Xplate

Xwind Wu

September 23, 2016

Contents

1 Tips

1.1 Debug

1.1.1 Linux

```
while true;
do
  ./gen > A.in
  ./run < A.in > A.out
  ./std < A.in > A.ans
  if diff A.out A.ans; then
    echo OK
  else
    echo WA
    break
  fi
done
```

1.1.2 Windows

```
:loop
gen >gen.in
* <gen.in >*.out
* <gen.in >*.ans
fc *.out *.ans
if not errorlevel 1 goto loop
pause
```

1.2 Stack

```
#pragma comment(linker,"/STACK:1024000000,1024000000")
```

1.3 Speeding Out

```
struct Tfai{
    static const int file=17000000;
    char s[file],*p;
    void build(){p=s;fread(s,1,file,stdin);}
    template<class Tsqy> inline void operator()(Tsqy &x){
    bool ok=false;while(*p<48 && *p!='-')++p;if(*p=='-')++p,ok=true;
    x=0;while(47<*p)x=x*10+*(p++)-48;if(ok)x=-x;
    }
}scan;
```

1.4 统计的库函数

```
int __builtin_popcount(i) 计算无符号 32 位整型二进制有几个 1
int __builtin_ffs (unsigned int x) 返回右起第一个‘1’的位置
int __builtin_clz (unsigned int x) 返回左起第一个‘1’之前 0 的个数
int __builtin_ctz (unsigned int x) 返回右起第一个‘1’之后的 0 的个数
此外，这些函数都有相应的 usigned long 和 usigned long long 版本，只需要在函数名后面
加上 l 或 ll 就可以了，比如int __builtin_clzll
```

1.5 Tips

string 的 erase 很慢，就算是 erase（s.begin()）也很慢
迷宫题注意判断起点与终点是不是在一起
计算连续小于某个值的区间，不能以大于那个值的数作为分割点计算区间长度，而应该边判断边加
线段树：更新非直接 lazy 区间要 tree[pos]+=num*(min(qr,r)-max(ql-1,mid));
　　注意 pos*2 不能越界，在叶子节点时
动规要求输出最小字典序时应该从右往左做。。。
定义多下标状态时，注意定义与使用时的下标顺序

2 Search

2.1 DLX 精确覆盖

```
const int maxv=1100000;
const int maxm=1005,maxn=1005;
struct DLX
{
        int n,m,sz;
        int L[maxv],R[maxv],U[maxv],D[maxv],Row[maxv],Col[maxv];
        int H[maxn],S[maxm];
        int ansd,ans[maxn];
        void init(int N,int M)
        {
                n=N;m=M;
                int i;
                for(i=0;i<=m;++i)
                {
                        S[i]=0;
                        U[i]=D[i]=i;
                        L[i]=(i+m)%(m+1);
                        R[i]=(i+1)%(m+1);
                }
                sz=m;
                for(i=1;i<=n;++i)H[i]=-1;
        }
        void Link(int r,int c)
        {
                ++S[Col[++sz]=c];
                Row[sz]=r;
                D[sz]=D[c];U[D[c]]=sz;
                U[sz]=c;D[c]=sz;
                if(H[r]<0)H[r]=L[sz]=R[sz]=sz;
                else
                {
                        R[sz]=R[H[r]];
                        L[R[H[r]]]=sz;
                        L[sz]=H[r];
                        R[H[r]]=sz;
                }
        }
        void remove(int c)
        {
```

```
40                  L[R[c]]=L[c];R[L[c]]=R[c];
41                  for(int i=D[c];i!=c;i=D[i])
42                          for(int j=R[i];j!=i;j=R[j])
43                          {
44                                  U[D[j]]=U[j];
45                                  D[U[j]]=D[j];
46                                  --S[Col[j]];
47                          }
48          }
49          void resume(int c)
50          {
51                  L[R[c]]=R[L[c]]=c;
52                  for(int i=U[c];i!=c;i=U[i])
53                          for(int j=L[i];j!=i;j=L[j])
54                                  ++S[Col[U[D[j]]=D[U[j]]=j]];
55          }
56          bool Dance(int d)
57          {
58                  if(R[0]==0)
59                  {
60                          ansd=d;
61                          return true;
62                  }
63                  int c=R[0];
64                  for(int i=R[0];i!=0;i=R[i])
65                          if(S[i]<S[c])c=i;
66                  remove(c);
67                  for(int i=D[c];i!=c;i=D[i])
68                  {
69                          ans[d]=Row[i];
70                          for(int j=R[i];j!=i;j=R[j])remove(Col[j]);
71                          if(Dance(d+1))return true;
72                          for(int j=L[i];j!=i;j=L[j])resume(Col[j]);
73                  }
74                  resume(c);
75                  return false;
76          }
77  }g;
```

2.2 DLX 重复覆盖

```
1   struct DLX
2   {
3           int n,m,sz;
4           int L[maxv],R[maxv],U[maxv],D[maxv],Row[maxv],Col[maxv];
5           int H[maxn],S[maxm];
6           int ansd;
7           void init(int N,int M)
8           {
9                   n=N;m=M;
10                  int i;
11                  for(i=0;i<=m;++i)
12                  {
13                          S[i]=0;
```

```
14                      U[i]=D[i]=i;
15                      L[i]=(i+m)%(m+1);
16                      R[i]=(i+1)%(m+1);
17              }
18              sz=m;
19              for(i=1;i<=n;++i)H[i]=-1;
20      }
21      void Link(int r,int c)
22      {
23              ++S[Col[++sz]=c];
24              Row[sz]=r;
25              D[sz]=D[c];U[D[c]]=sz;
26              U[sz]=c;D[c]=sz;
27              if(H[r]<0)H[r]=L[sz]=R[sz]=sz;
28              else
29              {
30                      R[sz]=R[H[r]];
31                      L[R[H[r]]]=sz;
32                      L[sz]=H[r];
33                      R[H[r]]=sz;
34              }
35      }
36      void remove(int c)
37      {
38              for(int i=D[c];i!=c;i=D[i])
39                      L[R[i]]=L[i],R[L[i]]=R[i];
40      }
41      void resume(int c)
42      {
43              for(int i=U[c];i!=c;i=U[i])
44                      L[R[i]]=R[L[i]]=i;
45      }
46      bool v[maxm];
47      int p()
48      {
49              int res=0;
50              for(int c=R[0];c!=0;c=R[c])v[c]=false;
51              for(int c=R[0];c!=0;c=R[c])
52                      if(!v[c])
53                      {
54                              ++res;
55                              v[c]=true;
56                              for(int i=D[c];i!=c;i=D[i])
57                                      for(int j=R[i];j!=i;j=R[j])
58                                              v[Col[j]]=true;
59                      }
60              return res;
61      }
62      void Dance(int d)
63      {
64              if(d+p()>=ansd)return;
65              if(R[0]==0)
66              {
67                      ansd=min(ansd,d);
```

```
68                          return;
69                  }
70                  int c=R[0];
71                  for(int i=R[0];i!=0;i=R[i])
72                          if(S[i]<S[c])c=i;
73                  for(int i=D[c];i!=c;i=D[i])
74                  {
75                          remove(i);
76                          for(int j=R[i];j!=i;j=R[j])remove(j);
77                          Dance(d+1);
78                          for(int j=L[i];j!=i;j=L[j])resume(j);
79                          resume(i);
80                  }
81          }
82  }g;
```

3 Graph Theory

3.1 zkw 费用流

```
1   struct ZKW_maxflowmincost
2   {
3           int head[maxn],dis[maxn],ednum,st,ed;
4           bool used[maxn];
5           struct Edge
6           {
7                   int to,cap,cost,next;
8                   Edge(){}
9                   Edge(int _to,int _cap,int _cost,int _next)
10                  {
11                          to=_to;cap=_cap;cost=_cost;next=_next;
12                  }
13          }edge[maxe];
14          void init()
15          {
16                  memset(head,-1,sizeof(head));
17                  ednum=0;
18          }
19          void addedge(int u,int v,int cap,int cost)
20          {
21                  edge[ednum]=Edge(v,cap,cost,head[u]);
22                  head[u]=ednum++;
23                  edge[ednum]=Edge(u,0,-cost,head[v]);
24                  head[v]=ednum++;
25          }
26          void spfa(int N)
27          {
28                  int u,p,v,i;
29                  ll dlen;
30                  for(i=0;i<=N;i++)dis[i]=INF;
31                  priority_queue<pair<int,ll> >q;
32                  q.push(make_pair(0,st));
33                  while(!q.empty())
```

```
34                  {
35                          u=q.top().second;dlen=-q.top().first;
36                          q.pop();
37                          if(dis[u]!=dlen)continue;
38                          for(p=head[u];p!=-1;p=edge[p].next)
39                          {
40                                  v=edge[p].to;
41                                  if(edge[p].cap&&dis[v]>dlen+edge[p].cost)
42                                  {
43                                          dis[v]=dlen+edge[p].cost;
44                                          q.push(make_pair(-dis[v],v));
45                                  }
46                          }
47                  }
48                  for(i=0;i<=N;i++)dis[i]=dis[ed]-dis[i];
49          }
50          int add_flow(int u,int flow,int &maxflow,int &mincost)
51          {
52                  if(u==ed)
53                  {
54                          maxflow+=flow;
55                          mincost+=dis[st]*flow;
56                          return flow;
57                  }
58                  used[u]=true;
59                  ll now=flow,tmp;
60                  int p,v;
61                  for(p=head[u];p!=-1;p=edge[p].next)
62                  {
63                          v=edge[p].to;
64                          if(edge[p].cap&&!used[v]&&dis[u]==dis[v]+edge[p].cost)
65                          {
66                                  tmp=add_flow(v,min(now,edge[p].cap*1ll),
                                        maxflow,mincost);
67                                  edge[p].cap-=tmp;
68                                  edge[p^1].cap+=tmp;
69                                  now-=tmp;
70                                  if(!now)break;
71                          }
72                  }
73                  return flow-now;
74          }
75          bool modify_label(int N)
76          {
77                  int i,u,v,d=INF,p;
78                  for(u=0;u<=N;u++)
79                          for(p=head[u];p!=-1&&used[u];p=edge[p].next)
80                          {
81                                  v=edge[p].to;
82                                  if(edge[p].cap&&!used[v])d=min(d,dis[v]+edge[p
                                        ].cost-dis[u]);
83                          }
84                  if(d==INF)return false;
85                  for(i=0;i<=N;i++)if(used[i])dis[i]+=d;
```

```
86                  return true;
87          }
88          void ZKW_MFMC(int s,int t,int N,int &maxflow,int &mincost)
89          {
90                  int i;
91                  st=s;ed=t;
92                  spfa(N);
93                  maxflow=mincost=0;
94                  do
95                  {
96                          for(i=0;i<=N;i++)used[i]=false;
97                  }while(add_flow(st,INF,maxflow,mincost)||modify_label(N));
98          }
99  }G;
```

