2;i<=tot;i++)
102     {
103         T.p.x=(last+Poi[i].x)/2;
104         for(j=1,cnt=0;j<=n;j++)
105         {
106             if(Cross(l[j][0],T))
107             {

```

```

108         A=getP(l[j][0],T).y;
109         if(Cross(l[j][1],T))
110             B=getP(l[j][1],T).y;
111         else
112             B=getP(l[j][2],T).y;
113         if(A>B) swap(A,B);
114         p[++cnt]=pr(A,B);
115     }
116 }
117 sort(p+1,p+cnt+1);
118 for(j=1,sum=0,A=-INF;j<=cnt;j++)
119 {
120     if(p[j].first>A)
121     {
122         sum+=p[j].second-p[j].first;
123         A=p[j].second;
124     }
125     else
126     {
127         if(p[j].second>A)
128             sum+=p[j].second-A,A=p[j].second;
129     }
130 }
131 ans+=(Poi[i].x-last)*sum;
132 last=Poi[i].x;
133 }
134 printf("%.2lf\n", (double)ans);
135 return 0;
136 }

```

5.6 K圆并

```

1  #include <cstdio>
2  #include <cstdlib>
3  #include <climits>
4  #include <iostream>
5  #include <algorithm>
6  #include <cstring>
7  #include <string>
8  #include <queue>
9  #include <map>
10 #include <vector>
11 #include <bitset>
12 #include <cmath>
13 #include <set>
14 #include <utility>
15 #include <ctime>
16 #define sqr(x) ((x)*(x))
17 using namespace std;
18
19 const int N = 1010;
20 const double eps = 1e-8;

```

```

21  const double pi = acos(-1.0);
22  double area[N];
23  int n;
24
25  int dcmp(double x) {
26      if (x < -eps) return -1; else return x > eps;
27  }
28
29  struct cp {
30      double x, y, r, angle;
31      int d;
32      cp() {}
33      cp(double xx, double yy, double ang = 0, int t = 0) {
34          x = xx; y = yy; angle = ang; d = t;
35      }
36      void get() {
37          scanf("%lf%lf%lf", &x, &y, &r);
38          d = 1;
39      }
40  } cir[N], tp[N * 2];
41
42  double dis(cp a, cp b) {
43      return sqrt(sqr(a.x - b.x) + sqr(a.y - b.y));
44  }
45
46  double cross(cp p0, cp p1, cp p2) {
47      return (p1.x - p0.x) * (p2.y - p0.y) - (p1.y - p0.y) * (p2.x - p0
        .x);
48  }
49
50  int CirCrossCir(cp p1, double r1, cp p2, double r2, cp &cp1, cp &cp2)
    {
51      double mx = p2.x - p1.x, sx = p2.x + p1.x, mx2 = mx * mx;
52      double my = p2.y - p1.y, sy = p2.y + p1.y, my2 = my * my;
53      double sq = mx2 + my2, d = -(sq - sqr(r1 - r2)) * (sq - sqr(r1 +
        r2));
54      if (d + eps < 0) return 0; if (d < eps) d = 0; else d = sqrt(d);
55      double x = mx * ((r1 + r2) * (r1 - r2) + mx * sx) + sx * my2;
56      double y = my * ((r1 + r2) * (r1 - r2) + my * sy) + sy * mx2;
57      double dx = mx * d, dy = my * d; sq *= 2;
58      cp1.x = (x - dy) / sq; cp1.y = (y + dx) / sq;
59      cp2.x = (x + dy) / sq; cp2.y = (y - dx) / sq;
60      if (d > eps) return 2; else return 1;
61  }
62
63  bool circmp(const cp& u, const cp& v) {
64      return dcmp(u.r - v.r) < 0;
65  }
66
67  bool cmp(const cp& u, const cp& v) {
68      if (dcmp(u.angle - v.angle)) return u.angle < v.angle;
69      return u.d > v.d;
70  }
71

```

