Template

ZTemp_Zec

December 12, 2019

目录

1 Geo **1**
1.1 Geo_jls 1

2 Math **7**
2.1 FT_fft_2D 7
2.2 FT_fft_2D_使用说明 8
2.3 FT_fft 基础版本 8
2.4 NTT_2D 9
2.5 Poly_扩展版 10
2.6 Poly_标准版 13
2.7 一维 FFT_单模式串_模式串带通配符匹配 . . 14
2.8 二维 FFT_单模式串_模式串带通配符匹配 . . 15
2.9 分治 FFT 16
2.10 扩展卢卡斯 17

3 String **18**
3.1 PAM_优化转移_偶回文切割方案数 18
3.2 PAM_优化转移_最小回文切割段数 19
3.3 PAM_单串_支持双端插入 19
3.4 PAM_单串_支持撤销_单加 log 20
3.5 PAM_单串_支持撤销_单加可退化 20
3.6 PAM_多串模式 21
3.7 PAM_标准版本 21
3.8 PAM_空间压缩_单链表 21
3.9 SAM_多串模式_串运行_树上合并_map . . . 22
3.10 SAM_广义_trie 树 22
3.11 SAM_标准版 23

4 Zec **24**
4.1 划分数 24
4.2 多项式相关 24

1 Geo

1.1 Geo_jls

```
int cmp(db k1,db k2){return sign(k1-k2);}
int inmid(db k1,db k2,db k3){return sign(k1-k3)*sign(k2-k3)<=0;}// k3 在 [k1,k2] 内
struct point{
    db x,y;
    point operator + (const point &k1) const{return (point){k1.x+x,k1.y+y};}
    point operator - (const point &k1) const{return (point){x-k1.x,y-k1.y};}
    point operator * (db k1) const{return (point){x*k1,y*k1};}
    point operator / (db k1) const{return (point){x/k1,y/k1};}
    int operator == (const point &k1) const{return cmp(x,k1.x)==0&&cmp(y,k1.y)==0;}
    // 逆时针旋转
    point turn(db k1){return (point){x*cos(k1)-y*sin(k1),x*sin(k1)+y*cos(k1)};}
    point turn90(){return (point){-y,x};}
    bool operator < (const point k1) const{
        int a=cmp(x,k1.x);
        if (a==-1) return 1; else if (a==1) return 0; else return cmp(y,k1.y)==-1;
    }
    db abs(){return sqrt(x*x+y*y);}
    db abs2(){return x*x+y*y;}
    db dis(point k1){return ((*this)-k1).abs();}
    point unit(){db w=abs(); return (point){x/w,y/w};}
    void scan(){double k1,k2; scanf("%lf%lf",&k1,&k2); x=k1; y=k2;}
    void print(){printf("%.11lf %.11lf\n",x,y);}
    db getw(){return atan2(y,x);}
    point getdel(){if (sign(x)==-1||(sign(x)==0&&sign(y)==-1)) return (*this)*(-1); else return (*this);}
   int getP() const{return sign(y)==1||(sign(y)==0&&sign(x)==-1);}
};
int inmid(point k1,point k2,point k3){return inmid(k1.x,k2.x,k3.x)&&inmid(k1.y,k2.y,k3.y);}
db cross(point k1,point k2){return k1.x*k2.y-k1.y*k2.x;}
db dot(point k1,point k2){return k1.x*k2.x+k1.y*k2.y;}
db rad(point k1,point k2){return atan2(cross(k1,k2),dot(k1,k2));}
// -pi -> pi
int compareangle (point k1,point k2){
    return k1.getP()<k2.getP()||(k1.getP()==k2.getP()&&sign(cross(k1,k2))>0);
}
point proj(point k1,point k2,point q){ // q 到直线 k1,k2 的投影
    point k=k2-k1; return k1+k*(dot(q-k1,k)/k.abs2());
}
point reflect(point k1,point k2,point q){return proj(k1,k2,q)*2-q;}
int clockwise(point k1,point k2,point k3){// k1 k2 k3 逆时针 1 顺时针 -1 否则 0
    return sign(cross(k2-k1,k3-k1));
}
int checkLL(point k1,point k2,point k3,point k4){// 求直线 (L) 线段 (S)k1,k2 和 k3,k4 的交点
    return cmp(cross(k3-k1,k4-k1),cross(k3-k2,k4-k2))!=0;
}
point getLL(point k1,point k2,point k3,point k4){
    db w1=cross(k1-k3,k4-k3),w2=cross(k4-k3,k2-k3); return (k1*w2+k2*w1)/(w1+w2);
}
int intersect(db l1,db r1,db l2,db r2){
    if (l1>r1) swap(l1,r1); if (l2>r2) swap(l2,r2); return cmp(r1,l2)!=-1&&cmp(r2,l1)!=-1;
}
int checkSS(point k1,point k2,point k3,point k4){
    return intersect(k1.x,k2.x,k3.x,k4.x)&&intersect(k1.y,k2.y,k3.y,k4.y)&&
    sign(cross(k3-k1,k4-k1))*sign(cross(k3-k2,k4-k2))<=0&&
    sign(cross(k1-k3,k2-k3))*sign(cross(k1-k4,k2-k4))<=0;
}
db disSP(point k1,point k2,point q){
    point k3=proj(k1,k2,q);
    if (inmid(k1,k2,k3)) return q.dis(k3); else return min(q.dis(k1),q.dis(k2));
}
db disSS(point k1,point k2,point k3,point k4){
    if (checkSS(k1,k2,k3,k4)) return 0;
    else return min(min(disSP(k1,k2,k3),disSP(k1,k2,k4)),min(disSP(k3,k4,k1),disSP(k3,k4,k2)));
}
int onS(point k1,point k2,point q){return inmid(k1,k2,q)&&sign(cross(k1-q,k2-k1))==0;}
struct circle{
    point o; db r;
    void scan(){o.scan(); scanf("%lf",&r);}
    int inside(point k){return cmp(r,o.dis(k));}
};
struct line{
    // p[0]->p[1]
    point p[2];
    line(point k1,point k2){p[0]=k1; p[1]=k2;}
    point& operator [] (int k){return p[k];}
    int include(point k){return sign(cross(p[1]-p[0],k-p[0]))>0;}
    point dir(){return p[1]-p[0];}
    line push(){ // 向外 ( 左手边 ) 平移 eps
        const db eps = 1e-6;
```

```
        point delta=(p[1]-p[0]).turn90().unit()*eps;
        return {p[0]-delta,p[1]-delta};
    }
};
point getLL(line k1,line k2){return getLL(k1[0],k1[1],k2[0],k2[1]);}
int parallel(line k1,line k2){return sign(cross(k1.dir(),k2.dir()))==0;}
int sameDir(line k1,line k2){return parallel(k1,k2)&&sign(dot(k1.dir(),k2.dir()))==1;}
int operator < (line k1,line k2){
    if (sameDir(k1,k2)) return k2.include(k1[0]);
    return compareangle(k1.dir(),k2.dir());
}
int checkpos(line k1,line k2,line k3){return k3.include(getLL(k1,k2));}
vector<line> getHL(vector<line> &L){ // 求半平面交 , 半平面是逆时针方向 , 输出按照逆时针
    sort(L.begin(),L.end()); deque<line> q;
    for (int i=0;i<(int)L.size();i++){
        if (i&&sameDir(L[i],L[i-1])) continue;
        while (q.size()>1&&!checkpos(q[q.size()-2],q[q.size()-1],L[i])) q.pop_back();
        while (q.size()>1&&!checkpos(q[1],q[0],L[i])) q.pop_front();
        q.push_back(L[i]);
    }
    while (q.size()>2&&!checkpos(q[q.size()-2],q[q.size()-1],q[0])) q.pop_back();
    while (q.size()>2&&!checkpos(q[1],q[0],q[q.size()-1])) q.pop_front();
    vector<line>ans; for (int i=0;i<q.size();i++) ans.push_back(q[i]);
    return ans;
}
db closepoint(vector<point>&A,int l,int r){ // 最近点对 , 先要按照 x 坐标排序
    if (r-l<=5){
        db ans=1e20;
        for (int i=l;i<=r;i++) for (int j=i+1;j<=r;j++) ans=min(ans,A[i].dis(A[j]));
        return ans;
    }
    int mid=l+r>>1; db ans=min(closepoint(A,l,mid),closepoint(A,mid+1,r));
    vector<point>B; for (int i=l;i<=r;i++) if (abs(A[i].x-A[mid].x)<=ans) B.push_back(A[i]);
    sort(B.begin(),B.end(),[](point k1,point k2){return k1.y<k2.y;});
    for (int i=0;i<B.size();i++) for (int j=i+1;j<B.size()&&B[j].y-B[i].y<ans;j++) ans=min(ans,B[i].dis(B[j]));
    return ans;
}
int checkposCC(circle k1,circle k2){// 返回两个圆的公切线数量
    if (cmp(k1.r,k2.r)==-1) swap(k1,k2);
    db dis=k1.o.dis(k2.o);  int w1=cmp(dis,k1.r+k2.r),w2=cmp(dis,k1.r-k2.r);
    if (w1>0) return 4; else if (w1==0) return 3; else if (w2>0) return 2;
    else if (w2==0) return 1; else return 0;
}
vector<point> getCL(circle k1,point k2,point k3){ // 沿着 k2->k3 方向给出 , 相切给出两个
    point k=proj(k2,k3,k1.o); db d=k1.r*k1.r-(k-k1.o).abs2();
    if (sign(d)==-1) return {};
    point del=(k3-k2).unit()*sqrt(max((db)0.0,d)); return {k-del,k+del};
}
vector<point> getCC(circle k1,circle k2){// 沿圆 k1 逆时针给出 , 相切给出两个
    int pd=checkposCC(k1,k2); if (pd==0||pd==4) return {};
    db a=(k2.o-k1.o).abs2(),cosA=(k1.r*k1.r+a-k2.r*k2.r)/(2*k1.r*sqrt(max(a,(db)0.0)));
    db b=k1.r*cosA,c=sqrt(max((db)0.0,k1.r*k1.r-b*b));
    point k=(k2.o-k1.o).unit(),m=k1.o+k*b,del=k.turn90()*c;
    return {m-del,m+del};
}
vector<point> TangentCP(circle k1,point k2){// 沿圆 k1 逆时针给出
    db a=(k2-k1.o).abs(),b=k1.r*k1.r/a,c=sqrt(max((db)0.0,k1.r*k1.r-b*b));
    point k=(k2-k1.o).unit(),m=k1.o+k*b,del=k.turn90()*c;
    return {m-del,m+del};
}
vector<line> TangentoutCC(circle k1,circle k2){
    int pd=checkposCC(k1,k2); if (pd==0) return {};
    if (pd==1){point k=getCC(k1,k2)[0]; return {(line){k,k}};}
    if (cmp(k1.r,k2.r)==0){
        point del=(k2.o-k1.o).unit().turn90().getdel();
        return {(line){k1.o-del*k1.r,k2.o-del*k2.r},(line){k1.o+del*k1.r,k2.o+del*k2.r}};
    } else {
        point p=(k2.o*k1.r-k1.o*k2.r)/(k1.r-k2.r);
        vector<point>A=TangentCP(k1,p),B=TangentCP(k2,p);
        vector<line>ans; for (int i=0;i<A.size();i++) ans.push_back((line){A[i],B[i]});
        return ans;
    }
}
vector<line> TangentinCC(circle k1,circle k2){
    int pd=checkposCC(k1,k2); if (pd<=2) return {};
    if (pd==3){point k=getCC(k1,k2)[0]; return {(line){k,k}};}
    point p=(k2.o*k1.r+k1.o*k2.r)/(k1.r+k2.r);
    vector<point>A=TangentCP(k1,p),B=TangentCP(k2,p);
    vector<line>ans; for (int i=0;i<A.size();i++) ans.push_back((line){A[i],B[i]});
    return ans;
}
vector<line> TangentCC(circle k1,circle k2){
```

```
    int flag=0; if (k1.r<k2.r) swap(k1,k2),flag=1;
    vector<line>A=TangentoutCC(k1,k2),B=TangentinCC(k1,k2);
    for (line k:B) A.push_back(k);
    if (flag) for (line &k:A) swap(k[0],k[1]);
    return A;
}
db getarea(circle k1,point k2,point k3){
    // 圆 k1 与三角形 k2 k3 k1.o 的有向面积交
    point k=k1.o; k1.o=k1.o-k; k2=k2-k; k3=k3-k;
    int pd1=k1.inside(k2),pd2=k1.inside(k3);
    vector<point>A=getCL(k1,k2,k3);
    if (pd1>=0){
        if (pd2>=0) return cross(k2,k3)/2;
        return k1.r*k1.r*rad(A[1],k3)/2+cross(k2,A[1])/2;
    } else if (pd2>=0){
        return k1.r*k1.r*rad(k2,A[0])/2+cross(A[0],k3)/2;
    }else {
        int pd=cmp(k1.r,disSP(k2,k3,k1.o));
        if (pd<=0) return k1.r*k1.r*rad(k2,k3)/2;
        return cross(A[0],A[1])/2+k1.r*k1.r*(rad(k2,A[0])+rad(A[1],k3))/2;
    }
}
circle getcircle(point k1,point k2,point k3){
    db a1=k2.x-k1.x,b1=k2.y-k1.y,c1=(a1*a1+b1*b1)/2;
    db a2=k3.x-k1.x,b2=k3.y-k1.y,c2=(a2*a2+b2*b2)/2;
    db d=a1*b2-a2*b1;
    point o=(point){k1.x+(c1*b2-c2*b1)/d,k1.y+(a1*c2-a2*c1)/d};
    return (circle){o,k1.dis(o)};
}
circle getScircle(vector<point> A){
    random_shuffle(A.begin(),A.end());
    circle ans=(circle){A[0],0};
    for (int i=1;i<A.size();i++)
        if (ans.inside(A[i])==-1){
            ans=(circle){A[i],0};
            for (int j=0;j<i;j++)
                if (ans.inside(A[j])==-1){
                    ans.o=(A[i]+A[j])/2; ans.r=ans.o.dis(A[i]);
                    for (int k=0;k<j;k++)
                        if (ans.inside(A[k])==-1)
                            ans=getcircle(A[i],A[j],A[k]);
                }
        }
    return ans;
}
db area(vector<point> A){ // 多边形用 vector<point> 表示 , 逆时针
    db ans=0;
    for (int i=0;i<A.size();i++) ans+=cross(A[i],A[(i+1)%A.size()]);
    return ans/2;
}
int checkconvex(vector<point>A){
    int n=A.size(); A.push_back(A[0]); A.push_back(A[1]);
    for (int i=0;i<n;i++) if (sign(cross(A[i+1]-A[i],A[i+2]-A[i]))==-1) return 0;
    return 1;
}
int contain(vector<point>A,point q){ // 2 内部 1 边界 0 外部
    int pd=0; A.push_back(A[0]);
    for (int i=1;i<A.size();i++){
        point u=A[i-1],v=A[i];
        if (onS(u,v,q)) return 1; if (cmp(u.y,v.y)>0) swap(u,v);
        if (cmp(u.y,q.y)>=0||cmp(v.y,q.y)<0) continue;
        if (sign(cross(u-v,q-v))<0) pd^=1;
    }
    return pd<<1;
}
vector<point> ConvexHull(vector<point>A,int flag=1){ // flag=0 不严格 flag=1 严格
    int n=A.size(); vector<point>ans(n*2);
    sort(A.begin(),A.end()); int now=-1;
    for (int i=0;i<A.size();i++){
        while (now>0&&sign(cross(ans[now]-ans[now-1],A[i]-ans[now-1]))<flag) now--;
        ans[++now]=A[i];
    } int pre=now;
    for (int i=n-2;i>=0;i--){
        while (now>pre&&sign(cross(ans[now]-ans[now-1],A[i]-ans[now-1]))<flag) now--;
        ans[++now]=A[i];
    } ans.resize(now); return ans;
}
db convexDiameter(vector<point>A){
    int now=0,n=A.size(); db ans=0;
    for (int i=0;i<A.size();i++){
        now=max(now,i);
        while (1){
```

```
            db k1=A[i].dis(A[now%n]),k2=A[i].dis(A[(now+1)%n]);
            ans=max(ans,max(k1,k2)); if (k2>k1) now++; else break;
        }
    }
    return ans;
}
vector<point> convexcut(vector<point>A,point k1,point k2){
    // 保留 k1,k2,p 逆时针的所有点
    int n=A.size(); A.push_back(A[0]); vector<point>ans;
    for (int i=0;i<n;i++){
        int w1=clockwise(k1,k2,A[i]),w2=clockwise(k1,k2,A[i+1]);
        if (w1>=0) ans.push_back(A[i]);
        if (w1*w2<0) ans.push_back(getLL(k1,k2,A[i],A[i+1]));
    }
    return ans;
}
int checkPoS(vector<point>A,point k1,point k2){
    // 多边形 A 和直线 ( 线段 )k1->k2 严格相交 , 注释部分为线段
    struct ins{
        point m,u,v;