3.2 spfa 费用流

```
1   struct MCMF
2   {
3       int n,m,ednum;
4       struct Edge
5       {
6       int from,to,cap,flow,cost,next;
7       Edge(int u,int v,int c,int f,int w,int n)
8       {from=u;to=v;cap=c;flow=f;cost=w;next=n;}
9       Edge(){}
10      }edge[maxe];
11      int head[maxn],dis[maxn],pre[maxn],flw[maxn],ednum;
12      bool inq[maxn];
13      void init(int vecnum)
14      {
15          n=vecnum;
16          memset(head,-1,sizeof(head));
17          ednum=0;
18      }
19      void addedge(int from,int to,int cap,int cost)
20      {
21          edge[ednum]=Edge(from,to,cap,0,cost,head[from]);
22          head[from]=ednum++;
23          edge[ednum]=Edge(to,from,0,0,-cost,head[to]);
24          head[to]=ednum++;
25      }
26      bool spfa(int s,int t,long long &flow,long long &cost)
27      {
28          int u,i;
29          for(i=0;i<=n;i++)
30              dis[i]=INF;
31          memset(inq,0,sizeof(inq));
32          dis[s]=0;inq[s]=1;pre[s]=0;flw[s]=INF;
33          queue<int> q;
34          q.push(s);
35          while(!q.empty())
36          {
37              u=q.front();
```

```
38              q.pop();
39              inq[u]=0;
40              for(i=head[u];i!=-1;i=edge[i].next)
41              {
42                  if(edge[i].cap>edge[i].flow&&dis[edge[i].to]>dis[u]+edge[i].
                        cost)
43                  {
44                      dis[edge[i].to]=dis[u]+edge[i].cost;
45                      pre[edge[i].to]=i;
46                      flw[edge[i].to]=min(flw[u],edge[i].cap-edge[i].flow);
47                      if(!inq[edge[i].to])
48                      {
49                          q.push(edge[i].to);
50                          inq[edge[i].to]=1;
51                      }
52                  }
53              }
54          }
55          if(dis[t]==INF)return false;
56          flow+=flw[t]*1ll;
57          cost+=dis[t]*1ll*flw[t];
58          for(i=t;i!=s;i=edge[pre[i]].from)
59          {
60              edge[pre[i]].flow+=flw[t];
61              edge[pre[i]^1].flow-=flw[t];
62          }
63          return true;
64      }
65      void MincostMaxflow(int s,int t,long long &flow,long long &cost)
66      {
67          flow=cost=0;
68          while(spfa(s,t,flow,cost));
69      }
70  }graph;
```

3.3 Dinic

```
1   struct Edge
2   {
3           int from,to,cap,next;
4           Edge(){}
5           Edge(int u,int v,int c,int n)
6           {
7                   from=u;to=v;cap=c;next=n;
8           }
9   }edge[20*maxn];
10  int ednum,head[maxn];
11  int n,m,dis[maxn];
12  void addedge(int u,int v,int c)
13  {
14          edge[ednum]=Edge(u,v,c,head[u]);
15          head[u]=ednum++;
16          edge[ednum]=Edge(v,u,0,head[v]);
17          head[v]=ednum++;
```

```
18  }
19  bool bfs(int st,int tp)
20  {
21          queue <int> q;
22          int now,i,u,v;
23          memset(dis,-1,sizeof(dis));
24          dis[st]=0;
25          q.push(st);
26          while(!q.empty())
27          {
28                  u=q.front();
29                  q.pop();
30                  for(i=head[u];i!=-1;i=edge[i].next)
31                  {
32                          v=edge[i].to;
33                          if(dis[v]==-1&&edge[i].cap>0)
34                          {
35                                  dis[v]=dis[u]+1;
36                                  q.push(v);
37                          }
38                  }
39          }
40          return (dis[tp]!=-1);
41  }
42  int dfs(int a,int b,int tp)
43  {
44          int res=0,v,i;
45          if(a==tp)return b;
46          for(i=head[a];i!=-1&&res<b;i=edge[i].next)
47          {
48                  v=edge[i].to;
49                  if(edge[i].cap>0&&dis[v]==dis[a]+1)
50                  {
51                          int x=min(edge[i].cap,b-res);
52                          x=dfs(v,x,tp);
53                          res+=x;
54                          edge[i].cap-=x;
55                          edge[i^1].cap+=x;
56                  }
57          }
58          if(!res)dis[a]=-2;
59          return res;
60  }
61  int dinic(int st,int tp)
62  {
63          int total=0,t;
64          while(bfs(st,tp))
65          {
66                  while(t=dfs(st,MAX,tp))
67                          total+=t;
68          }
69          return total;
70  }
```

3.4 哈密顿回路 (无向图)

```
bool mp[1001][1001];
bool inst[1001];
deque<int> q;
int n;
bool find(int u,int sign)
{
        for(int i=1;i<=n;i++)
                if(mp[u][i]&&!inst[i])
                {
                        inst[i]=true;
                        if(sign)q.push_back(i);
                        else q.push_front(i);
                        return true;
                }
        return false;
}
void Hamitton()
{
        int i;
        q.clear();
        memset(inst,0,sizeof(inst));
        q.push_back(1);inst[1]=true;
        while(find(q.back(),1));
        while(find(q.front(),0));
        do
        {
                if(!mp[q.front()][q.back()])
                {
                        for(i=1;i<q.size()-1;i++)
                                if(mp[q[i]][q.back()]&&mp[q[i+1]][q.front()])
                                {
                                        reverse(q.begin()+i+1,q.end());
                                        break;
                                }
                }
                if(q.size()<n)
                {
                        while(!find(q.front(),0))
                        {
                                q.push_back(q.front());
                                q.pop_front();
                        }
                        while(find(q.back(),1));
                        while(find(q.front(),0));
                }
        }while(q.size()<n||(!mp[q.front()][q.back()]));
}
```

3.5 欧拉回路

3.6 2-sat

4 Math

4.1 扩展 gcd

```
void ex_gcd(long long a,long long b,long long &d,long long &x,long long &y)
{
    if(!b){d=a;x=1;y=0;}
    else {ex_gcd(b,a%b,d,y,x);y-=x*(a/b);}
}
long long solve_mod_equation(long long a,long long b,long long c)
//a*x+b*y=c 求 x 的最小整数解
{
        long long x,y,d;
        ex_gcd(a,b,d,x,y);
        long long da=b/d,db=a/d,dc=c/d;
        x=(x%da+da)%da;
        x*=dc;
        x=(x%da+da)%da;
        return x;
}
```

4.2 快速幂

```
long long qukpow(long long k,long long base,long long mod)// base^k%mod
{
        long long res=1;
        while(k)
        {
                if(k&1)res*=base;
                base*=base;
        k>>=1;
        }
        return res;
}
```

4.3 线性筛素数

```
const int rn=1e5;//rangenum
const int pn=1e5;
bool isnp[rn];
int prime[pn];
int get_prime_number()
{
        memset(isnp,0,sizeof(isnp));
        int num=0,i,j;
        isnp[0]=isnp[1]=true;
        for(i=2;i<=rn;i++)
        {
                if(!isnp[i])prime[num++]=i;
```

```
                for(j=0;j<num&&i*prime[j]<=rn;j++)
                {
                        isnp[i*prime[j]]=1;
                        if(!(i%prime[j]))break;
                }
        }
        return num;
}
```

4.4 计算欧拉函数

```
int cal_euler_number(int n,int pnum) //should use the get_prime_number() above
{
        int i,k,ans=1;
        for(i=0;i<pnum&&prime[i]*prime[i]<=n;i++)
        {
                if(n%prime[i])continue;
                for(k=0;!(n/prime[i]%prime[i]);k++,n/=prime[i]);
                        ans*=prime[i];
                n/=prime[i];ans*=(prime[i]-1);
        }
        if(n>1)ans*=(n-1);
        return ans;
}
```