```

72 double calc(cp cir, cp cp1, cp cp2) {
73     double ans = (cp2.angle - cp1.angle) * sqr(cir.r)
74     - cross(cir, cp1, cp2) + cross(cp(0, 0), cp1, cp2);
75     return ans / 2;
76 }
77
78 void CirUnion(cp cir[], int n) {
79     cp cp1, cp2;
80     sort(cir, cir + n, circmp);
81     for (int i = 0; i < n; ++i)
82         for (int j = i + 1; j < n; ++j)
83             if (dcmp(dis(cir[i], cir[j]) + cir[i].r - cir[j].r) <= 0)
84                 cir[i].d++;
85     for (int i = 0; i < n; ++i) {
86         int tn = 0, cnt = 0;
87         for (int j = 0; j < n; ++j) {
88             if (i == j) continue;
89             if (CirCrossCir(cir[i], cir[i].r, cir[j], cir[j].r,
90                             cp2, cp1) < 2) continue;
91             cp1.angle = atan2(cp1.y - cir[i].y, cp1.x - cir[i].x);
92             cp2.angle = atan2(cp2.y - cir[i].y, cp2.x - cir[i].x);
93             cp1.d = 1;    tp[tn++] = cp1;
94             cp2.d = -1;   tp[tn++] = cp2;
95             if (dcmp(cp1.angle - cp2.angle) > 0) cnt++;
96         }
97         tp[tn++] = cp(cir[i].x - cir[i].r, cir[i].y, pi, -cnt);
98         tp[tn++] = cp(cir[i].x + cir[i].r, cir[i].y, -pi, cnt);
99         sort(tp, tp + tn, cmp);
100        int p, s = cir[i].d + tp[0].d;
101        for (int j = 1; j < tn; ++j) {
102            p = s; s += tp[j].d;
103            area[p] += calc(cir[i], tp[j - 1], tp[j]);
104        }
105    }
106 }
107
108 void solve() {
109     for (int i = 0; i < n; ++i)
110         cir[i].get();
111     memset(area, 0, sizeof(area));
112     CirUnion(cir, n);
113     for (int i = 1; i <= n; ++i) {
114         area[i] -= area[i - 1];
115         printf("%d] = %.3lf\n", i, area[i]);
116     }
117 }
118
119 int main() {
120     freopen("a.in", "r", stdin);
121     while (scanf("%d", &n) != EOF) {
122         solve();
123     }
124     return 0;
125 }

```

6 三维计算几何

6.1 基本定义

```
1 Point Cross(Point a, Point b)
2 {
3     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);
4 }
5 double Crossxy(Point a, Point b)
6 {
7     return a.x*b.y-a.y*b.x;
8 }
9 vector<Point> SegPlane(Seg seg, Plane p)
10 {
11     double s, t;
12     Point fa;
13     vector<Point> ans;
14     ans.clear();
15     fa=Cross(p.b-p.a, p.c-p.a);
16     if (F(Dot(fa, seg.b-seg.a))==0) return ans;
17     s=Dot(p.a-seg.a, fa)/Dist(fa);
18     t=Dot(p.a-seg.b, fa)/Dist(fa);
19     ans.push_back(seg.a+(seg.b-seg.a)*s/(s-t));
20     return ans;
21 }
```

6.2 一些补充

```
1  \\ mixed product
2  double Mix(Point3 a, Point3 b, Point3 c)
3  {
4      return Dot(Cross(a, b), c);
5  }
6  \\ distance from point to plane
7  double PP(Point3 pt, Plane pl)
8  {
9      Point3 fa=Cross(pl.b-pl.a, pl.c-pl.a);
10     return abs(Dot(fa, pt-pl.a))/Dist(fa);
11 }
12 \\ get the center point from 3D(need plane well prepared)
13 Point3 Getcenter(Point3 p[], int n, Plane pp[], int nn)
14 {
15     int i;
16     double sumv, tempv;
17     Point3 sum;
18     sumv=0;
19     sum=Point3(0,0,0);
20     for (i=0; i<nn; i++)
21     {
22         tempv=Mix(pp[i].b-pp[i].a, pp[i].c-pp[i].a, Point3
23             (0,0,0)-pp[i].a);
24         sum=sum+(pp[i].a+pp[i].b+pp[i].c)*tempv/4.0;
25     }
```

```

24         sumv+=tempv;
25     }
26     return sum/sumv;
27 }

```

7 bitset用法

```

1  C++ bitset 用法
2
3
4  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
5
6
7
8
9  bitset 复制代码
10
11
12     bitset<4> bitset1      ;//无参构造，长度为 4，默认每一位为 0
13
14     bitset<8> bitset2(12)   ;//长度为 8，二进制保存，前面用 0 补充
15
16     string s = "100101";
17     bitset<10> bitset3(s)   ;//长度为，前面用 0 补充 10
18
19     char s2[] = "10101";
20     bitset<13> bitset4(s2)   ;//长度为，前面用 0 补充 13
21
22     cout << bitset1 << endl ;//0000
23     cout << bitset2 << endl ;//00001100
24     cout << bitset3 << endl ;//0000100101
25     cout << bitset4 << endl ;//0000000010101复制代码注意：用字符串构造
   时，字符串只能包含
26
27
28
29
30
31     '0' 或 '1'，否则会抛出异常。构造时，需在中表明
32
33     <>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
34
35     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
36
37
38     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
39
40     string s = "100101"    ;

```