        int operator < (const ins& k) const {return m<k.m;}
    }; vector<ins>B;
    //if (contain(A,k1)==2||contain(A,k2)==2) return 1;
    vector<point>poly=A; A.push_back(A[0]);
    for (int i=1;i<A.size();i++) if (checkLL(A[i-1],A[i],k1,k2)){
        point m=getLL(A[i-1],A[i],k1,k2);
        if (inmid(A[i-1],A[i],m)/*&&inmid(k1,k2,m)*/) B.push_back((ins){m,A[i-1],A[i]});
    }
    if (B.size()==0) return 0; sort(B.begin(),B.end());
    int now=1; while (now<B.size()&&B[now].m==B[0].m) now++;
    if (now==B.size()) return 0;
    int flag=contain(poly,(B[0].m+B[now].m)/2);
    if (flag==2) return 1;
    point d=B[now].m-B[0].m;
    for (int i=now;i<B.size();i++){
        if (!(B[i].m==B[i-1].m)&&flag==2) return 1;
        int tag=sign(cross(B[i].v-B[i].u,B[i].m+d-B[i].u));
        if (B[i].m==B[i].u||B[i].m==B[i].v) flag+=tag; else flag+=tag*2;
    }
    //return 0;
    return flag==2;
}
int checkinp(point r,point l,point m){
   if (compareangle(l,r)){return compareangle(l,m)&&compareangle(m,r);}
   return compareangle(l,m)||compareangle(m,r);
}
int checkPosFast(vector<point>A,point k1,point k2){ // 快速检查线段是否和多边形严格相交
   if (contain(A,k1)==2||contain(A,k2)==2) return 1; if (k1==k2) return 0;
   A.push_back(A[0]); A.push_back(A[1]);
   for (int i=1;i+1<A.size();i++)
      if (checkLL(A[i-1],A[i],k1,k2)){
         point now=getLL(A[i-1],A[i],k1,k2);
         if (inmid(A[i-1],A[i],now)==0||inmid(k1,k2,now)==0) continue;
         if (now==A[i]){
            if (A[i]==k2) continue;
            point pre=A[i-1],ne=A[i+1];
            if (checkinp(pre-now,ne-now,k2-now)) return 1;
         } else if (now==k1){
            if (k1==A[i-1]||k1==A[i]) continue;
            if (checkinp(A[i-1]-k1,A[i]-k1,k2-k1)) return 1;
         } else if (now==k2||now==A[i-1]) continue;
         else return 1;
      }
   return 0;
}
// 拆分凸包成上下凸壳凸包尽量都随机旋转一个角度来避免出现相同横坐标
// 尽量特判只有一个点的情况凸包逆时针
void getUDP(vector<point>A,vector<point>&U,vector<point>&D){
    db l=1e100,r=-1e100;
    for (int i=0;i<A.size();i++) l=min(l,A[i].x),r=max(r,A[i].x);
    int wherel,wherer;
    for (int i=0;i<A.size();i++) if (cmp(A[i].x,l)==0) wherel=i;
    for (int i=A.size();i;i--) if (cmp(A[i-1].x,r)==0) wherer=i-1;
    U.clear(); D.clear(); int now=wherel;
    while (1){D.push_back(A[now]); if (now==wherer) break; now++; if (now>=A.size()) now=0;}
    now=wherel;
    while (1){U.push_back(A[now]); if (now==wherer) break; now--; if (now<0) now=A.size()-1;}
}
// 需要保证凸包点数大于等于 3,2 内部 ,1 边界 ,0 外部
int containCoP(const vector<point>&U,const vector<point>&D,point k){
    db lx=U[0].x,rx=U[U.size()-1].x;
    if (k==U[0]||k==U[U.size()-1]) return 1;
```

```
    if (cmp(k.x,lx)==-1||cmp(k.x,rx)==1) return 0;
    int where1=lower_bound(U.begin(),U.end(),(point){k.x,-1e100})-U.begin();
    int where2=lower_bound(D.begin(),D.end(),(point){k.x,-1e100})-D.begin();
    int w1=clockwise(U[where1-1],U[where1],k),w2=clockwise(D[where2-1],D[where2],k);
    if (w1==1||w2==-1) return 0; else if (w1==0||w2==0) return 1; return 2;
}
// d 是方向 , 输出上方切点和下方切点
pair<point,point> getTangentCow(const vector<point> &U,const vector<point> &D,point d){
    if (sign(d.x)<0||(sign(d.x)==0&&sign(d.y)<0)) d=d*(-1);
    point whereU,whereD;
    if (sign(d.x)==0) return mp(U[0],U[U.size()-1]);
    int l=0,r=U.size()-1,ans=0;
    while (l<r){int mid=l+r>>1; if (sign(cross(U[mid+1]-U[mid],d))<=0) l=mid+1,ans=mid+1; else r=mid;}
    whereU=U[ans]; l=0,r=D.size()-1,ans=0;
    while (l<r){int mid=l+r>>1; if (sign(cross(D[mid+1]-D[mid],d))>=0) l=mid+1,ans=mid+1; else r=mid;}
    whereD=D[ans]; return mp(whereU,whereD);
}
// 先检查 contain, 逆时针给出
pair<point,point> getTangentCoP(const vector<point>&U,const vector<point>&D,point k){
    db lx=U[0].x,rx=U[U.size()-1].x;
    if (k.x<lx){
        int l=0,r=U.size()-1,ans=U.size()-1;
        while (l<r){int mid=l+r>>1; if (clockwise(k,U[mid],U[mid+1])==1) l=mid+1; else ans=mid,r=mid;}
        point w1=U[ans]; l=0,r=D.size()-1,ans=D.size()-1;
        while (l<r){int mid=l+r>>1; if (clockwise(k,D[mid],D[mid+1])==-1) l=mid+1; else ans=mid,r=mid;}
        point w2=D[ans]; return mp(w1,w2);
    } else if (k.x>rx){
        int l=1,r=U.size(),ans=0;
        while (l<r){int mid=l+r>>1; if (clockwise(k,U[mid],U[mid-1])==-1) r=mid; else ans=mid,l=mid+1;}
        point w1=U[ans]; l=1,r=D.size(),ans=0;
        while (l<r){int mid=l+r>>1; if (clockwise(k,D[mid],D[mid-1])==1) r=mid; else ans=mid,l=mid+1;}
        point w2=D[ans]; return mp(w2,w1);
    } else {
        int where1=lower_bound(U.begin(),U.end(),(point){k.x,-1e100})-U.begin();
        int where2=lower_bound(D.begin(),D.end(),(point){k.x,-1e100})-D.begin();
        if ((k.x==lx&&k.y>U[0].y)||(where1&&clockwise(U[where1-1],U[where1],k)==1)){
            int l=1,r=where1+1,ans=0;
            while (l<r){int mid=l+r>>1; if (clockwise(k,U[mid],U[mid-1])==1) ans=mid,l=mid+1; else r=mid;}
            point w1=U[ans]; l=where1,r=U.size()-1,ans=U.size()-1;
            while (l<r){int mid=l+r>>1; if (clockwise(k,U[mid],U[mid+1])==1) l=mid+1; else ans=mid,r=mid;}
            point w2=U[ans]; return mp(w2,w1);
        } else {
            int l=1,r=where2+1,ans=0;
            while (l<r){int mid=l+r>>1; if (clockwise(k,D[mid],D[mid-1])==-1) ans=mid,l=mid+1; else r=mid;}
            point w1=D[ans]; l=where2,r=D.size()-1,ans=D.size()-1;
            while (l<r){int mid=l+r>>1; if (clockwise(k,D[mid],D[mid+1])==-1) l=mid+1; else ans=mid,r=mid;}
            point w2=D[ans]; return mp(w1,w2);
        }
    }
}
struct P3{
    db x,y,z;
    P3 operator + (P3 k1){return (P3){x+k1.x,y+k1.y,z+k1.z};}
    P3 operator - (P3 k1){return (P3){x-k1.x,y-k1.y,z-k1.z};}
    P3 operator * (db k1){return (P3){x*k1,y*k1,z*k1};}
    P3 operator / (db k1){return (P3){x/k1,y/k1,z/k1};}
    db abs2(){return x*x+y*y+z*z;}
    db abs(){return sqrt(x*x+y*y+z*z);}
    P3 unit(){return (*this)/abs();}
    int operator < (const P3 k1) const{
        if (cmp(x,k1.x)!=0) return x<k1.x;
        if (cmp(y,k1.y)!=0) return y<k1.y;
        return cmp(z,k1.z)==-1;
    }
    int operator == (const P3 k1){
        return cmp(x,k1.x)==0&&cmp(y,k1.y)==0&&cmp(z,k1.z)==0;
    }
    void scan(){
        double k1,k2,k3; scanf("%lf%lf%lf",&k1,&k2,&k3);
        x=k1; y=k2; z=k3;
    }
};
P3 cross(P3 k1,P3 k2){return (P3){k1.y*k2.z-k1.z*k2.y,k1.z*k2.x-k1.x*k2.z,k1.x*k2.y-k1.y*k2.x};}
db dot(P3 k1,P3 k2){return k1.x*k2.x+k1.y*k2.y+k1.z*k2.z;}
//p=(3,4,5),l=(13,19,21),theta=85 ans=(2.83,4.62,1.77)
P3 turn3D(db k1,P3 l,P3 p){
    l=l.unit(); P3 ans; db c=cos(k1),s=sin(k1);
    ans.x=p.x*(l.x*l.x*(1-c)+c)+p.y*(l.x*l.y*(1-c)-l.z*s)+p.z*(l.x*l.z*(1-c)+l.y*s);
    ans.y=p.x*(l.x*l.y*(1-c)+l.z*s)+p.y*(l.y*l.y*(1-c)+c)+p.z*(l.y*l.z*(1-c)-l.x*s);
    ans.z=p.x*(l.x*l.z*(1-c)-l.y*s)+p.y*(l.y*l.z*(1-c)+l.x*s)+p.z*(l.x*l.x*(1-c)+c);
    return ans;
}
```

```
typedef vector<P3> VP;
typedef vector<VP> VVP;
db Acos(db x){return acos(max(-(db)1,min(x,(db)1)));}
// 球面距离 , 圆心原点 , 半径 1
db Odist(P3 a,P3 b){db r=Acos(dot(a,b)); return r;}
db r; P3 rnd;
vector<db> solve(db a,db b,db c){
    db r=sqrt(a*a+b*b),th=atan2(b,a);
    if (cmp(c,-r)==-1) return {0};
    else if (cmp(r,c)<=0) return {1};
    else {
        db tr=pi-Acos(c/r); return {th+pi-tr,th+pi+tr};
    }
}
vector<db> jiao(P3 a,P3 b){
    // dot(rd+x*cos(t)+y*sin(t),b) >= cos(r)
    if (cmp(Odist(a,b),2*r)>0) return {0};
    P3 rd=a*cos(r),z=a.unit(),y=cross(z,rnd).unit(),x=cross(y,z).unit();
    vector<db> ret = solve(-(dot(x,b)*sin(r)),-(dot(y,b)*sin(r)),-(cos(r)-dot(rd,b)));
    return ret;
}
db norm(db x,db l=0,db r=2*pi){ // change x into [l,r)
    while (cmp(x,l)==-1) x+=(r-l); while (cmp(x,r)>=0) x-=(r-l);
    return x;
}
db disLP(P3 k1,P3 k2,P3 q){
    return (cross(k2-k1,q-k1)).abs()/(k2-k1).abs();
}
db disLL(P3 k1,P3 k2,P3 k3,P3 k4){
    P3 dir=cross(k2-k1,k4-k3); if (sign(dir.abs())==0) return disLP(k1,k2,k3);
    return fabs(dot(dir.unit(),k1-k2));
}
VP getFL(P3 p,P3 dir,P3 k1,P3 k2){
    db a=dot(k2-p,dir),b=dot(k1-p,dir),d=a-b;
    if (sign(fabs(d))==0) return {};
    return {(k1*a-k2*b)/d};
}
VP getFF(P3 p1,P3 dir1,P3 p2,P3 dir2){// 返回一条线
    P3 e=cross(dir1,dir2),v=cross(dir1,e);
    db d=dot(dir2,v); if (sign(abs(d))==0) return {};
    P3 q=p1+v*dot(dir2,p2-p1)/d; return {q,q+e};
}
// 3D Covex Hull Template
db getV(P3 k1,P3 k2,P3 k3,P3 k4){ // get the Volume
    return dot(cross(k2-k1,k3-k1),k4-k1);
}
db rand_db(){return 1.0*rand()/RAND_MAX;}
VP convexHull2D(VP A,P3 dir){
    P3 x={(db)rand(),(db)rand(),(db)rand()}; x=x.unit();
    x=cross(x,dir).unit(); P3 y=cross(x,dir).unit();
    P3 vec=dir.unit()*dot(A[0],dir);
    vector<point>B;
    for (int i=0;i<A.size();i++) B.push_back((point){dot(A[i],x),dot(A[i],y)});
    B=ConvexHull(B); A.clear();
    for (int i=0;i<B.size();i++) A.push_back(x*B[i].x+y*B[i].y+vec);
    return A;
}
namespace CH3{
    VVP ret; set<pair<int,int> >e;
    int n; VP p,q;
    void wrap(int a,int b){
        if (e.find({a,b})==e.end()){
            int c=-1;
            for (int i=0;i<n;i++) if (i!=a&&i!=b){
                if (c==-1||sign(getV(q[c],q[a],q[b],q[i]))>0) c=i;
            }
            if (c!=-1){
                ret.push_back({p[a],p[b],p[c]});
                e.insert({a,b}); e.insert({b,c}); e.insert({c,a});
                wrap(c,b); wrap(a,c);
            }
        }
    }
    VVP ConvexHull3D(VP _p){
        p=q=_p; n=p.size();
        ret.clear(); e.clear();
        for (auto &i:q) i=i+(P3){rand_db()*1e-4,rand_db()*1e-4,rand_db()*1e-4};
        for (int i=1;i<n;i++) if (q[i].x<q[0].x) swap(p[0],p[i]),swap(q[0],q[i]);
        for (int i=2;i<n;i++) if ((q[i].x-q[0].x)*(q[1].y-q[0].y)>(q[i].y-q[0].y)*(q[1].x-q[0].x)) swap(q[1],q[i]),swap(p[1],
        p[i]);
        wrap(0,1);
        return ret;
```

```
    }
}
VVP reduceCH(VVP A){
    VVP ret; map<P3,VP> M;
    for (VP nowF:A){
        P3 dir=cross(nowF[1]-nowF[0],nowF[2]-nowF[0]).unit();
        for (P3 k1:nowF) M[dir].pb(k1);
    }
    for (pair<P3,VP> nowF:M) ret.pb(convexHull2D(nowF.se,nowF.fi));
    return ret;
}
// 把一个面变成 ( 点 , 法向量 ) 的形式
pair<P3,P3> getF(VP F){
    return mp(F[0],cross(F[1]-F[0],F[2]-F[0]).unit());
}
// 3D Cut 保留 dot(dir,x-p)>=0 的部分
VVP ConvexCut3D(VVP A,P3 p,P3 dir){
    VVP ret; VP sec;
    for (VP nowF: A){
        int n=nowF.size(); VP ans; int dif=0;
        for (int i=0;i<n;i++){
            int d1=sign(dot(dir,nowF[i]-p));
            int d2=sign(dot(dir,nowF[(i+1)%n]-p));
            if (d1>=0) ans.pb(nowF[i]);
            if (d1*d2<0){
                P3 q=getFL(p,dir,nowF[i],nowF[(i+1)%n])[0];
                ans.push_back(q); sec.push_back(q);
            }
            if (d1==0) sec.push_back(nowF[i]); else dif=1;
            dif|=(sign(dot(dir,cross(nowF[(i+1)%n]-nowF[i],nowF[(i+1)%n]-nowF[i])))==-1);
        }
        if (ans.size()>0&&dif) ret.push_back(ans);
    }
    if (sec.size()>0) ret.push_back(convexHull2D(sec,dir));
    return ret;
}
db vol(VVP A){
    if (A.size()==0) return 0; P3 p=A[0][0]; db ans=0;
    for (VP nowF:A)
        for (int i=2;i<nowF.size();i++)
            ans+=abs(getV(p,nowF[0],nowF[i-1],nowF[i]));
    return ans/6;
}
VVP init(db INF) {
    VVP pss(6,VP(4));
    pss[0][0] = pss[1][0] = pss[2][0] = {-INF, -INF, -INF};
    pss[0][3] = pss[1][1] = pss[5][2] = {-INF, -INF, INF};
    pss[0][1] = pss[2][3] = pss[4][2] = {-INF, INF, -INF};
    pss[0][2] = pss[5][3] = pss[4][1] = {-INF, INF, INF};
    pss[1][3] = pss[2][1] = pss[3][2] = {INF, -INF, -INF};
    pss[1][2] = pss[5][1] = pss[3][3] = {INF, -INF, INF};
    pss[2][2] = pss[4][3] = pss[3][1] = {INF, INF, -INF};
    pss[5][0] = pss[4][0] = pss[3][0] = {INF, INF, INF};
    return pss;
}
```