4.5 解二次方程

```
vector<double> solve_quadratic_equation(double a,double b,double c)
{
    double x1,x2,delta=b*b-4*a*c;
    const double eps=1e-8;
    vector<double> ans;
    if(delta<0)return ans;
    else if(delta<eps)
    {
        x1=-b/(2.0*a);
        ans.push_back(x1);
    }
    else
    {
        x1=(-b+sqrt(delta))/(2.0*a);
        x2=(-b-sqrt(delta))/(2.0*a);
        ans.push_back(x1);
        ans.push_back(x2);
    }
    return ans;
}
```

4.6 辛普森积分

```
double simpson(double (*f)(double),double a,double b,double sA,double eps,
    double fa,double fc,double fb,int depth)
```

```
//f 为待求函数，depth 为最大递归深度
{
        double c=(a+b)/2.0;
        double d=(a+c)/2.0,e=(c+b)/2.0;
        double fd=f(d),fe=f(e);
        double sL=(fa+4*fd+fc)/6.0*(c-a);
        double sR=(fc+4*fe+fb)/6.0*(b-c);
        if(depth==0||abs(sL+sR-sA)<15*eps)return sL+sR+(sL+sR-sA)/15.0;
        return simpson(f,a,c,sL,eps/2.0,fa,fd,fc,depth-1)+
                simpson(f,c,b,sR,eps/2.0,fc,fe,fb,depth-1);
}
double simpson(double (*f)(double),double a,double b,double eps,int depth)
{
        double fa=f(a),fb=f(b),fc=f((a+b)/2.0);
        double sA=(fa+4*fc+fb)/6.0*(b-a);
        return simpson(f,a,b,sA,eps,fa,fc,fb,depth);
}
```

4.7 龙贝格积分

```
double T[20][20];
int er[22];
LL si[22];

double Romberg(double a,double b,double eps)
{
    er[0]=1;si[0]=1;
    for (int j=1;j<22;j++) er[j]=er[j-1]*2,si[j]=si[j-1]*4;
    double h=b-a;
    T[0][0]=h/2*(f(a)+f(b));
    int now=0;
    do{
        now+=1;
        h/=2;
        double temp=0;
        for (int j=0;j<er[now]/2;j++)
            temp+=f((2*j+1.0)/er[now]*(b-a)+a);
        T[now][0]=T[now-1][0]/2+h*temp;
        for (int j=1;j<=now;j++){
            T[now][j]=(si[j]*T[now][j-1]-T[now-1][j-1])/(si[j]-1);
        }
    }while (abs(T[now][now]-T[now-1][now-1])>eps);
    return T[now][now];
}
```

4.8 找原根

```
const int max_prime_num=10000;
long long fprtmp[max_prime_num];
long long findPrimitiveRoot(long long mod)// g^euler_phi(mod)=1 %mod
//g^h=a mod m 充要条件为 h=log(g)a mod euler_phi(mod)
{
        if(mod==2)return 1;
```

```
7           int cnt=0,num=mod-1,j;// num=euler_phi(mod) 默认 p 是质数
8           for(int i=0;i<pnum&&prime[i]*prime[i]<=num&&num>1;i++)
9                   if(num%prime[i]==0)
10                  {
11                          fprtmp[cnt++]=prime[i];
12                          while(num%prime[i]==0)num/=prime[i];
13                  }
14          if(num!=1)fprtmp[cnt++]=num;
15          for(long long i=2;i<=mod-1;i++) // gcd(i,mod)=1
16          {
17                  for(j=0;j<cnt;j++)
18                          if(qukpow((mod-1)/fprtmp[j],i,mod)==1)break;
19                  if(j==cnt)return i;
20          }
21  }
```

4.9 FFT

```
1   const int maxn=2e5+10;
2   const double pi=acos(-1);
3   struct Virt
4   {
5           double r,i;
6           Virt(double R=0,double I=0)
7           {
8                   r=R;i=I;
9           }
10          Virt operator +(const Virt &p)
11          {
12                  return Virt(r+p.r,i+p.i);
13          }
14          Virt operator -(const Virt &p)
15          {
16                  return Virt(r-p.r,i-p.i);
17          }
18          Virt operator *(const Virt &p)
19          {
20                  return Virt(r*p.r-i*p.i,r*p.i+i*p.r);
21          }
22  }x[maxn<<2];
23  int num[maxn<<2];
24  void rader(Virt y[],int len)
25  {
26          int i,j,k;
27          for(i=1,j=len/2;i<len-1;i++)
28          {
29                  if(i<j)swap(y[i],y[j]);
30                  k=len/2;
31                  for(;j>=k;j-=k,k>>=1);
32                  if(j<k)j+=k;
33          }
34  }
35  void fft(Virt y[],int len,int on)
36  {
```

```
37          int h,k,j;
38          rader(y,len);
39          Virt u,t,w,wn;
40          for(h=2;h<=len;h<<=1)
41                  for(j=0,wn=Virt(cos(-on*2*pi/h),sin(-on*2*pi/h));j<len;j+=h)
42                          for(k=j,w=Virt(1,0);k<j+h/2;k++)
43                          {
44                                  u=y[k];
45                                  t=w*y[k+h/2];
46                                  y[k]=u+t;
47                                  y[k+h/2]=u-t;
48                                  w=w*wn;
49                          }
50          if(on==-1)for(j=0;j<len;j++)y[j].r/=len;
51  }
52  /*
53  while(len<maxv+maxv+2)len<<=1;   len 保证为 2 的幂次数
54  for(i=0;i<len;i++)
55          x[i]=Virt(num[i],0);
56  fft(x,len,1);     参数为 1 时,FFT
57  for(i=0;i<len;i++)x[i]=x[i]*x[i];
58  fft(x,len,-1);    参数为-1 时,IFFT
59  */
```

4.10 NNT

```
1  #include<bits/stdc++.h>
2  using namespace std;
3  typedef long long LL;
4  const int maxn = 300005;
5  const long long mod = 880803841;
6
7  LL powmod(LL a, LL b, LL p)
8  {
9      LL base = a, res = 1;
10     while(b) {
11         if(b % 2) res = res * base % p;
12         base = base * base % p;
13         b /= 2;
14     }
15     return res;
16 }
17
18 namespace NTT {
19     const int r = 26, gl = 25;  //r 是 p 的原根
20     LL p, rp[50], irp[50];
21     void setMod(LL _p = 880803841) {
22         p = _p;
23         for(int i = 0; i < gl; i++) rp[i] = powmod(r, (p-1)/(1<<i), p);
24     }
25     void FFT(LL a[], int n, LL wt[] = rp)
26     {
27         for(int i = 0, j = 0; i < n; i++) {
28             if(j > i) swap(a[i], a[j]);
```

```
29              int k = n;
30              while(j & (k >>= 1)) j &= ~k;
31              j |= k;
32          }
33          for(int m = 1, b = 1; m < n; m<<=1, b++)
34          for(int k = 0, w = 1; k < m; ++k) {
35              for(int i = k; i < n; i += m<<1) {
36                  int v = a[i+m] * w % p;
37                  if((a[i+m] = a[i] - v) < 0) a[i+m] += p;
38                  if((a[i] += v) >= p) a[i] -= p;
39              }
40              w = w * wt[b] % p;
41          }
42
43      }
44      void IFFT(LL a[], int n) {
45          for(int i = 0; i < gl; i++) irp[i] = powmod(rp[i], n-1, p);
46          FFT(a, n, irp);
47          LL inv = powmod(n, p-2, p);
48          for(int i = 0; i < n; i++) a[i] = a[i] * inv % p;
49      }
50      void Mul(LL a[], LL b[], LL n, LL c[]) {
51          FFT(a, n);FFT(b, n);
52          for(int i = 0; i < n; i++) c[i] = a[i] * b[i] % p;
53          IFFT(c, n);
54      }
55  }
56
57  LL a[maxn];
58  LL b[maxn];
59  LL c[maxn];
60  /*
61          scanf("%d%d%d",&N,&M,&D);
62          int n=1;
63          while (N>=n) n>>=1;
64          a[0]=1;b[0]=1;
65          for (int j=1;j<=n;j++){
66              a[j]=a[j-1]*inverse(j)%mod*(-1);a[j]=(mod+a[j])%mod;
67              b[j]=kuaisumi(j,N)*inverse(jiecheng[j])%mod;
68          }
69          n*=2;
70      NTT::setMod();
71      NTT::Mul(a, b, n, c);
72
73  */
```