```

41     bitset<4> bitset2(s)      ;//的 ssize，而=6的 bitsetSize，只取前面部分，
    即=41001
42
43     char s2[] = "11101";
44     bitset<4> bitset3(s2)    ;//与同理，只取前面部分，即 bitset21110
45
46     cout << bitset1 << endl    ;//00
47     cout << bitset2 << endl    ;//1001
48     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
49
50
51
52
53     bitset 复制代码
54
55
56     bitset<4> foo (string("1001"));
57     bitset<4> bar (string("0011"));
58
59     cout << (foo^=bar) << endl;    // 1010 (对按位异或后赋值
    给 foobarfoo)
60     cout << (foo&=bar) << endl;    // 0010 按位与后赋值给 (foo)
61     cout << (foo|=bar) << endl;    // 0011 按位或后赋值给 (foo)
62
63     cout << (foo<<=2) << endl;    // 1100 左移2位，低位补0，有自身赋
    值()
64     cout << (foo>>=1) << endl;    // 0110 右移1位，高位补0，有自身赋
    值()
65
66     cout << (~bar) << endl;        // 1100 按位取反()
67     cout << (bar<<1) << endl;    // 0110 左移，不赋值()
68     cout << (bar>>1) << endl;    // 0001 右移，不赋值()
69
70     cout << (foo==bar) << endl;    // false 为(0110==0011false)
71     cout << (foo!=bar) << endl;    // true 为 (0110!=0011true)
72
73     cout << (foo&bar) << endl;    // 0010 按位与，不赋值()
74     cout << (foo|bar) << endl;    // 0111 按位或，不赋值()
75     cout << (foo^bar) << endl;    // 0101 按位异或，不赋值()复制代码
    此外，可以通过
76
77     [ ] 访问元素类似数组()，注意最低位下标为0，如下：
78
79     bitset<4> foo ("1011");
80
81     cout << foo[0] << endl    ;//1
82     cout << foo[1] << endl    ;//1
83     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：
84
85
86
87
88

```

```

89  bitset复制代码
90
91
92      bitset<8> foo ("10011011");
93
94      cout << foo.count() << endl      ;//      (5函数用来求中的位数，中共有 5 个
          1 countbitset1foo
95      cout << foo.size() << endl      ; //      (8函数用来求的大小，一共有 8
          位 sizebitset
96
97      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 true test false true foo 为 1，返回 [0] true
98      cout << foo.test(2) << endl      ;//      (同理， false foo 为 0，返
          回 [2] false
99
100     cout << foo.any() << endl      ;//      (函数检查中是否有 1 true anybitset
101     cout << foo.none() << endl      ;//      (函数检查中是否没有
          1 false nonebitset
102     cout << foo.all() << endl      ;//      (函数检查中是全部为
          1 false allbitset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
103
104
105     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
106
107
108
109
110
111
112     bitset<8> foo ("10011011");
113
114     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 flip foo 反转，即 0 变 1，1 变 0 ""
115     cout << foo.flip() << endl      ; //      (01100000函数不指定参数时，将每
          一位全部取反 flipbitset
116
117     cout << foo.set() << endl      ;//      (11111111函数不指定参数时，将
          的每一位全部置为 1 setbitset
118     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
119     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
120
121     cout << foo.reset(4) << endl      ;//      (11101111函数传一个参数时将参
          数下标处置为 0 reset
122     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
123
124
125
126
127
128
129     bitset<8> foo ("10011011");
130

```



```

131     string s = foo.to_string()    ;//将转换成类型 bitsetstring
132     unsigned long a = foo.to_ulong()    ;//将转换成 bitsetunsigned 类
        型 long
133     unsigned long long b = foo.to_ullong()    ;//将转换
        成 bitsetunsigned long 类型 long
134
135     cout << s << endl    ;//10011011
136     cout << a << endl    ;//155
137     cout << b << endl    ;//155复制代码

```