2 Math

2.1 FT_fft_2D

```
const int _M = 2050, _N = N;
template <class V>
struct FT {
   struct cp { double x, y; } tmp[_M * 2 + 5]; cp aa[_M][_M], bb[_M][_M];
   friend cp operator + (cp &a, cp &b) { return cp{ a.x + b.x,a.y + b.y }; }
   friend cp operator - (cp &a, cp &b) { return cp{ a.x - b.x,a.y - b.y }; }
   friend cp operator * (cp &a, cp &b) { return cp{ a.x*b.x - a.y*b.y,a.x*b.y + a.y*b.x }; }
   cp get(double x) { return cp{ cos(x),sin(x) }; }
   void FFT(cp *a, int n, int op) {
      for (int i = (n >> 1), j = 1; j < n; j++) {
         if (i < j) swap(a[i], a[j]);
         int k; for (k = (n >> 1); k&i; i ^= k, k >>= 1); i ^= k;
      }
      for (int m = 2; m <= n; m <<= 1) {
         cp w = get(2 * PI*op / m); tmp[0] = cp{ 1,0 };
         for (int j = 1; j < (m >> 1); j++) tmp[j] = tmp[j - 1] * w;
         for (int i = 0; i < n; i += m)
            for (int j = i; j < i + (m >> 1); j++) {
               cp u = a[j], v = a[j + (m >> 1)] * tmp[j - i];
               a[j] = u + v, a[j + (m >> 1)] = u - v;
            }
```

```
        }
        if (op == -1) rep(i, 0, n) a[i] = cp{ a[i].x / n,a[i].y / n };
    }
    void FFT(cp a[][_M], int n, int op) { rep(i, 0, n) FFT(a[i], n, op); }
    template <class T>
    void Transpose(T a[][_M], int n) {
        rep(i, 0, n) rep(j, 0, i) swap(a[i][j], a[j][i]);
    }
    void Reverse(V a[][_M], int n, int m) {
        rep(i, 0, (n - 1 >> 1) + 1) rep(j, 0, m) swap(a[i][j], a[n - 1 - i][j]);
        rep(i, 0, n) rep(j, 0, (m - 1 >> 1) + 1) swap(a[i][j], a[i][m - 1 - j]);
    }
    void Shift(V a[][_M], int n, int m, int p, int q) {
        rep(i, n, n + p) rep(j, m, m + q) a[i - n][j - m] = a[i][j];
    }
    void In(cp p[][_M], int len, V a[][_M], int n, int m) {
        rep(i, 0, len) rep(j, 0, len) p[i][j] = cp{ i < n&&j < m ? (double)a[i][j] : 0,0 };
    }
    void Out(V a[][_M], int n, int m, cp p[][_M], int len) {
        rep(i, 0, n) rep(j, 0, m) a[i][j] = (V)(p[i][j].x + 0.5) % _p;
    }
    void Multiply(V A[][_M], int n, V B[][_M], int m, V C[][_M], int &len, int op = 0) {
        if (op) Reverse(A, n, n);
        len = 1; while (len < n + m - 1) len <<= 1;
        In(aa, len, A, n, n), In(bb, len, B, m, m), FFT(aa, len, 1), FFT(bb, len, 1);
        Transpose(aa, len), Transpose(bb, len), FFT(aa, len, 1), FFT(bb, len, 1);
        rep(i, 0, len) rep(j, 0, len) aa[i][j] = aa[i][j] * bb[i][j];
        FFT(aa, len, -1), Transpose(aa, len), FFT(aa, len, -1), Out(C, len, len, aa, len);
        if (op) Shift(C, n - 1, n - 1, m, m), len = m, Reverse(A, n, n);
    }
};

inline void Random(int a[][_M], int n) {
    rep(i, 0, n) rep(j, 0, n) a[i][j] = rand();
}
```