4.11 二次剩余

```
1  #include <cstdio>
2  #include <cstring>
3  #include <algorithm>
4  #define mod 1000000009
5  using namespace std;
6  long long sqrt5,s,r1,r2;
```

```
7  long long ts,w;
8  struct D{
9      long long p,d;
10 };
11 void egcd(long long a,long long b,long long &x,long long &y)
12 {
13     if (b==0)
14     {
15         x=1;
16         y=0;
17         return;
18     }
19     egcd(b,a%b,x,y);
20     int t=x;
21     x=y,y=t-a/b*y;
22     return;
23 }
24 long long mypow(long long x,long long y,long long p)
25 {
26     long long res=1,mul=x;
27     while (y)
28     {
29         if (y & 1)
30             res=res * mul % p;
31         mul=mul * mul % p;
32         y/=2;
33     }
34     return res;
35 }
36 D mul(D a,D b,long long m)
37 {
38     D ans;
39     ans.p=(a.p * b.p % m +a.d * b.d %m *w % m)%m;
40     ans.d=(a.p * b.d % m +a.d * b.p% m)%m;
41     return ans;
42 }
43 D power(D a,long long b,long long m)
44 {
45     D ans;
46     ans.p = 1;
47     ans.d = 0;
48     while (b)
49     {
50         if (b & 1)
51         {
52             ans=mul(ans,a,m);
53         }
54         b/=2;
55         a=mul(a,a,m);
56     }
57     return ans;
58 }
59 long long sqre(long long x,long long y)
60 {
```

```
61      if (y==2) return 1;
62      if (mypow(x,(y-1)>>1,y)+1 == y)
63          return -1;
64      long long a,t;
65      for (a=1;a<y;a++)
66      {
67          t= a * a - x;
68          w= (t + y) % y;
69          if (mypow(w,(y-1)>>1,y)+1 == y) break;
70      }
71      D tmp;
72      tmp.p=a;
73      tmp.d=1;
74      D ans = power(tmp,(y+1)>>1,y);
75      return ans.p;
76  }
77  int main()
78  {
79      sqrt5=sqre(5,mod);
80      printf("%I64d\n",sqrt5);
81      long long x,y;
82      egcd(5,mod,x,y);
83      x=(x+mod)%mod;
84      s=(sqrt5*x)%mod;
85      printf("%I64d\n",s);
86
87      egcd(2,mod,x,y);
88      x=(x+mod)%mod;
89      r1=((sqrt5+1)*x)%mod;
90      r2=((-sqrt5+1+mod)*x)%mod;
91
92      printf("%I64d\n",r1);
93      printf("%I64d\n",r2);
94      int T;
95      return 0;
96  }
```

4.12 单纯形

```
1   struct Simplex
2   {
3           int n,m,B[maxn],N[maxn];
4           double ans[maxn],A[maxn][maxn],b[maxn],c[maxn],v;
5           void init(int _n,int _m)
6           {
7                   n=_n,m=_m;
8                   for(int i=1;i<=n;++i)N[i]=i;
9                   for(int i=n+1;i<=m;++i)B[i]=i;
10                  v=0;
11          }
12          void pivot(int l,int e)
13          {
14                  int i,j;
15                  double tmp=A[l][e];
```

```
16                  b[l]/=tmp;A[l][e]=1/tmp;
17                  for(i=1;i<=m;++i)if(i!=e)A[l][i]/=tmp;
18                  for(i=1;i<=n;++i)
19                          if(i!=l)
20                          {
21                                  b[i]-=A[i][e]*b[l];
22                                  for(j=1;j<=m;++j)
23                                          if(j!=e)A[i][j]-=A[i][e]*A[l][j];
24                                  A[i][e]=-A[i][e]/tmp;
25                          }
26                  v+=b[l]*c[e];
27                  for(i=1;i<=m;++i)
28                          if(i!=e)c[i]-=c[e]*A[l][i];
29                  c[e]*=-A[l][e];
30                  swap(B[l],N[e]);
31          }
32          bool simplex()
33          {
34                  int i,j,x,l,s=-1;
35                  double tmp=-inf,tmp1,tmp2;
36                  for(i=1;i<=m;++i)
37                          if(sgn(c[i])>0)
38                          {
39                                  tmp1=inf;
40                                  for(j=1;j<=n;++j)
41                                          if(sgn(A[j][i])>0)
42                                          {
43                                                  tmp2=b[j]/A[j][i];
44                                                  if(tmp2<tmp1)tmp1=tmp2,x=j;
45                                          }
46                                  if(tmp<tmp1*c[i])
47                                          s=i,l=x,tmp=tmp1*c[i];
48                          }
49                  if(s==-1)return false;
50                  pivot(l,s);
51                  return true;
52          }
53          void solve()
54          {
55                  while(simplex());
56                  memset(ans,0,sizeof(ans));
57                  for(int i=1;i<=n;++i)
58                          if(B[i]<=m)ans[B[i]]=b[i];
59          }
60  }simp;
61  int main()
62  {
63          int i,j,n,m;
64          while(scanf("%d%d",&n,&m)!=EOF)
65          {
66                  for(i=1;i<=n;++i)scanf("%lf",&simp.c[i]);
67                  for(i=1;i<=m;++i)
68                  {
69                          for(j=1;j<=n;++j)scanf("%lf",&simp.A[i][j]);
```

```
                        scanf("%lf",&simp.b[i]);
                }
                simp.init(m,n);
                simp.solve();
        printf("Nasa can spend %.0lf taka.\n",ceil(simp.v*m));
        }
        return 0;
}
```

4.13 FWT

```
typedef long long LL;
LL a[1<<20],b[1<<20],c[1<<20];
LL dp[2][1<<20];

//xor
void xortf(LL a[],int n)
{
        LL *x=dp[0],*y=dp[1];
        for(int i=0;i<(1<<n);++i)x[i]=a[i];
        for(int i=0;i<n;++i)
        {
                swap(x,y);
                for(int j=0;j<(1<<n);++j)
                        if(j>>i&1)
                                x[j]=y[j^(1<<i)]-y[j];
                        else
                                x[j]=y[j]+y[j^(1<<i)];
        }
        for(int i=0;i<(1<<n);++i)a[i]=x[i];
}
void xorutf(LL a[],int n)
{
        LL *x=dp[0],*y=dp[1];
        for(int i=0;i<(1<<n);++i)x[i]=a[i];
        for(int i=0;i<n;++i)
        {
                swap(x,y);
                for(int j=0;j<(1<<n);++j)
                        if(j>>i&1)
                                x[j]=y[j^(1<<i)]-y[j]>>1;
                        else
                                x[j]=y[j]+y[j^(1<<i)]>>1;
        }
        for(int i=0;i<(1<<n);++i)a[i]=x[i];
}
//or
void ortf(LL a[],int n)
{
        LL *x=dp[0],*y=dp[1];
        for(int i=0;i<(1<<n);++i)x[i]=a[i];
        for(int i=0;i<n;++i)
        {
                swap(x,y);
```

```
44                      for(int j=0;j<(1<<n);++j)
45                              if(j>>i&1)
46                                      x[j]=y[j^(1<<i)]+y[j];
47                              else
48                                      x[j]=y[j];
49              }
50              for(int i=0;i<(1<<n);++i)a[i]=x[i];
51      }
52      void orutf(LL a[],int n)
53      {
54              LL *x=dp[0],*y=dp[1];
55              for(int i=0;i<(1<<n);++i)x[i]=a[i];
56              for(int i=0;i<n;++i)
57              {
58                      swap(x,y);
59                      for(int j=0;j<(1<<n);++j)
60                              if(j>>i&1)
61                                      x[j]=y[j]-y[j^(1<<i)];
62                              else
63                                      x[j]=y[j];
64              }
65              for(int i=0;i<(1<<n);++i)a[i]=x[i];
66      }
67      //and
68      void andtf(LL a[],int n)
69      {
70              LL *x=dp[0],*y=dp[1];
71              for(int i=0;i<(1<<n);++i)x[i]=a[i];
72              for(int i=0;i<n;++i)
73              {
74                      swap(x,y);
75                      for(int j=0;j<(1<<n);++j)
76                              if(j>>i&1)
77                                      x[j]=y[j];
78                              else
79                                      x[j]=y[j]+y[j^(1<<i)];
80              }
81              for(int i=0;i<(1<<n);++i)a[i]=x[i];
82      }
83      void andutf(LL a[],int n)
84      {
85              LL *x=dp[0],*y=dp[1];
86              for(int i=0;i<(1<<n);++i)x[i]=a[i];
87              for(int i=0;i<n;++i)
88              {
89                      swap(x,y);
90                      for(int j=0;j<(1<<n);++j)
91                              if(j>>i&1)
92                                      x[j]=y[j];
93                              else
94                                      x[j]=y[j]-y[j^(1<<i)];
95              }
96              for(int i=0;i<(1<<n);++i)a[i]=x[i];
97      }
```

5 Geometry

5.1 旋转和镜像

```
Point Rotate(const double &theta)
{
        return Point(x*cos(theta)-y*sin(theta),x*sin(theta)+y*cos(theta));
}
Point Mirror(const double &theta)
{
        return Point(x*cos(2*theta)+y*sin(2*theta),x*sin(2*theta)-y*cos(2*
            theta));
}
```

5.2 凸包

```
bool isright(Point p,Point a,Point b) 浮点加 sgn
{
        if(det(p-b,p-a)==0)
        {
                if(dot(p-a,a-b)>=0)return true;
                else return false;
        }
        else return det(p-b,p-a)>0;
}
int convex(Point p[],int n) 不能含有重点
{
        int i,tmp,sn=0;
        for(i=0;i<n;i++)
        {
                while(sn>1&&!isright(p[i],p[s[sn-1]],p[s[sn-2]]))sn--;
                s[sn++]=i;
        }
        tmp=sn;
        for(i=n-2;i>=0;i--)
        {
                while(sn>tmp&&!isright(p[i],p[s[sn-1]],p[s[sn-2]]))sn--;
                s[sn++]=i;
        }
        if(sn>1)sn--;
        return sn;
}
```

5.3 两条直线求交点

```
Point Insect(Segment &p,Segment &q)
{
        double u=det(p.ed-p.st,q.st-p.st),v=det(p.st-p.ed,q.ed-p.st);
        return Point((q.st.x*v+q.ed.x*u)/(u+v),(q.st.y*v+q.ed.y*u)/(u+v));
}
```