2.2 FT_fft_2D_使用说明

```
/*
* FT_fft_2D 使用说明
*
* 【接口说明】
*
* cp get(double x) ：获取一个辐角为 x 的复数
*
* void FFT(cp *a,int n,int op) ：变换接口，注意： op=1 为正卷积， op=-1 为逆卷积
*
* void FFT(cp a[ ][_M],int n,int op) ：行变换接口，逐行进行正 / 逆变换
*
* void Transpose(T a[ ][_M],int n) ：转置接口
*
* void Reverse(V a[ ][_M],int n,int m) ：翻转接口
*
* void Shift(V a[ ][_M],int n,int m,int p,int q) ：移位接口，将矩阵 a 的 (n,m) 整体移位到 (0,0) 长度保留 p 和 q ( 长和宽 )
*
* void In(cp p[][_M],int len,V a[][_M],int n,int m) ：数据填充接口
*
* void Out(V a[][_M],int n,int m,cp p[][_M],int len) ：数据提取接口
*
* void Multiply(V A[][_M],int n,V B[][_M],int m,V C[][_M],int &len,int op=0) ：
*
* 乘法接口，表示 n * n 的矩阵 A 乘上 m*m 的矩阵 B ，结果放到 C 中，规模为 len * len 且计算并返回到 len 中， op=1 为差卷积
*/
```

2.3 FT_fft 基础版本

```
const int _M = N, _N = N;
template <class V>
struct FT {
    struct cp { double x, y; } tmp[_M * 2 + 5];
    friend cp operator + (cp &a, cp &b) { return cp{ a.x + b.x,a.y + b.y }; }
    friend cp operator - (cp &a, cp &b) { return cp{ a.x - b.x,a.y - b.y }; }
    friend cp operator * (cp &a, cp &b) { return cp{ a.x*b.x - a.y*b.y,a.x*b.y + a.y*b.x }; }
    cp get(double x) { return cp{ cos(x),sin(x) }; }
    vector <cp> aa, bb;
    void FFT(vector<cp> &a, int n, int op) {
        for (int i = (n >> 1), j = 1; j < n; j++) {
            if (i < j) swap(a[i], a[j]);
            int k; for (k = (n >> 1); k&i; i ^= k, k >>= 1); i ^= k;
        }
        for (int m = 2; m <= n; m <<= 1) {
            cp w = get(2 * PI*op / m); tmp[0] = cp{ 1,0 };
```

```
        for (int j = 1; j < (m >> 1); j++) tmp[j] = tmp[j - 1] * w;
        for (int i = 0; i < n; i += m)
           for (int j = i; j < i + (m >> 1); j++) {
              cp u = a[j], v = a[j + (m >> 1)] * tmp[j - i];
              a[j] = u + v, a[j + (m >> 1)] = u - v;
           }
     }
     if (op == -1) rep(i, 0, n) a[i] = cp{ a[i].x / n,a[i].y / n };
  }
  vector<V> multiply(vector<V> A, vector<V> B, int op = 0) {
     if (op) reverse(all(A));
     int lena = A.size(), lenb = B.size(), len = 1;
     while (len < lena + lenb) len <<= 1;
     aa = vector<cp>(len), bb = vector<cp>(len);
     rep(i, 0, lena) aa[i] = cp{ (double)A[i],0 };
     rep(i, 0, lenb) bb[i] = cp{ (double)B[i],0 };
     FFT(aa, len, 1), FFT(bb, len, 1);
     rep(i, 0, len) aa[i] = aa[i] * bb[i];
     FFT(aa, len, -1); A.clear();
     if (!op) rep(i, 0, len) A.pb((ll)(aa[i].x + 0.5)); else
        rep(i, lena - 1, lena + lenb - 2 + 1) A.pb((ll)(aa[i].x + 0.5));
     return A;
  }
};
```

2.4 NTT_2D

```
typedef vector<vector<int>> vii;

const int _M = N, _N = N;
template <class V>
struct FT {
  vector <V> aa, bb; int _p, K, _m, N; V w[2][_M * 2 + 5], rev[_M * 2 + 5], tmp, w0;
  inline void Init(int _K, int p) { K = _K, _p = p; }
  ll Pow(ll x, ll k, ll _p) { ll ans = 1; for (; k; k >>= 1, x = x*x%_p) if (k & 1) (ans *= x) %= _p; return ans; }
  inline void get_len(int a, int b, int &len, int &L) { len = 1, L = 0; while (len < a + b) len <<= 1, ++L; }
  inline void init_w(int m) {
     N = 1 << m, w0 = Pow(3, (_p - 1) / N, _p); w[0][0] = w[1][0] = 1;
     rep(i, 1, N) w[0][i] = w[1][N - i] = (ll)w[0][i - 1] * w0%_p;
     rep(i, 1, N) rev[i] = (rev[i >> 1] >> 1) | (i & 1) << m - 1;
  }
  inline void FFT(vector<V>& A, int m, int op) {
     if (m != _m) init_w(_m = m);
     rep(i, 0, N) if (i < rev[i]) swap(A[i], A[rev[i]]);
     for (int i = 1; i < N; i <<= 1)
        for (int j = 0, t = N / (i << 1); j < N; j += i << 1)
           for (int k = j, l = 0; k < j + i; k++, l += t) {
              V x = A[k], y = (ll)w[op][l] * A[k + i] % _p;
              A[k] = (x + y) % _p, A[k + i] = (x - y + _p) % _p;
           }
     if (op) { tmp = Pow(N, _p - 2, _p); rep(i, 0, N) A[i] = 1ll * A[i] * tmp%_p; }
  }
  inline void multiply(const vector<V>& A, const vector<V>& B, vector<V> *C) {
     int lena = A.size(), lenb = B.size(), len = 1, L = 0; aa = A, bb = B;
     get_len(lena, lenb, len, L), aa.resize(len), bb.resize(len);
     FFT(aa, L, 0), FFT(bb, L, 0), (*C).resize(len);
     rep(i, 0, len) (*C)[i] = (ll)aa[i] * bb[i] % _p;
     FFT(*C, L, 1); if (K < len - 1) (*C).resize(K + 1);
  }
};

struct Matrix {
  int n, m; vii a;
  inline void Set_m(int _m, int x = 0) { m = _m; rep(i, 0, n) a[i].resize(m, x); }
  inline void Set_n(int _n) { n = _n, a.resize(n); }
  inline void Set(int _n, int _m, int x = 0) { Set_n(_n), Set_m(_m, x); }
  Matrix(int n = 0, int m = 0, int x = 0) :n(n), m(m) {
     a.clear(), a.resize(n); Set_m(m, x);
  }
  inline void Transpose() {
     Matrix t = Matrix(m, n, 0);
     rep(i, 0, n) rep(j, 0, m) t.a[j][i] = a[i][j];
     *this = t;
  }
  inline void Reverse() {
     Matrix t = Matrix(n, m, 0);
     rep(i, 0, n) rep(j, 0, m) t.a[n - 1 - i][m - 1 - j] = a[i][j];
     *this = t;
  }
  inline void Shift(int x, int y) {
     rep(i, x, n - 1 + x + 1) rep(j, y, m - 1 + y + 1) a[i - x][j - y] = (i < n&&j < m) ? a[i][j] : 0;
  }
```

```
    inline void FFT(FT<int> &T, int len, int op) {
        if (!op) Set_m(1 << len, 0);
        rep(i, 0, n) T.FFT(a[i], len, op);
    }
    inline void print() const;
    inline void Normalize(int _p);
    inline void Random();
};

inline void Matrix::print() const {
    printf("\n\n\n\n\nn => %d     m => %d\n", n, m);
    debug_arr2(a, n - 1, m - 1);
}

inline void Matrix::Normalize(int _p) {
    rep(i, 0, n) rep(j, 0, m) if (a[i][j] < 0) a[i][j] += _p;
}

inline void Matrix::Random() {
    rep(i, 0, n) rep(j, 0, m) a[i][j] = (rand() << 15) + rand();
}

inline bool operator==(const Matrix &A, const Matrix &B) {
    if (A.n != B.n || A.m != B.m) return 0;
    rep(i, 0, A.n) rep(j, 0, A.m) if (A.a[i][j] != B.a[i][j]) return 0;
    return 1;
}

struct Calculator {
    Matrix aa, bb, cc; FT<int> T; int len, L, _p;
    inline void Init(int p) { _p = p; }
    inline void Multiply(const Matrix &A, const Matrix &B, Matrix &C, int op = 0) {
        aa = A, bb = B; if (op) aa.Reverse(); T.get_len(A.m, B.m, len, L), T.Init(cc.m = A.n + B.n - 1, _p);
        aa.FFT(T, L, 0), bb.FFT(T, L, 0), aa.Transpose(), bb.Transpose(), cc.Set_n(aa.n);
        rep(i, 0, aa.n) T.multiply(aa.a[i], bb.a[i], &cc.a[i]);
        cc.Transpose(), cc.FFT(T, L, 1), cc.Set_m(A.m + B.m - 1), C = cc;
        if (op) C.Shift(A.n - 1, A.m - 1);
    }
    inline void add(int &x, int y) { x += y, x %= _p; }
    inline int mul(int x, int y) { return (ll)x*y%_p; }
    inline void Multiply_B(const Matrix &A, const Matrix &B, Matrix &C) {
        C.Set(A.n + B.n - 1, A.m + B.m - 1);
        rep(xa, 0, A.n) rep(ya, 0, A.m) rep(xb, 0, B.n) rep(yb, 0, B.m)
            add(C.a[xa + xb][ya + yb], mul(A.a[xa][ya], B.a[xb][yb]));
    }
    inline void Multiply_B_sub(const Matrix &A, const Matrix &B, Matrix &C) {
        C.Set(A.n + B.n - 1, A.m + B.m - 1);
        rep(xa, 0, A.n) rep(ya, 0, A.m) rep(xb, xa, B.n) rep(yb, ya, B.m)
            add(C.a[xb - xa][yb - ya], mul(A.a[xa][ya], B.a[xb][yb]));
    }
};
```

2.5 Poly_扩展版

```
const int _N=200005; ll inv[_N<<2],fac[_N<<2],fac_inv[_N<<2];
inline ll add(ll x,ll y) { x+=y; return x%P; }
inline ll mul(ll x,ll y) { return (ll)x*y%P; }
inline ll Pow(ll x,ll k) { ll ans=1; for (;k;k>>=1,x=x*x%P) if (k&1) (ans*=x)%=P; return ans; }
inline void init_inv(int n) { inv[1]=1; rep(i,2,n+1) inv[i]=mul(P-P/i,inv[P%i]); }
inline void init_fac(int n) {
    fac[0]=fac_inv[0]=1;
    rep(i,1,n+1) fac[i]=mul(fac[i-1],i),fac_inv[i]=mul(fac_inv[i-1],inv[i]);
}

template <class V>
struct FT{
    int n,nn; V w[2][_N<<2],rev[_N<<2],tmp;
    inline int init_len(int _n) { for (n=1; n<=_n; n<<=1); return n; }
    inline int Init(int _n) {
        init_len(_n); if (n==nn) return n; nn=n;
        V w0=Pow(3,(P-1)/n); w[0][0]=w[1][0]=1;
        rep(i,1,n) w[0][i]=w[1][n-i]=mul(w[0][i-1],w0);
        rep(i,0,n) rev[i]=(rev[i>>1]>>1)|((i&1)*(n>>1)); return n;
    }
    void FFT(V A[],int op){
        rep(i,0,n) if (i<rev[i]) swap(A[i],A[rev[i]]);
        for (int i=1; i<n; i<<=1)
            for (int j=0,t=n/(i<<1); j<n; j+=i<<1)
                for (int k=j,l=0; k<j+i; k++,l+=t) {
                    V x=A[k],y=mul(w[op][l],A[k+i]);
                    A[k]=add(x,y),A[k+i]=add(x-y,P);
                }
```

```
        if (op) { tmp=inv[n]; rep(i,0,n) A[i]=mul(A[i],tmp); }
    }
};

template <class V>
struct Calculator{
    FT<V> T; V X[_N<<2],Y[_N<<2],A[_N<<2],B[_N<<2],C[_N<<2];
    inline void Fill(V a[],V b[],int n,int len) {
        if (a!=b) memcpy(a,b,sizeof(V)*n); fill(a+n,a+len,0);
    }
    inline void Add(V a[],int n,V b[],int m,V c[],int t=1) {
        n=max(n,m); rep(i,0,n) c[i]=add(a[i],t*b[i]);
    }
    inline void Dot_Mul(V a[],V b[],int len,V c[]) {
        rep(i,0,len) c[i]=mul(a[i],b[i]);
    }
    inline void Dot_Mul(V a[],int len,V v,V c[]) {
        rep(i,0,len) c[i]=mul(a[i],v);
    }
    inline void Mul(V a[],int n,V b[],int m,V c[]) {
        int len=T.Init(n+m-1); Fill(X,a,n,len),Fill(Y,b,m,len);
        T.FFT(X,0),T.FFT(Y,0),Dot_Mul(X,Y,len,c),T.FFT(c,1);
    }
    inline void Int(V a[],int n,V b[]) {
        per(i,0,n) b[i+1]=mul(a[i],inv[i+1]); b[0]=0;
    }
    inline void Der(V a[],int n,V b[]) {
        rep(i,1,n) b[i-1]=mul(a[i],i); b[n-1]=0;
    }
    inline void Inv(V a[],int n,V b[]) {
        if (n==1) { b[0]=Pow(a[0],P-2),b[1]=0; return; }
        Inv(a,(n+1)>>1,b); int len=T.Init(2*n-1);
        Fill(X,a,n,len),Fill(b,b,n,len),T.FFT(X,0),T.FFT(b,0);
        rep(i,0,len) b[i]=mul(b[i],2-mul(b[i],X[i]));
        T.FFT(b,1),Fill(b,b,n,len);
    }
    inline void Log(V a[],int n,V b[]) {
        static V A[_N<<2],B[_N<<2];
        Der(a,n,A),Inv(a,n,B),Mul(A,n,B,n,b);
        Int(b,n,b),Fill(b,b,n,T.n);
    }
    inline void Exp(V a[],int n,V b[]) {
        if (n==1) { b[0]=exp(a[0]),b[1]=0; return; }
        Exp(a,(n+1)>>1,A),Log(A,n,B),Add(a,n,B,n,B,-1);
        (B[0]+=1)%=P,Mul(A,n,B,n,b),Fill(b,b,n,T.n);
    }
    inline void Sqrt(V a[],int n,V b[]) {
        if (n==1) { b[0]=sqrt(a[0]),b[1]=0; return; }
        Sqrt(a,(n+1)>>1,b),Inv(b,n,B),Mul(a,n,B,n,B);
        Add(b,n,B,n,b),Dot_Mul(b,n,inv[2],b),Fill(b,b,n,T.n);
    }
    inline void Power(V a[],int n,ll k,V b[]) {
        Log(a,n,C),Dot_Mul(C,n,k,C),Exp(C,n,b),Fill(b,b,n,T.n);
    }
    inline V Lagrange(V a[],int n,int k) {
        Inv(a,n,A),Power(A,n,k,B); return mul(B[k-1],inv[k]);
    }
    inline void Div(V a[],int n,V b[],int m,V d[],V r[]) {
        int len=T.init_len(2*n-1); Fill(A,a,n,len),Fill(B,b,m,len);
        reverse(A,A+n),reverse(B,B+m),Inv(B,n-m+1,Y);
        Mul(A,n,Y,n,d),Fill(d,d,n-m+1,len),reverse(d,d+n-m+1);
        reverse(B,B+m),Fill(A,d,n-m+1,len);
        Mul(A,n,B,n,r),Add(a,n,r,n,r,-1),Fill(r,r,n,len);
    }
    inline void Sinh(V a[],int n,V b[]) {
        Exp(a,n,b); for (int i=0; i<n; i+=2) b[i]=0;
    }
    inline void Cosh(V a[],int n,V b[]) {
        Exp(a,n,b); for (int i=1; i<n; i+=2) b[i]=0;
    }
    inline void Dirichlet_Mul(V a[],int n,V b[],int m,V c[],int L) {
        int len=min((ll)n*m,L); Fill(c,c,0,L+1);
        rep(i,1,n+1) for (int j=1; j<=m && (ll)i*j<=len; j++)
            c[i*j]=add(c[i*i],mul(a[i],b[j]));
    }
    inline void Der_k(V a[],int n,int k,V b[]) {
        Der(a,n,b); rep(i,1,k) Der(b,n,b);
    }
    inline void Int_k(V a[],int n,int k,V b[]) {
        Int(a,n,b); rep(i,1,k) Int(b,n,b);
    }
    inline void Grow(V a[],int n,V b[]) {
```

```
      rep(i,0,n) b[i]=mul(a[i],i);
   }
   inline void Grow_k(V a[],int n,int k,V b[]) {
      rep(i,0,n) b[i]=mul(a[i],Pow(i,k));
   }
   inline void Shl(V a[],int n,int k,V b[]) {
      rep(i,k,n) b[i-k]=a[i]; Fill(b,b,n-k,n);
   }
   inline void Shr(V a[],int n,int k,V b[]) {
      per(i,k,n) b[i]=a[i-k]; Fill(b,b,0,k);
   }
   inline void To_egf(V a[],int n,V b[]) { Dot_Mul(a,fac,n,b); }
   inline void To_ogf(V a[],int n,V b[]) { Dot_Mul(a,fac_inv,n,b); }
   inline void Bin_Mul(V a[],int n,V b[],int m,V c[]) {
      static V A[_N<<2],B[_N<<2];
      To_ogf(a,n,A),To_ogf(b,m,B),Mul(A,n,B,m,c),To_egf(c,T.n,c);
   }
   inline void POW(V a[],int n,ll k,V b[],int t=0) {
      if (k*t>=n || !k) { Fill(b,b,0,n),b[0]=!k; return; }
      if (t) Shl(a,n,t,a); Power(a,n-t,k,b),Shr(b,n,k*t,b);
   }
   inline void Reverse(V a[],int n,V b[]) { reverse_copy(a,a+n,b); }
   inline void Init_Com_Num_H_B(V a[],int n,ll k) {
      a[0]=1; rep(i,1,n) a[i]=mul(a[i-1],inv[i]*(k-i+1)%P);
   }
   inline void Init_Com_Num_L_B(V a[],int n,ll k) {
      a[0]=1; rep(i,1,n) a[i]=mul(a[i-1],inv[i]*(k+i)%P);
   }
   inline void Pre_Sum(V a[],int n,V b[]) {
      b[0]=a[0]; rep(i,1,n) b[i]=add(b[i-1],a[i]);
   }
   inline void Pre_Sum_k(V a[],int n,ll k,V b[]) {
      Init_Com_Num_L_B(b,n,k-1),Mul(a,n,b,n,b);
   }
   inline void Fly(V a[],int n,ll k,V b[]) {
      k%=P; for (int i=0,t=1; i<n; ++i,t=mul(t,k)) b[i]=mul(a[i],t);
   }
   inline void Crossify(V a[],int n) { Fly(a,n,-1,a); }
   inline void Diff(V a[],int n,V b[]) {
      rep(i,0,n-1) b[i]=a[i+1]-a[i]; b[n-1]=-a[n-1];
   }
   inline void Diff_k(V a[],int n,int k,V b[]) {
      Init_Com_Num_H_B(b,k+1,k),Crossify(b,k+1),Mul(a,n,b,k+1,b),Shl(b,n+k,k,b);
   }
   inline void Get_all_one(V a[],int n) { rep(i,0,n) a[i]=1; }
   inline void Get_exp_x(V a[],int n) { Fill(a,fac_inv,n,n); }
   inline void Get_log_1_add_x(V a[],int n) {
      a[0]=0; int t=1; rep(i,1,n) a[i]=t*inv[i],t=-t;
   }
   inline void Init_Bell_Num(V a[],int n) {
      Get_exp_x(C,n),C[0]=0,Exp(C,n,a),To_egf(a,n,a);
   }
   inline void Init_Bernoulli_Num(V a[],int n) {
      Get_exp_x(C,n+1),Shl(C,n+1,1,C),Inv(C,n,a),To_egf(a,n,a);
   }
   inline V Get_Num_Power_Sum(ll n,int k) {
      n%=P; V ans=0; static V C[_N<<2];
      Init_Com_Num_H_B(C,k+2,k+1),Init_Bernoulli_Num(B,k+1);
      for (int i=1,t=n*(k&1?-1:1); i<=k+1; ++i,t=mul(t,-n))
         ans=add(ans,mul(C[i],B[k+1-i])*t%P);
      ans=mul(ans,inv[k+1]); return ans;
   }
   inline void Init_Stiriling_Num_2_H_B(V a[],int n,ll k) {
      k%=P-1; static V A[_N<<2],B[_N<<2];
      rep(i,0,n) A[i]=(i&1)?-1:1,B[i]=Pow(i,k);
      Bin_Mul(A,n,B,n,a),To_ogf(a,n,a);
   }
   inline void Init_Stiriling_Num_2_L(V a[],int n,int k) {
      static V A[_N<<2]; Get_exp_x(A,n+k),A[0]=0,POW(A,n+k,k,a,1);
      Dot_Mul(a,n+k,fac_inv[k],a),To_egf(a,n+k,a),Shl(a,n+k,k,a);
   }
   inline void Init_Stiriling_Num_1_L(V a[],int n,int k) {
      static V A[_N<<2]; Get_log_1_add_x(A,n+k),POW(A,n+k,k,a,1);
      Dot_Mul(a,n+k,((k&1)?-1:1)*fac_inv[k],a);
      To_egf(a,n+k,a),Crossify(a,n+k),Shl(a,n+k,k,a);
   }
   inline void Mod_p(V a[],int n) {
      rep(i,0,n) a[i]=(a[i]%P+P)%P;
   }
};
```

2.6 Poly_标准版

```
const int _N=200005; ll inv[_N<<2];
inline ll add(ll x,ll y) { x+=y; return x%P; }
inline ll mul(ll x,ll y) { return (ll)x*y%P; }
inline ll Pow(ll x,ll k) { ll ans=1; for (;k;k>>=1,x=x*x%P) if (k&1) (ans*=x)%=P; return ans; }
inline void init_inv(int n) { inv[1]=1; rep(i,2,n+1) inv[i]=mul(P-P/i,inv[P%i]); }

template <class V>
struct FT{
   int n,nn; V w[2][_N<<2],rev[_N<<2],tmp;
   inline int init_len(int _n) { for (n=1; n<=_n; n<<=1); return n; }
   inline int Init(int _n) {
      init_len(_n); if (n==nn) return n; nn=n;
      V w0=Pow(3,(P-1)/n); w[0][0]=w[1][0]=1;
      rep(i,1,n) w[0][i]=w[1][n-i]=mul(w[0][i-1],w0);
      rep(i,0,n) rev[i]=(rev[i>>1]>>1)|((i&1)*(n>>1)); return n;
   }
   void FFT(V A[],int op){
      rep(i,0,n) if (i<rev[i]) swap(A[i],A[rev[i]]);
      for (int i=1; i<n; i<<=1)
         for (int j=0,t=n/(i<<1); j<n; j+=i<<1)
            for (int k=j,l=0; k<j+i; k++,l+=t) {
               V x=A[k],y=mul(w[op][l],A[k+i]);
               A[k]=add(x,y),A[k+i]=add(x-y,P);
            }
      if (op) { tmp=inv[n]; rep(i,0,n) A[i]=mul(A[i],tmp); }
   }
};

template <class V>
struct Calculator{
   FT<V> T; V X[_N<<2],Y[_N<<2],A[_N<<2],B[_N<<2],C[_N<<2];
   inline void Fill(V a[],V b[],int n,int len) {
      if (a!=b) memcpy(a,b,sizeof(V)*n); fill(a+n,a+len,0);
   }
   inline void Add(V a[],int n,V b[],int m,V c[],int t=1) {
      n=max(n,m); rep(i,0,n) c[i]=add(a[i],t*b[i]);
   }
   inline void Dot_Mul(V a[],V b[],int len,V c[]) {
      rep(i,0,len) c[i]=mul(a[i],b[i]);
   }
   inline void Dot_Mul(V a[],int len,V v,V c[]) {
      rep(i,0,len) c[i]=mul(a[i],v);
   }
   inline void Mul(V a[],int n,V b[],int m,V c[]) {
      int len=T.Init(n+m-1); Fill(X,a,n,len),Fill(Y,b,m,len);
      T.FFT(X,0),T.FFT(Y,0),Dot_Mul(X,Y,len,c),T.FFT(c,1);
   }
   inline void Int(V a[],int n,V b[]) {
      per(i,0,n) b[i+1]=mul(a[i],inv[i+1]); b[0]=0;
   }
   inline void Der(V a[],int n,V b[]) {
      rep(i,1,n) b[i-1]=mul(a[i],i); b[n-1]=0;
   }
   inline void Inv(V a[],int n,V b[]) {
      if (n==1) { b[0]=Pow(a[0],P-2),b[1]=0; return; }
      Inv(a,(n+1)>>1,b); int len=T.Init(2*n-1);
      Fill(X,a,n,len),Fill(b,b,n,len),T.FFT(X,0),T.FFT(b,0);
      rep(i,0,len) b[i]=mul(b[i],2-mul(b[i],X[i]));
      T.FFT(b,1),Fill(b,b,n,len);
   }
   inline void Log(V a[],int n,V b[]) {
      static V A[_N<<2],B[_N<<2];
      Der(a,n,A),Inv(a,n,B),Mul(A,n,B,n,b);
      Int(b,n,b),Fill(b,b,n,T.n);
   }
   inline void Exp(V a[],int n,V b[]) {
      if (n==1) { b[0]=exp(a[0]),b[1]=0; return; }
      Exp(a,(n+1)>>1,A),Log(A,n,B),Add(a,n,B,n,B,-1);
      (B[0]+=1)%=P,Mul(A,n,B,n,b),Fill(b,b,n,T.n);
   }
   inline void Sqrt(V a[],int n,V b[]) {
      if (n==1) { b[0]=sqrt(a[0]),b[1]=0; return; }
      Sqrt(a,(n+1)>>1,b),Inv(b,n,B),Mul(a,n,B,n,B);
      Add(b,n,B,n,b),Dot_Mul(b,n,inv[2],b),Fill(b,b,n,T.n);
   }
   inline void Power(V a[],int n,ll k,V b[]) {
      Log(a,n,C),Dot_Mul(C,n,k,C),Exp(C,n,b),Fill(b,b,n,T.n);
   }
   inline V Lagrange(V a[],int n,int k) {
      Inv(a,n,A),Power(A,n,k,B); return mul(B[k-1],inv[k]);
```

```
    }
    inline void Div(V a[],int n,V b[],int m,V d[],V r[]) {
        int len=T.init_len(2*n-1); Fill(A,a,n,len),Fill(B,b,m,len);
        reverse(A,A+n),reverse(B,B+m),Inv(B,n-m+1,Y);
        Mul(A,n,Y,n,d),Fill(d,d,n-m+1,len),reverse(d,d+n-m+1);
        reverse(B,B+m),Fill(A,d,n-m+1,len);
        Mul(A,n,B,n,r),Add(a,n,r,n,r,-1),Fill(r,r,n,len);
    }
    inline void Sinh(V a[],int n,V b[]) {
        Exp(a,n,b); for (int i=0; i<n; i+=2) b[i]=0;
    }
    inline void Cosh(V a[],int n,V b[]) {
        Exp(a,n,b); for (int i=1; i<n; i+=2) b[i]=0;
    }
    inline void Dirichlet_Mul(V a[],int n,V b[],int m,V c[],int L) {
        int len=min((ll)n*m,L); Fill(c,c,0,L+1);
        rep(i,1,n+1) for (int j=1; j<=m && (ll)i*j<=len; j++)
            c[i*j]=add(c[i*i],mul(a[i],b[j]));
    }
};
```