5.4 过定点求圆的公切线

```
vector<Point> GetTangent(Point C,double r,Point a)// 返回切点
{
        vector<Point> tag;
        double x=a.x-C.x,y=a.y-C.y;
        double delta=sqr(x)+sqr(y)-sqr(r);
        if(sgn(delta)==0)
                tag.PB(Point(x-y,y+x)+C);          //在圆上，返回切线上某一点
        else
        {
                Point t1=Point(r*x-y*sqrt(delta),r*y+x*sqrt(delta))*(r/(sqr(x)
                    +sqr(y)));
                Point t2=Point(r*x+y*sqrt(delta),r*y-x*sqrt(delta))*(r/(sqr(x)
                    +sqr(y)));
                tag.PB(t1+C);tag.PB(t2+C);
        }
        return tag;
}
```

5.5 点与多边形的关系

```
bool inpolygon(Point p)
{
        int con=0;
        Segment now,tmp;
        Point cp;
        now.st=p;
        now.ed=Point(p.x+27051995,p.y+19952705);          注意取到无穷远
        for(int i=0;i<n;++i)
        {
                tmp.st=pt[i];
                tmp.ed=pt[(i+1)%n];
                if(sgn(det(tmp.ed-tmp.st,now.ed-now.st)))
                {
                        cp=Insect(now,tmp);
                        if(sgn(dot(tmp.ed-cp,cp-tmp.st))>0&&sgn(dot(now.ed-cp,
                            cp-now.st))>0)
                                ++con;
                        else if(sgn(dot(tmp.ed-cp,cp-tmp.st))==0)
                        {
                                cout <<"waring-Endpoints:"<<cp.x<<" "<<cp.y<<
                                    endl;
                                while(1);
                        }
                }
        }
        return con%2;
}
```

5.6 两圆面积交

```
double CirCalArea(Circle A,Circle B)
{
        double d=(A.c-B.c).len();
        if(d+eps>A.r+B.r)return 0;
        else if(A.r+d<B.r+eps)return sqr(A.r)*pi;
        else if(B.r+d<A.r+eps)return sqr(B.r)*pi;
        double angA=acos((sqr(A.r)+sqr(d)-sqr(B.r))/2/A.r/d);
        double angB=acos((sqr(B.r)+sqr(d)-sqr(A.r))/2/B.r/d);
        return angA*sqr(A.r)+angB*sqr(B.r)-sin(angA)*A.r*d;
}
```

5.7 最小圆覆盖 (随机增量)

```
void CirCover()
{
    int i,j,k;
    random_shuffle(pt,pt+n);
    cir=pt[0];r=0;
    for(i=1;i<n;++i)
        if(out(pt[i]))
        {
            cir=pt[i];r=0;
            for(j=0;j<i;++j)
                if(out(pt[j]))
                {
                    cir.x=(pt[i].x+pt[j].x)/2;
                    cir.y=(pt[i].y+pt[j].y)/2;
                    r=(pt[i]-cir).len();
                    for(k=0;k<j;++k)
                        if(out(pt[k]))
                        {
                            cir=TriAndCir(pt[i],pt[j],pt[k]);
                            r=(pt[i]-cir).len();
                        }
                }
        }
}
```

5.8 半平面交

```
struct Segment
{
        Point st,ed;
        double angle;
        void get_angle()
        {
                angle=atan2(ed.y-st.y,ed.x-st.x);
        }
        bool operator <(const Segment &p)const
        {
                if(sgn(angle-p.angle))return angle<p.angle;
                else return sgn(det(ed-st,p.ed-st))<0;
        }
```

```
14          double len()
15          {
16                  return sqrt(sqr(st.x-ed.x)+sqr(st.y-ed.y));
17          }
18  }seg[maxn],deq[maxn*2];
19  bool isout(Segment p,Point q)
20  {
21          return sgn(det(p.ed-p.st,q-p.st))<0;
22  }
23  bool isparallel(Segment p,Segment q)
24  {
25          return sgn(det(p.ed-p.st,q.ed-q.st))==0;
26  }
27  void HPI(int n)
28  {
29          int i,len=1,bot=0,top=1;
30          m=0;
31          for(i=0;i<n;++i)seg[i].arg();
32          sort(seg,seg+n);
33          for(i=1;i<n;++i)
34                  if(sgn(seg[i].angle-seg[len-1].angle))seg[len++]=seg[i];
35          deq[0]=seg[0];deq[1]=seg[1];
36          for(i=2;i<len;++i)
37          {
38                  if(isparallel(deq[top],deq[top-1])||isparallel(deq[bot],deq[
                        bot+1]))return;
39                  while(bot<top&&isout(seg[i],Insect(deq[top],deq[top-1])))--top
                        ;
40                  while(bot<top&&isout(seg[i],Insect(deq[bot],deq[bot+1])))++bot
                        ;
41                  deq[++top]=seg[i];
42          }
43          while(bot<top&&isout(deq[bot],Insect(deq[top],deq[top-1])))--top;
44          while(bot<top&&isout(deq[top],Insect(deq[bot],deq[bot+1])))++bot;
45          if(bot+1>=top)return;
46          for(i=bot;i<top;i++)
47                  p[m++]=Insect(deq[i],deq[i+1]);
48          p[m++]=Insect(deq[bot],deq[top]);
49  }
```

5.9 三角形的外接圆

```
1   Point TriAndCir(Point a,Point b,Point c) 返回圆心
2   {
3           double t,u,v;
4           t=2*(a.x-b.x)*(a.y-c.y)-2*(a.x-c.x)*(a.y-b.y);
5           u=(sqr(a.x)+sqr(a.y)-sqr(b.x)-sqr(b.y))/t;
6           v=(sqr(a.x)+sqr(a.y)-sqr(c.x)-sqr(c.y))/t;
7           return Point((a.y-c.y)*u-(a.y-b.y)*v,-(a.x-c.x)*u+(a.x-b.x)*v);
8   }
```

5.10 三维凸包

```
const double PR=1e-8;
const int N=510;
struct TPoint
{
        double x,y,z;
        TPoint(){}
        TPoint(double _x,double _y,double _z):x(_x),y(_y),z(_z){}
        TPoint operator-(const TPoint p)
        {return TPoint(x-p.x,y-p.y,z-p.z);}
        TPoint operator*(const TPoint p)
        {return TPoint(y*p.z-z*p.y,z*p.x-x*p.z,x*p.y-y*p.x);}//叉积
        double operator^(const TPoint p)
        {return x*p.x+y*p.y+z*p.z;}//点积
};
struct fac
{
        int a,b,c;//凸包一个面上的三个点的编号
        bool ok;//该面是否是最终凸包中的面
};
struct T3dhull
{
        int n;//初始点数
        TPoint ply[N];//初始点
        int trianglecnt;//凸包上三角形数
        fac tri[N];//凸包三角形, 理论 8 倍以上
        int vis[N][N];//点 i 到点 j 是属于哪个面
        double dist(TPoint a){return sqrt(a.x*a.x+a.y*a.y+a.z*a.z);}//两点长度
        double area(TPoint a,TPoint b,TPoint c){return dist((b-a)*(c-a));}//三
            角形面积 *2
        double volume(TPoint a,TPoint b,TPoint c,TPoint d){return (b-a)*(c-a)
            ^(d-a);}//四面体有向体积
            *6
        double ptoplane(TPoint &p,fac &f)//正：点在面同向
        {
                TPoint m=ply[f.b]-ply[f.a],n=ply[f.c]-ply[f.a],t=p-ply[f.a];
                return (m*n)^t;
        }
        void deal(int p,int a,int b)
        {
                int f=vis[a][b];//与当前面 (cnt) 共边 (ab) 的那个面
                fac add;
                if(tri[f].ok)
                {
                        if((ptoplane(ply[p],tri[f]))>PR) dfs(p,f);//如果 p 点能看
                            到该面 f，则继续深度探索 f 的 3 条边，以便更新新的凸包面
                        else//否则因为 p 点只看到 cnt 面，看不到 f 面，则 p 点和 a、b 点组成
                            一个三角形。
                        {
                                add.a=b,add.b=a,add.c=p,add.ok=1;
                                vis[p][b]=vis[a][p]=vis[b][a]=trianglecnt;
                                tri[trianglecnt++]=add;
                        }
                }
        }
```

```
50          void dfs(int p,int cnt)//维护凸包，如果点 p 在凸包外更新凸包
51          {
52                  tri[cnt].ok=0;//当前面需要删除，因为它在更大的凸包里面
53
54                  //下面把边反过来 (先 b, 后 a)，以便在 deal() 中判断与当前面 (cnt) 共边 (ab) 的那
                        个面。即判断与当头面 (cnt) 相邻的 3 个面 (它们与当前面的共边是反向的，如下
                        图中 (1) 的法线朝外 (即逆时针) 的面 130 和 312, 它们共边 13，但一个方向是
                        13, 另一个方向是 31)
55
56                  deal(p,tri[cnt].b,tri[cnt].a);
57                  deal(p,tri[cnt].c,tri[cnt].b);
58                  deal(p,tri[cnt].a,tri[cnt].c);
59          }
60          bool same(int s,int e)//判断两个面是否为同一面
61          {
62                  TPoint a=ply[tri[s].a],b=ply[tri[s].b],c=ply[tri[s].c];
63                  return fabs(volume(a,b,c,ply[tri[e].a]))<PR
64                          &&fabs(volume(a,b,c,ply[tri[e].b]))<PR
65                          &&fabs(volume(a,b,c,ply[tri[e].c]))<PR;
66          }
67          void construct()//构建凸包
68          {
69                  int i,j;
70                  trianglecnt=0;
71                  if(n<4) return ;
72                  random_shuffle(ply,ply+n);
73                  bool tmp=true;
74                  for(i=1;i<n;i++)//前两点不共点
75                  {
76                          if((dist(ply[0]-ply[i]))>PR)
77                          {
78                                  swap(ply[1],ply[i]); tmp=false; break;
79                          }
80                  }
81                  if(tmp) return;
82                  tmp=true;
83                  for(i=2;i<n;i++)//前三点不共线
84                  {
85                          if((dist((ply[0]-ply[1])*(ply[1]-ply[i])))>PR)
86                          {
87                                  swap(ply[2],ply[i]); tmp=false; break;
88                          }
89                  }
90                  if(tmp) return ;
91                  tmp=true;
92                  for(i=3;i<n;i++)//前四点不共面
93                  {
94                          if(fabs((ply[0]-ply[1])*(ply[1]-ply[2])^(ply[0]-ply[i
                                ]))>PR)
95                          {
96                                  swap(ply[3],ply[i]); tmp=false; break;
97                          }
98                  }
99                  if(tmp) return ;
```

```
100                 fac add;
101                 for(i=0;i<4;i++)//构建初始四面体 (4 个点为 ply[0],ply[1],ply[2],ply[3])
102                 {
103                         add.a=(i+1)%4,add.b=(i+2)%4,add.c=(i+3)%4,add.ok=1;
104                         if((ptoplane(ply[i],add))>0) swap(add.b,add.c);//保证逆
                                时针，即法向量朝外，这样新点才可看到。
105                         vis[add.a][add.b]=vis[add.b][add.c]=vis[add.c][add.a]=
                                trianglecnt;//逆向的有向边保
                                存
106                         tri[trianglecnt++]=add;
107                 }
108                 for(i=4;i<n;i++)//构建更新凸包
109                 {
110                         for(j=0;j<trianglecnt;j++)//对每个点判断是否在当前 3 维凸包
                                内或外 (i 表示当前点,j 表示当前面)
111                         {
112                                 if(tri[j].ok&&(ptoplane(ply[i],tri[j]))>PR)//对
                                        当前凸包面进行判断，看是否点能否看到这个面
113                                 {
114                                         dfs(i,j); break;//点能看到当前面，更新凸包
                                                的面 (递归，可能不止更新一个面)。当前点更新
                                                完成后 break 跳出循环
115
116                                 }
117                         }
118                 }
119                 int cnt=trianglecnt;
120                 //这些面中有一些 tri[i].ok=0，它们属于开始建立但后来因为在更大凸包内故需删除
                        的，所以下面几行代码的作用是只保存最外层的凸包
121                 trianglecnt=0;
122                 for(i=0;i<cnt;i++)
123                 {
124                         if(tri[i].ok)
125                                 tri[trianglecnt++]=tri[i];
126                 }
127         }
128         double area()//表面积
129         {
130                 double ret=0;
131                 for(int i=0;i<trianglecnt;i++)
132                         ret+=area(ply[tri[i].a],ply[tri[i].b],ply[tri[i].c]);
133                 return ret/2;
134         }
135         double volume()//体积
136         {
137                 TPoint p(0,0,0);
138                 double ret=0;
139                 for(int i=0;i<trianglecnt;i++)
140                         ret+=volume(p,ply[tri[i].a],ply[tri[i].b],ply[tri[i].c
                                ]);
141                 return fabs(ret/6);
142         }
143         int facetri() {return trianglecnt;}//表面三角形数
144         int facepolygon()//表面多边形数
```

```
145         {
146                 int ans=0,i,j,k;
147                 for(i=0;i<trianglecnt;i++)
148                 {
149                         for(j=0,k=1;j<i;j++)
150                         {
151                                 if(same(i,j)) {k=0;break;}
152                         }
153                         ans+=k;
154                 }
155                 return ans;
156         }
157 }hull;
```

5.11 多边形与圆求面积交

```
1  const double eps=1e-9;
2  inline double max (double a, double b) { if (a > b) return a; else return b; }
3  inline double min (double a, double b) { if (a < b) return a; else return b; }
4  class Vector
5  {
6          public:
7                  double x, y;
8                  Vector (void) {}
9                  Vector (double x0, double y0) : x(x0), y(y0) {}
10                 double operator * (const Vector& a) const { return x * a.y - y
                       * a.x; }
11                 double operator % (const Vector& a) const { return x * a.x + y
                       * a.y; }
12                 Vector operator - (const Vector& a) const { return Vector(x -
                      a.x, y - a.y); }
13                 Vector operator + (const Vector& a) const { return Vector(x +
                      a.x, y + a.y); }
14                 Vector verti (void) const { return Vector(-y, x); }
15                 double length (void) const { return sqrt(x * x + y * y); }
16                 Vector adjust (double len)
17                 {
18                         double ol = len / length();
19                         return Vector(x * ol, y * ol);
20                 }
21                 Vector oppose (void) { return Vector(-x, -y); }
22 };
23 typedef Vector Point;
24 class segment
25 {
26         public:
27                 Point a, b;
28                 segment (void) {}
29                 segment (Point a0, Point b0) : a(a0), b(b0) {}
30                 Point intersect (const segment& s) const
31                 {
32                         Vector v1 = s.a - a, v2 = s.b - a, v3 = s.b - b, v4 =
                              s.a - b;
33                         double s1 = v1 * v2, s2 = v3 * v4;
```

```
34                          double se = s1 + s2;
35                          s1 /= se, s2 /= se;
36                          return Point(a.x * s2 + b.x * s1, a.y * s2 + b.y * s1)
                                ;
37                  }
38                  Point pverti (const Point& p) const
39                  {
40                          Vector t = (b - a).verti();
41                          segment uv(p, p + t);
42                          return intersect(uv);
43                  }
44                  bool on_segment (const Point& p) const
45                  {
46                          if (sgn(min(a.x, b.x) - p.x) <= 0 && sgn(p.x - max(a.x
                                , b.x)) <= 0 &&
47                                  sgn(min(a.y, b.y) - p.y) <= 0 && sgn(p
                                      .y - max(a.y, b.y)) <= 0) return
                                      true;
48                          else return false;
49                  }
50  };
51  double radius;
52  Point polygon[510],A,B,P1,P2;
53  double kuras_area (Point a, Point b, double cx, double cy)// ceneter is(cx,cy)
        and radius should be set
54  {
55          Point ori(cx, cy);
56          int sg = sgn((b - ori) * (a - ori));
57          double da = (a - ori).length(), db = (b - ori).length();
58          int ra = sgn(da - radius), rb = sgn(db - radius);
59          double angle = acos(((b - ori) % (a - ori)) / (da * db));
60          segment t(a, b); Point h, u; Vector seg;
61          double ans, dlt, mov, tangle;
62
63          if (sgn(da) == 0 || sgn(db) == 0) return 0;
64          else if (sg == 0) return 0;
65          else if (ra <= 0 && rb <= 0) return fabs((b - ori) * (a - ori)) / 2 *
                sg;
66          else if (ra >= 0 && rb >= 0)
67          {
68                  h = t.pverti(ori);
69                  dlt = (h - ori).length();
70                  if (!t.on_segment(h) || sgn(dlt - radius) >= 0)
71                          return radius * radius * (angle / 2) * sg;
72                  else
73                  {
74                          ans = radius * radius * (angle / 2);
75                          tangle = acos(dlt / radius);
76                          ans -= radius * radius * tangle;
77                          ans += radius * sin(tangle) * dlt;
78                          return ans * sg;
79                  }
80          }
81          else
```

```
82          {
83                  h = t.pverti(ori);
84                  dlt = (h - ori).length();
85                  seg = b - a;
86                  mov = sqrt(radius * radius - dlt * dlt);
87                  seg = seg.adjust(mov);
88                  if (t.on_segment(h + seg)) u = h + seg;
89                  else u = h + seg.oppose();
90                  if (ra == 1) swap(a, b);
91                  ans = fabs((a - ori) * (u - ori)) / 2;
92                  tangle = acos(((u - ori) % (b - ori)) / ((u - ori).length() *
                        (b - ori).length()));
93                  ans += radius * radius * (tangle / 2);
94                  return ans * sg;
95          }
96  }
97  const double pi = acos(-1.0);
98  int n;
99  void solve(int cas,double cx,double cy,double r)
100 {
101         double area;
102         radius=r;
103         area = 0;
104         for (int i = 0; i <n; i++)
105                 area += kuras_area(polygon[i], polygon[(i + 1) % n], cx, cy);
106         printf("Case %d: %.10f\n",cas,fabs(area));
107 }
```

6 String

6.1 KMP

```
1  void getfail(char *P,int *F)
2  {
3      int m=strlen(P);
4      F[0]=0;F[1]=0;
5      for (int i=1;i<m;i++){
6          int j=F[i];
7          while (j && P[i]!=P[j]) j=F[j];
8          F[i+1]=P[i]==P[j]?j+1:0;
9
10     }
11 }
12
13 void KMP(char *T,char *P,int *F)
14 {
15     int n=strlen(T),m=strlen(P);
16     getfail(P,F);
17     int j=0;
18     for (int i=0;i<n;i++){
19         while (j && P[j]!=T[i]) j=F[j];
20         if (P[j]==T[i]) j++;
21         if (j==m) {
```

```
                printf("here\n");
                j=F[j];
            }
        }
}
```