2.7 一维 FFT_单模式串_模式串带通配符匹配

```
struct cp { double x, y; };
inline cp operator + (cp &a, cp &b) { return cp{ a.x + b.x,a.y + b.y }; }
inline cp operator - (cp &a, cp &b) { return cp{ a.x - b.x,a.y - b.y }; }
inline cp operator * (cp &a, cp &b) { return cp{ a.x*b.x - a.y*b.y,a.x*b.y + a.y*b.x }; }
inline cp get(double x) { return cp{ cos(x),sin(x) }; }
inline ostream& operator<<(ostream &out, const cp &t) { out << "(" << t.x << "," << t.y << ")"; return out; }

const int _M = 1 << 18, _N = N;
struct FT {
    cp tmp[_M * 2 + 5], aa[_M], bb[_M];
    void FFT(cp *a, int n, int op) {
        for (int i = (n >> 1), j = 1; j < n; j++) {
            if (i < j) swap(a[i], a[j]);
            int k; for (k = (n >> 1); k&i; i ^= k, k >>= 1); i ^= k;
        }
        for (int m = 2; m <= n; m <<= 1) {
            cp w = get(2 * PI*op / m); tmp[0] = cp{ 1,0 };
            for (int j = 1; j < (m >> 1); j++) tmp[j] = tmp[j - 1] * w;
            for (int i = 0; i < n; i += m)
                for (int j = i; j < i + (m >> 1); j++) {
                    cp u = a[j], v = a[j + (m >> 1)] * tmp[j - i];
                    a[j] = u + v, a[j + (m >> 1)] = u - v;
                }
        }
        if (op == -1) rep(i, 0, n) a[i] = cp{ a[i].x / n,a[i].y / n };
    }
    void In(cp p[], int len, cp a[], int n) {
        rep(i, 0, len) p[i] = i < n ? a[i] : cp{ 0,0 };
    }
    void Out(int a[], int n, cp p[], int len) {
        rep(i, 0, n) a[i] = (int)(p[i].x + eps);
    }
    void Shift(int a[], int n, int p) { rep(i, n, n + p) a[i - n] = a[i]; }
    void Multiply(cp A[], int n, cp B[], int m, int C[], int &len, int op = 0) {
        if (op) reverse(A, A + n);
        len = 1; while (len < n + m - 1) len <<= 1;
        In(aa, len, A, n), In(bb, len, B, m), FFT(aa, len, 1), FFT(bb, len, 1);
        rep(i, 0, len) aa[i] = aa[i] * bb[i];
        FFT(aa, len, -1), Out(C, n + m - 1, aa, len);
        if (op) Shift(C, n - 1, m), len = m, reverse(A, A + n);
    }
};

void Build(cp A[], int n, char s[], int M, int op, int cc = 'a') {
    rep(i, 0, n) A[i] = (s[i] == '?') ? cp{ 0,0 } : get(2 * PI / M*(s[i] - cc)*op);
}

int n, m, len, tot = 0, tt; char s[N], t[N]; FT T; cp A[_M], B[_M]; int C[_M]; vi ans;

int main() {
    //file_put();

    scanf("%s%s", s, t), n = strlen(s), m = strlen(t);
    rep(i, 0, m) tot += (t[i] != '?');
    Build(A, n, s, 26, 1), Build(B, m, t, 26, -1);
    T.Multiply(B, m, A, n, C, len, 1);
    //debug_arr(C,len-1);
    rep(i, 0, n - m + 1) if (C[i] >= tot) ans.pb(i);
    printf("%d\n", tt = ans.size());
```

```
    rep(i, 0, tt) printf("%d\n", ans[i]);

    return 0;
}
```

2.8 二维 FFT_单模式串_模式串带通配符匹配

```
struct cp { double x, y; };
inline cp operator + (cp &a, cp &b) { return cp{ a.x + b.x,a.y + b.y }; }
inline cp operator - (cp &a, cp &b) { return cp{ a.x - b.x,a.y - b.y }; }
inline cp operator * (cp &a, cp &b) { return cp{ a.x*b.x - a.y*b.y,a.x*b.y + a.y*b.x }; }
inline cp get(double x) { return cp{ cos(x),sin(x) }; }
inline ostream& operator<<(ostream &out, const cp &t) { out << "(" << t.x << "," << t.y << ")"; return out; }

const int _M = 2048, _N = N;
template <class V>
struct FT {
    cp tmp[_M * 2 + 5], aa[_M][_M], bb[_M][_M];
    void FFT(cp *a, int n, int op) {
        for (int i = (n >> 1), j = 1; j < n; j++) {
            if (i < j) swap(a[i], a[j]);
            int k; for (k = (n >> 1); k&i; i ^= k, k >>= 1); i ^= k;
        }
        for (int m = 2; m <= n; m <<= 1) {
            cp w = get(2 * PI*op / m); tmp[0] = cp{ 1,0 };
            for (int j = 1; j < (m >> 1); j++) tmp[j] = tmp[j - 1] * w;
            for (int i = 0; i < n; i += m)
                for (int j = i; j < i + (m >> 1); j++) {
                    cp u = a[j], v = a[j + (m >> 1)] * tmp[j - i];
                    a[j] = u + v, a[j + (m >> 1)] = u - v;
                }
        }
        if (op == -1) rep(i, 0, n) a[i] = cp{ a[i].x / n,a[i].y / n };
    }
    void FFT(cp a[][_M], int n, int op) { rep(i, 0, n) FFT(a[i], n, op); }
    template <class T>
    void Transpose(T a[][_M], int n) {
        rep(i, 0, n) rep(j, 0, i) swap(a[i][j], a[j][i]);
    }
    void Reverse(V a[][_M], int n, int m) {
        rep(i, 0, (n - 1 >> 1) + 1) rep(j, 0, m) swap(a[i][j], a[n - 1 - i][j]);
        rep(i, 0, n) rep(j, 0, (m - 1 >> 1) + 1) swap(a[i][j], a[i][m - 1 - j]);
    }
    void Shift(int a[][_M], int n, int m, int p, int q) {
        rep(i, n, n + p) rep(j, m, m + q) a[i - n][j - m] = a[i][j];
    }
    void In(cp p[][_M], int len, V a[][_M], int n, int m) {
        rep(i, 0, len) rep(j, 0, len) p[i][j] = i < n&&j < m ? a[i][j] : cp{ 0,0 };
    }
    void Out(int a[][_M], int n, int m, cp p[][_M], int len) {
        rep(i, 0, n) rep(j, 0, m) a[i][j] = (int)(p[i][j].x + eps);
    }
    void Multiply(V A[][_M], int n, V B[][_M], int m, int C[][_M], int &len, int op = 0) {
        if (op) Reverse(A, n, n);
        len = 1; while (len < n + m - 1) len <<= 1;
        In(aa, len, A, n, n), In(bb, len, B, m, m), FFT(aa, len, 1), FFT(bb, len, 1);
        Transpose(aa, len), Transpose(bb, len), FFT(aa, len, 1), FFT(bb, len, 1);
        rep(i, 0, len) rep(j, 0, len) aa[i][j] = aa[i][j] * bb[i][j];
        FFT(aa, len, -1), Transpose(aa, len), FFT(aa, len, -1), Out(C, len, len, aa, len);
        if (op) Shift(C, n - 1, n - 1, m, m), len = m, Reverse(A, n, n);
    }
};

void Build(cp A[][_M], int n, int m, char s[][405], int M, int op, int cc = 'a') {
    rep(i, 0, n) rep(j, 0, m) A[i][j] = (s[i][j] == '?') ? cp{ 0,0 } : get(2 * PI / M*(s[i][j] - cc)*op);
}

int n1, n2, m1, m2, nn, mm, len, tot = 0; char s[405][405], t[405][405]; FT<cp> T; cp A[_M][_M], B[_M][_M]; int C[_M][_M];

int main() {
    //file_put();

    scanf("%d%d", &n1, &m1);
    rep(i, 0, n1) scanf("%s", s[i]);
    scanf("%d%d", &n2, &m2), nn = n1 + n2, mm = m1 + m2;
    rep(i, 0, n2) scanf("%s", t[i]);
    rep(i, 0, n2) rep(j, 0, m2) tot += (t[i][j] != '?');
    Build(A, n1, m1, s, 26, 1), Build(B, n2, m2, t, 26, -1);
    rep(i, 0, nn) rep(j, 0, mm) {
        if (i < n1 && j < m1) continue;
        A[i][j] = A[i%n1][j%m1];
    }
```

```
    //debug_arr2(A,nn-1,mm-1);
    //debug_arr2(B,n2-1,m2-1);
    T.Multiply(B, max(n2, m2), A, max(nn, mm), C, len, 1);
    //debug_arr2(C,len-1,len-1);
    rep(i, 0, n1) {
        rep(j, 0, m1) printf("%c", "01"[C[i][j] >= tot]);
        printf("\n");
    }

    return 0;
}
```