可以直接建字符数组然后引用

6.2 AC 自动机

```
const int maxn=5e5+10;
struct Trie
{
        int next[maxn][26],fail[maxn],end[maxn];
        int root,id;
        int newnode()
        {
                ++id;
                for(int i=0;i<26;++i)next[id][i]=0;
                end[id]=0;
                return id;
        }
        void init()
        {
                id=0;
                root=newnode();
        }
        void insert(char buf[])
        {
                int len=strlen(buf);
                int now=root;
                for(int i=0;i<len;++i)
                {
                        if(!next[now][buf[i]-'a'])
                                next[now][buf[i]-'a']=newnode();
                        now=next[now][buf[i]-'a'];
                }
                ++end[now];
        }
        void build()
        {
                queue<int>q;
                fail[root]=root;
                for(int i=0;i<26;++i)
                        if(!next[root][i])next[root][i]=root;
                        else
                        {
                                fail[next[root][i]]=root;
                                q.push(next[root][i]);
                        }
                while(!q.empty())
                {
                        int now=q.front();q.pop();
                        for(int i=0;i<26;++i)
```

```
45                                      if(!next[now][i])next[now][i]=next[fail[now]][
                                            i];
46                                      else
47                                      {
48                                              fail[next[now][i]]=next[fail[now]][i];
49                                              q.push(next[now][i]);
50                                      }
51                      }
52              }
53              int query(char buf[])
54              {
55                      int len=strlen(buf);
56                      int now=root,res=0;
57                      for(int i=0;i<len;++i)
58                      {
59                              now=next[now][buf[i]-'a'];
60                              int tmp=now;
61                              while(tmp!=root)
62                              {
63                                      res+=end[tmp];
64                                      end[tmp]=0;
65                                      tmp=fail[tmp];
66                              }
67                      }
68                      return res;
69              }
70      }ac;
```

6.3 后缀数组

```
1   const int MAXN=200100
2   int wa[MAXN],wb[MAXN],wv[MAXN],wss[MAXN],sa[MAXN],r[MAXN];
3   inline int cmp(int *r,int a,int b,int l){
4       return r[a]==r[b] && r[a+l]==r[b+l];
5   }
6   inline void BuildSa(int *r,int *sa,int n,int m){
7       int i,j,p,*x=wa,*y=wb,*t;
8       for(int i=0;i<m;i++) wss[i]=0;
9       for(int i=0;i<n;i++) wss[x[i]=r[i]]+=1;
10      for(int i=1;i<m;i++) wss[i]+=wss[i-1];
11      for(int i=n-1;i>=0;i--) sa[--wss[x[i]]]=i;
12      for(j=1,p=1;p<n;j*=2,m=p){
13          for(p=0,i=n-j;i<n;i++) y[p++]=i;
14          for(int i=0;i<n;i++) if(sa[i]>=j) y[p++]=sa[i]-j;
15          for(int i=0;i<n;i++) wv[i]=x[y[i]];
16          for(int i=0;i<m;i++) wss[i]=0;
17          for(int i=0;i<n;i++) wss[wv[i]]+=1;
18          for(int i=0;i<m;i++) wss[i]+=wss[i-1];
19          for(int i=n-1;i>=0;i--) sa[--wss[wv[i]]]=y[i];
20          for(t=x,x=y,y=t,p=1,x[sa[0]]=0,i=1;i<n;i++)
21              x[sa[i]]=cmp(y,sa[i-1],sa[i],j)?p-1:p++;
22      }
23  }
24  int ranking[MAXN],height[MAXN];
```

```
inline void BuildHeight(int *r,int *sa,int n){
    int i,j,k=0;
    for(int i=1;i<=n;i++) ranking[sa[i]]=i;
    for(i=0;i<n;height[ranking[i++]]=k){
        for(k?k--:0,j=sa[ranking[i]-1];r[i+k]==r[j+k];k++);
    }
}
//注意二分时不能以 height[i]<len 作为分割线，因为最后的可能出现连续大于 len 的情况
//一般参考下面的写法
bool suan(int len,int K,int n) {
    int sum=1;
    for (int j=1; j<n; j++) {
        if (height[j]>=len) {
            sum++;
            if (sum>=K) return true;

        }else sum=1;
    }
    return false;
}
```

6.4 Manarcher

```
char str[M*2+2];//start from index 1
int p[M*2+2];  //p[i]-1 is the length of palindrome
char s[M];
int pre(char *s) {
    int i,j,k;
    int n = strlen(s);
    str[0] = '$';
    str[1] = '#';
    for (i=0;i<n;i++) {
        str[i*2 + 2] = s[i];
        str[i*2 + 3] = '#';
    }
    n = n*2 + 2;
    str[n] = 0;
    return n;
}
void Manarcher(char *s) {
    int i,id,mx=0,n=pre(s);
    for(i=1; i<n; i++) {
        if( mx > i )
            p[i] = min( p[2*id-i], p[id]+id-i );
        else
            p[i] = 1;
        for(; str[i+p[i]] == str[i-p[i]];++p[i]);
        if( p[i] + i > mx ) {
            mx = p[i] + i;
            id = i;
        }
    }
}
```

6.5 回文树

```
const int maxn=1e5+100;
int nxt[maxn][26],len[maxn],sufl[maxn];
int sz,suff;
int newnode(int l,int sl)
{
        ++sz;
        len[sz]=l;
        sufl[sz]=sl;
        for(int i=0;i<26;++i)nxt[sz][i]=0;
        return sz;
}
void init()
{
        sz=0,suff=1;
        newnode(-1,1);
        newnode(0,1);
}
int suffgo(int cur,char buf[],int pos)
{
        while(cur>1&&buf[pos-1-len[cur]]!=buf[pos])
                        cur=sufl[cur];
        return cur;
}
void update(char buf[],int pos)
{
        int cur=suffgo(suff,buf,pos),alp=buf[pos]-'a';
        if(nxt[cur][alp])
        {
                suff=nxt[cur][alp];
                return;
        }
        if(cur>1)
                suff=newnode(len[cur]+2,nxt[suffgo(sufl[cur],buf,pos)][alp]);
        else
                suff=newnode(len[cur]+2,2);
        nxt[cur][alp]=suff;
}
```

6.6 回文分解

```
const int maxn=3e5+100;
const int inf=(~0u)>>2;
int ans[maxn][2],dp[maxn][2];// 0 odd 1 even
namespace Eertree
{
        int nxt[maxn][26],len[maxn],sufl[maxn];
        int diff[maxn],serlink[maxn];
        int sz,suff;
        int newnode(int l,int sl)
        {
                ++sz;
```

```
12                      len[sz]=l;
13                      sufl[sz]=sl;
14                      diff[sz]=len[sz]-len[sl];
15                      for(int i=0;i<26;++i)nxt[sz][i]=0;
16                      if(diff[sz]==diff[sufl[sz]])
17                              serlink[sz]=serlink[sufl[sz]];
18                      else
19                              serlink[sz]=sufl[sz];
20                      return sz;
21              }
22              void init()
23              {
24                      sz=0,suff=1;
25                      newnode(-1,1);
26                      serlink[1]=1;
27                      newnode(0,1);
28                      diff[2]=0;
29              }
30              int suffgo(int cur,char buf[],int pos)
31              {
32                      while(cur>1&&buf[pos-1-len[cur]]!=buf[pos])
33                              cur=sufl[cur];
34                      return cur;
35              }
36              void update(char buf[],int pos)
37              {
38                      int cur=suffgo(suff,buf,pos),alp=buf[pos]-'a';
39                      if(!nxt[cur][alp])
40                      {
41                              if(cur>1)
42                                      nxt[cur][alp]=newnode(len[cur]+2,nxt[suffgo(
                                            sufl[cur],buf,pos)][alp]);
43                              else
44                                      nxt[cur][alp]=newnode(len[cur]+2,2);
45                      }
46                      suff=nxt[cur][alp];
47              }
48              void trans(int n)
49              {
50                      ans[n][0]=ans[n][1]=inf;
51                      for(int v=suff;len[v]>0;v=serlink[v])
52                      {
53                              dp[v][0]=ans[n-(len[serlink[v]]+diff[v])][0];
54                              dp[v][1]=ans[n-(len[serlink[v]]+diff[v])][1];
55                              if(diff[v]==diff[sufl[v]])
56                              {
57                                      dp[v][0]=min(dp[v][0],dp[sufl[v]][0]);
58                                      dp[v][1]=min(dp[v][1],dp[sufl[v]][1]);
59                              }
60                              ans[n][0]=min(ans[n][0],dp[v][1]+1);
61                              ans[n][1]=min(ans[n][1],dp[v][0]+1);
62                      }
63              }
64      }
```

```
65  char s[maxn];
66  int main()
67  {
68          scanf("%s",s);
69          int n=strlen(s);
70          Eertree::init();
71          ans[0][0]=inf;
72          ans[0][1]=0;
73          for(int i=1;i<=n;++i)
74          {
75                  Eertree::update(s,i-1);
76                  Eertree::trans(i);
77          }
78          int l,r;
79          for(int i=1;i<=n;++i)
80          {
81                  l=ans[i][0];
82                  r=ans[i][1];
83                  if(l==inf)l=-1;
84                  if(r==inf)r=-2;
85                  printf("%d %d\n",l,r);
86          }
87          return 0;
88  }
```

7 Data Structure

7.1 树链剖分

```
1   int sz[maxn],hv[maxn],dep[maxn],fa[maxn],anc[maxn],id[maxn],idx;
2   void pre_dfs(int u,int pre)
3   {
4           sz[u]=1;hv[u]=0;dep[u]=dep[pre]+1;fa[u]=pre;
5           int i,v;
6           for(i=head[u];i!=-1;i=edge[i].next)
7           {
8                   v=edge[i].v;
9                   if(v==pre)continue;
10                  pre_dfs(v,u);
11                  sz[u]+=sz[v];
12                  if(sz[hv[u]]<sz[v])hv[u]=v;
13          }
14  }
15  void pre_build(int u,int last)
16  {
17          anc[u]=last;id[u]=++idx;//edges or vertices
18          if(hv[u])pre_build(hv[u],last);
19          int i,v;
20          for(i=head[u];i!=-1;i=edge[i].next)
21          {
22                  v=edge[i].v;
23                  if(v==fa[u]||v==hv[u])continue;
24                  pre_build(v,v);
```

```
25          }
26  }
27  void operation(int x,int y)
28  {
29          int fx=anc[x],fy=anc[y];
30          while(fx!=fy)
31          {
32                  if(dep[fx]<dep[fy])
33                          swap(x,y),swap(fx,fy);
34                  deal();//path.update(1,1,idx,id[fx],id[x]);
35                  x=fa[fx];fx=anc[x];
36          }
37          if(dep[x]<dep[y])swap(x,y);
38          deal();//if(y!=x)path.update(1,1,idx,id[hv[y]],id[x]);
39  }
```

7.2 Splay

```
1   const int maxn=3e5+200;
2   #define lch(rt) nxt[(rt)][0]
3   #define rch(rt) nxt[(rt)][1]
4   int w[maxn],n;
5   struct SplayTree
6   {
7           int idx,root;
8           int nxt[maxn][2],pre[maxn];
9           LL key[maxn],lazy[maxn];
10          int sz[maxn];
11          bool rev[maxn];
12          void init()
13          {
14                  idx=root=0;
15                  sz[root]=pre[root]=nxt[root][0]=nxt[root][1]=0;
16          }
17          inline void push(int &rt)
18          {
19                  if(rev[rt])
20                  {
21                          swap(lch(rt),rch(rt));
22                          rev[lch(rt)]^=1;
23                          rev[rch(rt)]^=1;
24                  }
25                  key[lch(rt)]+=lazy[rt];lazy[lch(rt)]+=lazy[rt];
26                  key[rch(rt)]+=lazy[rt];lazy[rch(rt)]+=lazy[rt];
27                  rev[rt]=lazy[rt]=0;
28          }
29          inline void up(int &rt)
30          {
31                  sz[rt]=sz[lch(rt)]+sz[rch(rt)]+1;
32          }
33          inline void newnode(int &rt,int fa,int val)
34          {
35                  rt=++idx;
36                  sz[rt]=1;
```

```
37              rev[rt]=lazy[rt]=0;
38              lch(rt)=rch(rt)=0;
39              pre[rt]=fa;
40              key[rt]=val;
41      }
42      void build(int &rt,int l,int r,int fa)
43      {
44              if(l>r)return;
45              int mid=l+r>>1;
46              newnode(rt,fa,w[mid]);
47              build(lch(rt),l,mid-1,rt);
48              build(rch(rt),mid+1,r,rt);
49              up(rt);
50      }
51      void Rotate(int x,int kind)
52      {
53              int y=pre[x];
54              push(y);push(x);
55              nxt[y][!kind]=nxt[x][kind];
56              pre[nxt[x][kind]]=y;
57              pre[x]=pre[y];
58              if(pre[y])nxt[pre[y]][nxt[pre[y]][1]==y]=x;
59              nxt[x][kind]=y;pre[y]=x;
60              up(y);
61      }
62      void Splay(int x,int goal)
63      {
64              while(pre[x]!=goal)
65                      Rotate(x,nxt[pre[x]][0]==x);
66              if(!goal)root=x;
67              up(x);
68      }
69      int rank(int k)
70      {
71              int x;
72              for(x=root,push(x);k!=sz[lch(x)]+1;)
73              {
74                      if(k<=sz[lch(x)])x=lch(x);
75                      else
76                      {
77                              k-=(sz[lch(x)]+1);
78                              x=rch(x);
79                      }
80                      push(x);
81              }
82              return x;
83      }
84      void flip(int a,int b)
85      {
86              Splay(rank(a),0);
87              Splay(rank(b+2),root);
88              rev[lch(rch(root))]^=1;
89      }
90      void trouble(int &a,int b)
```

```
91          {
92                  int r=b-n,tmp;
93                  Splay(rank(1),0);
94                  Splay(rank(r+2),root);
95                  tmp=lch(rch(root));
96                  lch(rch(root))=0;
97                  up(rch(root));
98                  up(root);
99                  //[n-r,n-r]
100                 Splay(rank(n-r+1),0);
101                 Splay(rank(n-r+2),root);
102                 lch(rch(root))=tmp;
103                 pre[tmp]=rch(root);
104                 up(rch(root));
105                 up(root);
106                 a-=r;
107         }
108         void add(int a,int b,LL v)
109         {
110                 Splay(rank(a),0);
111                 Splay(rank(b+2),root);
112                 lazy[lch(rch(root))]+=v;
113                 key[lch(rch(root))]+=v;
114         }
115         void insert(int a,LL v)
116         {
117                 Splay(rank(a+1),0);
118                 Splay(rank(a+2),root);
119                 newnode(lch(rch(root)),rch(root),v);
120                 up(rch(root));
121                 up(root);
122         }
123         void del(int a)
124         {
125                 Splay(rank(a),0);
126                 Splay(rank(a+2),root);
127                 lch(rch(root))=0;
128                 pre[lch(rch(root))]=0;
129                 up(rch(root));
130                 up(root);
131         }
132         LL query(int a)
133         {
134                 return key[rank(a+1)];
135         }
136 }sp;
137 /*
138 w[0]=w[n+1]=-1;
139 sp.init();
140 sp.build(sp.root,0,n+1,0);
141 */
```

7.3 树分治

```
int n,S,root,rootv,sz[maxn];
void findroot(int u,int pre)
{
        int i,v,tmp=0;
        sz[u]=1;
        for(i=head[u];i!=-1;i=edge[i].next)
        {
                v=edge[i].v;
                if(v==pre||vis[v])continue;
                findroot(v,u);
                sz[u]+=sz[v];
                tmp=max(tmp,sz[v]);
        }
        tmp=max(tmp,S-sz[u]);
        if(tmp<rootv)
        {
                root=u;
                rootv=tmp;
        }
}
/*
   tree              subtree
   rootv=S=n;        rootv=S=sz[v];
   findroot(1,0);    findroot(v,u);
   merge(root);      merge(root);
 */
```

7.4 k-d 树

```
const int maxn=50005;
const int maxk=5;
#define MP make_pair
int K,m,which;
struct Point
{
        int x[maxk];
        bool operator <(const Point &p) const
        {
                return x[which]<p.x[which];
        }
}x[maxn],aim,ans[12];
typedef pair<double,Point> tp;
priority_queue<tp> q;
inline double sqr(double v)
{
        return v*v;
}
inline double length(const Point &p,const Point &q)
{
        double res=0;
        for(int i=0;i<K;i++)res+=sqr(p.x[i]-q.x[i]);
        return res;
}
void build(int l,int r,int dep)
```

```
{
        if(l>r)return;
        int mid=(l+r)>>1;
        which=dep%K;
        nth_element(x+l,x+mid,x+r+1);
        build(l,mid-1,dep+1);build(mid+1,r,dep+1);
}
void query(int l,int r,int dep)
{
        if(l>r)return;
        int mid=(l+r)>>1,loc=dep%K;
        double tmplen=length(aim,x[mid]);
        if(q.size()<m)q.push(MP(length(aim,x[mid]),x[mid]));
        else if(tmplen<q.top().first)q.pop(),q.push(MP(tmplen,x[mid]));
        if(l<r)
        {
                if(aim.x[loc]<x[mid].x[loc])
                {
                        query(l,mid-1,dep+1);
                        if(q.size()<m||aim.x[loc]+sqrt(q.top().first)>x[mid].x
                            [loc])query(mid+1,r,dep+1);
                }
                else
                {
                        query(mid+1,r,dep+1);
                        if(q.size()<m||aim.x[loc]-sqrt(q.top().first)<x[mid].x
                            [loc])query(l,mid-1,dep+1);
                }
        }
}
```

7.5 主席树

```
void build(int pre,int &rt,int l,int r,int pos,int v)
{
        rt=++idx;
        lch[rt]=lch[pre];rch[rt]=rch[pre],sum[rt]=sum[pre]+v;
        if(l==r)return;
        int mid=l+r>>1;
        if(pos<=mid)build(lch[pre],lch[rt],l,mid,pos,v);
        else build(rch[pre],rch[rt],mid+1,r,pos,v);
}
void update(int &rt,int l,int r,int pos,int v)
{
        if(rt==0)
        {
                rt=++idx;
                lch[rt]=rch[rt]=0;
        }
        sum[rt]+=v;
        if(l==r)return;
        int mid=l+r>>1;
        if(pos<=mid)update(lch[rt],l,mid,pos,v);
        else update(rch[rt],mid+1,r,pos,v);
```

```
22  }
```