2.9 分治 FFT

```
const int M = 1 << 17 << 1;

int f[M], a[M], b[M], T, n, ok, len, ans[M], g[M];
stack<pii> sta;

struct NTT{
    static const int M = :: M, G = 3, P = 998244353; //P = C*2^k + 1
    int N, na, nb, a[M], b[M], w[2][M], rev[M];
    ll kpow(ll a, int b){
        ll c = 1;
        for (; b; b >>= 1,a = a * a % P) if (b & 1) c = c * a %P;
        return c;
    }
    void FFT(int *a, int f){
        rep(i, 0, N) if (i < rev[i]) swap(a[i], a[rev[i]]);
        for (int i = 1; i < N; i <<= 1)
            for (int j = 0, t = N / (i << 1); j < N; j += i << 1)
                for (int k = 0, l = 0, x, y; k < i; k++, l += t)
                    x = (ll) w[f][l] * a[j+k+i] % P, y = a[j+k], a[j+k] = (y+x) % P, a[j+k+i] = (y-x+P) % P;
        if (f) for (int i = 0, x = kpow(N, P-2); i < N; i++) a[i] = (ll)a[i] * x % P;
    }
    void work(){
        rep(i, 0, N){
            int x = i, y = 0;
            for (int k = 1; k < N; x>>=1, k<<=1) (y<<=1) |= x&1;
            rev[i] = y;
        }
        w[0][0] = w[1][0] = 1;
        for (int i = 1, x = kpow(G, (P-1) / N), y = kpow(x, P-2); i < N; i++)
            w[0][i] = (ll)x * w[0][i-1] % P, w[1][i] = (ll) y * w[1][i-1] % P;
    }
    void doit(int *a, int *b, int na, int nb){ // [0, na)
        for (N = 1; N < na + nb - 1; N <<= 1);
        rep(i, na, N) a[i] = 0;
        rep(i, nb, N) b[i] = 0;
        work(), FFT(a,0), FFT(b,0);
        rep(i, 0, N) a[i] = (ll)a[i] * b[i] % P;
        FFT(a, 1);
        //rep(i, 0, N) cout << a[i] << endl;
    }
} ntt;

//f[i] = f[i-1] \cdot (i-1) + \sum_{j=2}^{n-2}{f[j] \cdot f[n-j] \cdot (j-1)}

//f[L, mid] * g[L, mid] -> [mid+1, R]
//f[L, mid] * g[L, min(L - 1, R - l)] -> [mid+1, R]
//f[1, min(L - 1, R - L)] * g[L, mid] -> [mid+1, R]


void solve(int l, int r) {
    if (l == r) {
        f[l] = add(mul(f[l-1], l - 1), f[l]);
        return;
    }
    int mid = l + r >> 1;
    solve(l, mid);
    if (r >= l * 2) {
        int ed = min(mid, r - l);
        rep(i, l, ed + 1) a[i - l] = mul(f[i], i - 1);
        rep(i, l, ed + 1) b[i - l] = f[i];
        ntt.doit(a, b, ed - l + 1, ed - l + 1);
        rep(i, max(2 * l, mid + 1), r+1) if (i <= 2 * ed) f[i] = add(f[i], a[i - 2 * l]); else break;
    }
    if (l + 2 <= r && l + l - 1 >= mid + 1) {
        int ed = min(l - 1, r - l);
        rep(i, l, mid+1) a[i - l] = mul(f[i], i - 1);
        rep(i, 2, ed + 1) b[i - 2] = f[i];
        ntt.doit(a, b, mid + 1 - l, ed + 1 - 2);
```

```
      rep(i, max(l+2, mid+1), r+1) if (i <= mid + ed) f[i] = add(f[i], a[i - l - 2]);else break;
      rep(i, l, mid+1) a[i - l] = f[i];
      rep(i, 2, ed + 1) b[i - 2] = mul(f[i], i - 1);
      ntt.doit(a, b, mid + 1 - l, ed + 1 - 2);
      rep(i, max(l+2, mid+1), r+1) if (i <= mid + ed) f[i] = add(f[i], a[i - l - 2]);else break;
   }
   solve(mid+1, r);
}

int main() {
   freopen("a.in","r",stdin);
   ios::sync_with_stdio(0);
   cin.tie(0);
   //cout << setiosflags(ios::fixed);
   //cout << setprecision(2);
   cin >> T >> n;
   f[0] = 1; f[1] = 2;
   solve(2, n);
   rep(cas, 0, T) {
      ok = 1;
      while (!sta.empty()) sta.pop();
      rep(i, 1, n+1) {
         cin >> len;
         int tmp = 1, cnt = 0;
         if (!ok) continue;
         while (!sta.empty() && sta.top().fi - sta.top().se >= i - len) {
            cnt++;tmp = mul(tmp, ans[sta.top().fi]); sta.pop();
         }
         if (!sta.empty() && i - len + 1 <= sta.top().fi) ok = 0;
         sta.push(mp(i, len));
         ans[i] = mul(tmp, f[cnt]);
      }
      if (!ok || sz(sta) > 1) cout << 0 << endl;else cout << ans[n] << endl;
   }
   return 0;
}
```

2.10 扩展卢卡斯

```
namespace exlucas {
   const int N = 1e6;
   typedef long long ll;
   ll n, m, p;
   inline ll power(ll a, ll b, const ll p = LLONG_MAX) {
      ll ans = 1;
      while (b) {
         if (b & 1)
            ans = ans * a % p;
         a = a * a % p;
         b >>= 1;
      }
      return ans;
   }
   ll fac(const ll n, const ll p, const ll pk) {
      if (!n)
         return 1;
      ll ans = 1;
      for (int i = 1; i < pk; i++)
         if (i % p)
            ans = ans * i % pk;
      ans = power(ans, n / pk, pk);
      for (int i = 1; i <= n % pk; i++)
         if (i % p)
            ans = ans * i % pk;
      return ans * fac(n / p, p, pk) % pk;
   }
   ll exgcd(const ll a, const ll b, ll &x, ll &y) {
      if (!b) {
         x = 1, y = 0;
         return a;
      }
      ll xx, yy, g = exgcd(b, a % b, xx, yy);
      x = yy;
      y = xx - a / b * yy;
      return g;
   }
   ll inv(const ll a, const ll p) {
      ll x, y;
      exgcd(a, p, x, y);
      return (x % p + p) % p;
   }
   ll C(const ll n, const ll m, const ll p, const ll pk) {
```

```
        if (n < m)
            return 0;
        ll f1 = fac(n, p, pk), f2 = fac(m, p, pk), f3 = fac(n - m, p, pk), cnt = 0;
        for (ll i = n; i; i /= p)
            cnt += i / p;
        for (ll i = m; i; i /= p)
            cnt -= i / p;
        for (ll i = n - m; i; i /= p)
            cnt -= i / p;
        return f1 * inv(f2, pk) % pk * inv(f3, pk) % pk * power(p, cnt, pk) % pk;
    }
    ll a[N], c[N];
    int cnt;
    inline ll CRT() {
        ll M = 1, ans = 0;
        for (int i = 0; i < cnt; i++)
            M *= c[i];
        for (int i = 0; i < cnt; i++)
            ans = (ans + a[i] * (M / c[i]) % M * inv(M / c[i], c[i]) % M) % M;
        return ans;
    }
    ll exlucas(const ll n, const ll m, ll p) {
        ll tmp = sqrt(p);
        for (int i = 2; p > 1 && i <= tmp; i++) {
            ll tmp = 1;
            while (p % i == 0)
                p /= i, tmp *= i;
            if (tmp > 1)
                a[cnt] = C(n, m, i, tmp), c[cnt++] = tmp;
        }
        if (p > 1)
            a[cnt] = C(n, m, p, p), c[cnt++] = p;
        return CRT();
    }
    int work() {
        ios::sync_with_stdio(false);
        cin >> n >> m >> p;
        cout << exlucas(n, m, p);
        return 0;
    }
}
```

3 String

3.1 PAM_优化转移_偶回文切割方案数

```
const int M = 26;
struct PAM {
    int s[N], len[N], next[N][M], fail[N], cnt[N], dep[N], id[N], no[N], last, n, p, cur, now; int df[N], slink[N], pre[N]; ll
     ans, dp[N], res[N];
    inline int new_node(int _l) { mem(next[p], 0); cnt[p] = dep[p] = 0, len[p] = _l; return p++; }
    inline void Init() { new_node(p = 0), new_node(s[0] = -1), fail[last = n = 0] = 1; }
    inline int get_fail(int x) { for (; s[n - len[x] - 1] != s[n]; x = fail[x]); return x; }
    inline void I(int c) {
        c -= 'a', s[++n] = c, cur = get_fail(last);
        if (!next[cur][c]) {
            now = new_node(len[cur] + 2);
            fail[now] = next[get_fail(fail[cur])][c];
            next[cur][c] = now;
            dep[now] = dep[fail[now]] + 1; //...
        }
        last = next[cur][c], cnt[last]++; id[n] = last, no[last] = n;
        df[last] = len[last] - len[fail[last]];
        slink[last] = (df[last] == df[fail[last]]) ? slink[fail[last]] : fail[last]; //...
    }
    inline void Insert(char s[], int op = 0, int _n = 0) {
        if (!_n) _n = strlen(s);  if (!op) rep(i, 0, _n) I(s[i]); else per(i, 0, _n) I(s[i]);
    }
    inline void count() { per(i, 0, p) cnt[fail[i]] += cnt[i]; }
    inline void Q() {
        rep(i, 0, n + 2) dp[i] = 0; dp[0] = 1;
        rep(i, 1, n + 1) for (int t = id[i]; len[t] > 0; t = slink[t]) {
            res[t] = dp[i - len[slink[t]] - df[t]];
            if (df[t] == df[fail[t]]) (res[t] += res[fail[t]]) %= P;
            if (!odd(i)) (dp[i] += res[t]) %= P;
        }
        printf("%I64d\n", dp[n]);
    }
};

/*【题目】
```

```
codeforces 932G 求原串偶数段的段式回文切割的方案数目首尾间隔取字母构成新串，转化为求偶回文切割方案数（每段长度为偶数）


*/
```

3.2 PAM_优化转移_最小回文切割段数

```
const int M = 26;
struct PAM {
   int s[N], len[N], next[N][M], fail[N], cnt[N], dep[N], id[N], no[N], last, n, p, cur, now; int df[N], slink[N], dp[N], res
   [N]; ll ans;
   inline int new_node(int _l) { mem(next[p], 0); cnt[p] = dep[p] = 0, len[p] = _l; return p++; }
   inline void Init() { new_node(p = 0), new_node(s[0] = -1), fail[last = n = 0] = 1; }
   inline int get_fail(int x) { for (; s[n - len[x] - 1] != s[n]; x = fail[x]); return x; }
   inline void I(int c) {
      c -= 'a', s[++n] = c, cur = get_fail(last);
      if (!next[cur][c]) {
         now = new_node(len[cur] + 2);
         fail[now] = next[get_fail(fail[cur])][c];
         next[cur][c] = now;
         dep[now] = dep[fail[now]] + 1; //...
      }
      last = next[cur][c], cnt[last]++; id[n] = last, no[last] = n;
      df[last] = len[last] - len[fail[last]];
      slink[last] = (df[last] == df[fail[last]]) ? slink[fail[last]] : fail[last]; //...
   }
   inline void Insert(char s[], int op = 0, int _n = 0) {
      if (!_n) _n = strlen(s);  if (!op) rep(i, 0, _n) I(s[i]); else per(i, 0, _n) I(s[i]);
   }
   inline void count() { per(i, 0, p) cnt[fail[i]] += cnt[i]; }
   inline int Q() {
      rep(i, 0, n + 2) dp[i] = oo; dp[0] = 0;
      rep(i, 1, n + 1) {
         for (int t = id[i]; len[t] > 0; t = slink[t]) {
            res[t] = dp[i - len[slink[t]] - df[t]];
            if (df[t] == df[fail[t]]) res[t] = min(res[t], res[fail[t]]);
            dp[i] = min(dp[i], res[t] + 1);
         }
      }
      return dp[n];
   }
};

/*【题目】求给定串的最小回文切割段数【注意】如果要求段的长度为奇数或偶数，在


1.dp赋值语句前限制，请不要乱改自动机[]如果要求段数为奇数或偶数，
2.和开两维dpresdp[i表示段数为偶数的答案][0]

*/
```

3.3 PAM_单串_支持双端插入

```
const int M = 26;
struct PAM {
   int ss[2 * N], *s, len[N], next[N][M], fail[N], cnt[N], dep[N], id[N], no[N], last[2], n[2], p, cur, now, t; ll sum_cnt,
   ans;
   inline int new_node(int _l) { mem(next[p], 0); cnt[p] = dep[p] = 0, len[p] = _l; return p++; }
   inline void Init() { s = ss + N; new_node(p = 0), new_node(s[0] = s[1] = -1), fail[last[0] = last[1] = n[1] = 0] = n[0] =
   1; /*...*/ sum_cnt = 0; }
   inline int get_fail(int x, int op) { int t = op * 2 - 1; for (; s[n[op] - t*(len[x] + 1)] != s[n[op]]; x = fail[x]);
   return x; }
   inline void I(int c, int op) {
      c -= 'a', t = op * 2 - 1, s[n[op] += t] = c, s[n[op] + t] = -1, cur = get_fail(last[op], op);
      if (!next[cur][c]) {
         now = new_node(len[cur] + 2);
         fail[now] = next[get_fail(fail[cur], op)][c];
         next[cur][c] = now;
         dep[now] = dep[fail[now]] + 1; //...
      }
      last[op] = next[cur][c], cnt[last[op]]++;
      if (len[last[op]] == n[1] - n[0] + 1) last[op ^ 1] = last[op];
      id[n[op]] = last[op]; no[last[op]] = n[op] + (len[last[op]] - 1) * !op; //...
      sum_cnt += dep[last[op]];
   }
   inline void Insert(char s[], int back = 1, int op = 0, int _n = 0) {
      if (!_n) _n = strlen(s);  if (!op) rep(i, 0, _n) I(s[i], back); else per(i, 0, _n) I(s[i], back);
   }
   inline void count() { per(i, 0, p) cnt[fail[i]] += cnt[i]; }
   inline void Q() { /*count();*/ }
};
```

```
/*注：

 1) 支持在线维护本质不同回文串个数 p-2 , 所有回文串个数 sum_cnt , 每个回文串出现的一次起点下标 ( 不保证最左最右 )

2) 注意 id[x] 是不准确的

*/

/*

6
1 a
1 b
2 a
2 c
3
4
8
1 a
2 a
2 a
1 a
3
1 b
3
4

4
5
4
5
11

1 左 2 右
3 p-2
4 sum_cnt

*/
```

3.4 PAM_单串_支持撤销_单加 log

```
const int M = 26;
struct PAM {
   int s[N], len[N], next[N][M], fail[N], cnt[N], id[N], no[N], pre[N], qlink[N], dep[N], last, n, p, cur, now; ll ans;
   inline int new_node(int _l) { mem(next[p], 0); cnt[p] = dep[p] = 0, len[p] = _l; qlink[p] = 0; return p++; }
   inline void Init() { new_node(p = 0), new_node(s[0] = -1), fail[last = n = 0] = 1; /* ... */ }
   inline bool ok(int x, int y, int d = 0) { return s[n - len[x] - d] == s[n - len[y]]; }
   inline int get_fail(int x) { for (; !ok(x, 0, 1); x = qlink[x]) if (ok(fail[x], 0, 1)) return fail[x]; return x; }
   inline void I(int c) {
      c -= 'a', s[++n] = c, cur = get_fail(last);
      if (!next[cur][c]) {
         now = new_node(len[cur] + 2);
         fail[now] = next[get_fail(fail[cur])][c];
         next[cur][c] = now; pre[now] = cur;
         dep[now] = dep[fail[now]] + 1; //...
         if (len[now] > 1) qlink[now] = ok(fail[now], fail[fail[now]]) ? qlink[fail[now]] : fail[fail[now]];
      }
      last = next[cur][c], cnt[last]++; id[n] = last, no[last] = n; //...
   }
   inline void D() { if (p <= 1) return; if (!(--cnt[last])) next[pre[last]][s[n]] = 0, --p; last = id[--n]; }
   inline void Insert(char s[], int op = 0, int _n = 0) {
      if (!_n) _n = strlen(s);  if (!op) rep(i, 0, _n) I(s[i]); else per(i, 0, _n) I(s[i]);
   }
   inline void count() { per(i, 0, p) cnt[fail[i]] += cnt[i]; }
   inline void Q() { /*count();*/ }
};

/*注：



1) 此模板如果不使用回退操作，总复杂度仍为线性，单次插入不超过  log(n) ，回退操作总是常数

2) 若有回退操作，总复杂度退化；使用可持久化线段树优化的 dLink 链接，单次插入可降为 log 字符集

*/
```

3.5 PAM_单串_支持撤销_单加可退化

```
const int M = 26;
struct PAM {
```

```
   int s[N], len[N], next[N][M], fail[N], cnt[N], dep[N], id[N], no[N], pre[N], last, n, p, cur, now; ll ans;
   inline int new_node(int _l) { mem(next[p], 0); cnt[p] = dep[p] = 0, len[p] = _l; return p++; }
   inline void Init() { new_node(p = 0), new_node(s[0] = -1), fail[last = n = 0] = 1; }
   inline int get_fail(int x) { for (; s[n - len[x] - 1] != s[n]; x = fail[x]); return x; }
   inline void I(int c) {
      c -= 'a', s[++n] = c, cur = get_fail(last);
      if (!next[cur][c]) {
         now = new_node(len[cur] + 2);
         fail[now] = next[get_fail(fail[cur])][c];
         next[cur][c] = now; pre[now] = cur;
         dep[now] = dep[fail[now]] + 1; //...
      }
      last = next[cur][c], cnt[last]++; id[n] = last, no[last] = n; //...
   }
   inline void D() { if (p <= 1) return; if (!(--cnt[last])) next[pre[last]][s[n]] = 0, --p; last = id[--n]; }
   inline void Insert(char s[], int op = 0, int _n = 0) {
      if (!_n) _n = strlen(s);  if (!op) rep(i, 0, _n) I(s[i]); else per(i, 0, _n) I(s[i]);
   }
   inline void count() { per(i, 0, p) cnt[fail[i]] += cnt[i]; }
   inline void Q() { /*count();*/ }
};
```

3.6 PAM_多串模式

```
const int M = 26;
struct PAM {
   int s[N], len[N], next[N][M], fail[N], cnt[N][11], dep[N], id[N], no[N], last, n, p, cur, now, str_cnt, d0; ll ans;
   inline int new_node(int _l) { mem(next[p], 0); mem(cnt[p], 0); dep[p] = 0, len[p] = _l; return p++; }
   inline void Init() { new_node(p = 0), new_node(s[0] = -1), fail[last = n = 0] = 1; str_cnt = 0; d0 = -1; }
   inline int get_fail(int x) { for (; s[n - len[x] - 1] != s[n]; x = fail[x]); return x; }
   inline void I(int c) {
      if (c < 0) { s[++n] = c; last = 1; return; }
      c -= 'a', s[++n] = c, cur = get_fail(last);
      if (!next[cur][c]) {
         now = new_node(len[cur] + 2);
         fail[now] = next[get_fail(fail[cur])][c];
         next[cur][c] = now;
         dep[now] = dep[fail[now]] + 1; //...
      }
      last = next[cur][c], cnt[last][str_cnt]++; id[n] = last, no[last] = n; //...
   }
   inline void Insert(char s[], int op = 0, int _n = 0) {
      if (str_cnt) I(--d0); if (!_n) _n = strlen(s);
      if (!op) rep(i, 0, _n) I(s[i]); else per(i, 0, _n) I(s[i]); ++str_cnt;
   }
   inline void count() { per(i, 0, p) rep(j, 0, str_cnt) cnt[fail[i]][j] += cnt[i][j]; }
   inline ll Q() { count(); /* ... */ }
};
```

3.7 PAM_标准版本

```
const int M = 26;
struct PAM {
   int s[N], len[N], next[N][M], fail[N], cnt[N], dep[N], id[N], no[N], last, n, p, cur, now; ll ans;
   inline int new_node(int _l) { mem(next[p], 0); cnt[p] = dep[p] = 0, len[p] = _l; return p++; }
   inline void Init() { new_node(p = 0), new_node(s[0] = -1), fail[last = n = 0] = 1; }
   inline int get_fail(int x) { for (; s[n - len[x] - 1] != s[n]; x = fail[x]); return x; }
   inline void I(int c) {
      c -= 'a', s[++n] = c, cur = get_fail(last);
      if (!next[cur][c]) {
         now = new_node(len[cur] + 2);
         fail[now] = next[get_fail(fail[cur])][c];
         next[cur][c] = now;
         dep[now] = dep[fail[now]] + 1; //...
      }
      last = next[cur][c], cnt[last]++; id[n] = last, no[last] = n; //...
   }
   inline void Insert(char s[], int op = 0, int _n = 0) {
      if (!_n) _n = strlen(s);  if (!op) rep(i, 0, _n) I(s[i]); else per(i, 0, _n) I(s[i]);
   }
   inline void count() { per(i, 0, p) cnt[fail[i]] += cnt[i]; }
   inline void Q() { /*count();*/ }
};
```

3.8 PAM_空间压缩_单链表

```
struct Link {
   struct node { int id, x; node *nxt; }; node *head, *p;  Link() { head = NULL; }
   void I(int id, int x) { head = new node{ id,x,head }; }
   int F(int id) { for (p = head; p; p = p->nxt) if (p->id == id) return p->x; return 0; }
};
```

```
struct PAM {
   int s[N], len[N], fail[N], cnt[N][3], dep[N], id[N], no[N], last, n, p, cur, now, str_cnt, d0, tt; Link next[N]; ll ans;
   inline int new_node(int _l) { mem(cnt[p], 0); dep[p] = 0, len[p] = _l; return p++; }
   inline void Init() { new_node(p = 0), new_node(s[0] = -1), fail[last = n = 0] = 1; str_cnt = 0; d0 = -1; }
   inline int get_fail(int x) { for (; s[n - len[x] - 1] != s[n]; x = fail[x]); return x; }
   inline void I(int c) {
      if (c < 0) { s[++n] = c; last = 1; return; }
      c -= 'a', s[++n] = c, cur = get_fail(last); tt = now = 0;
      if (!(tt = next[cur].F(c))) {
         now = new_node(len[cur] + 2);
         fail[now] = next[get_fail(fail[cur])].F(c);
         next[cur].I(c, now);
         dep[now] = dep[fail[now]] + 1; //...
      }
      last = tt + now, cnt[last][str_cnt]++; id[n] = last, no[last] = n; //...
   }
   inline void Insert(char s[], int op = 0, int _n = 0) {
      if (str_cnt) I(--d0); if (!_n) _n = strlen(s);
      if (!op) rep(i, 0, _n) I(s[i]); else per(i, 0, _n) I(s[i]); ++str_cnt;
   }
   inline void count() { per(i, 0, p) rep(j, 0, str_cnt) cnt[fail[i]][j] += cnt[i][j]; }
   inline ll Q() { count(); /* ... */ }
};
```

3.9 SAM_多串模式_串运行_树上合并_map

```
struct SAM {
   static const int M = 26; ll dp[N << 1], ans; map<int, int> SS[N << 1];
   int go[N << 1][M], pre[N << 1], step[N << 1], rr[N << 1], temp[N << 1], toop[N << 1], id[N << 1], dep[N << 1], str_cnt,
   cnt, S, T, n, p, q, nq;
   inline int h(int c) { return c - 'a'; }
   inline int new_node(int _s, int c) { step[++cnt] = _s, pre[cnt] = dep[cnt] = 0, rr[cnt] = c; /* */mem(go[cnt], 0); return
   cnt; }
   inline void Init() { n = cnt = str_cnt = 0, S = T = new_node(0, 0); }
   inline void I(int c) {
      ++n, c = h(c), p = T; if (!go[p][c]) T = new_node(step[T] + 1, c);
      for (; p && !go[p][c]; p = pre[p]) go[p][c] = T;
      if (!p) pre[T] = S; else {
         q = go[p][c]; int &X = (p == T ? T : pre[T]);
         if (step[p] + 1 == step[q]) X = q; else {
            nq = new_node(step[p] + 1, c);
            rep(j, 0, M) go[nq][j] = go[q][j];
            for (; p && go[p][c] == q; p = pre[p]) go[p][c] = nq;
            pre[nq] = pre[q], X = pre[q] = nq;
         }
      }
      id[n] = T; SS[T][str_cnt]++; //...
   }
   inline void Insert(const char s[], int _n = 0, int op = 0) {
      if (!_n) _n = strlen(s); ++str_cnt; T = S;
      if (!op) rep(i, 0, _n) I(s[i]); else per(i, 0, _n) I(s[i]);
   }
   inline void Go(int &p, int c) { while (p && !go[p][c]) p = pre[p]; if (p) p = go[p][c], ans += dp[p]; }
   inline int Run(const char s[], int _n = 0, int op = 0) {
      if (!_n) _n = strlen(s); int pos = S, p = op*(_n - 1), q = (op ? -1 : _n); op = 1 - 2 * op; ans = 0;
      for (int i = p; i != q; i += op) if (pos) Go(pos, h(s[i])); else break; return pos;
   }
   inline int get_dep(int x) { return (!x || dep[x]) ? dep[x] : dep[x] = get_dep(pre[x]) + 1; }
   inline void Toop() {
      rep(i, 0, n + 1) temp[i] = 0; rep(i, 1, cnt + 1) temp[get_dep(i)]++;
      rep(i, 1, n + 1) temp[i] += temp[i - 1]; per(i, 1, cnt + 1) toop[temp[dep[i]]--] = i;
   }
   inline void Merge(int x, int y) { rep_it(it, SS[y]) SS[x][it->fir] += it->sec; }
   inline void Count() { per(i, 1, cnt + 1) Merge(pre[toop[i]], toop[i]); }
   inline void Q() {
      Toop(), Count(); rep(i, 1, cnt + 1) if (SS[i].size() >= k) dp[i] = step[i] - step[pre[i]]; else dp[i] = 0;
      rep(i, 1, cnt + 1) dp[toop[i]] += dp[pre[toop[i]]]; rep(i, 1, str_cnt + 1) Run(st[i].c_str()), printf("%lld ", ans);
   }//...
};
```

3.10 SAM_广义_trie 树

```
struct SAM {
   static const int M = 26; int go[N << 1][M], pre[N << 1], step[N << 1], rr[N << 1], temp[N << 1], toop[N << 1], num[N <<
   1], dep[N << 1], cnt, S, n, p, q, nq; ll dp[N << 1], ans = 0;
   inline int h(int c) { return c - 'a'; }
   inline int new_node(int _s, int c) { step[++cnt] = _s, pre[cnt] = num[cnt] = 0, rr[cnt] = c; /* */mem(go[cnt], 0); return
   cnt; }
   inline void Init() { n = cnt = 0, S = new_node(0, 0); }
   inline int I(int T, int c) {
      if (go[T][c = h(c)] && step[go[T][c]] == step[T] + 1) return go[T][c];
```

```
      ++n, p = T, T = new_node(step[T] + 1, c);
      for (; p && !go[p][c]; p = pre[p]) go[p][c] = T;
      if (!p) pre[T] = S; else {
         q = go[p][c];
         if (step[p] + 1 == step[q]) pre[T] = q; else {
            nq = new_node(step[p] + 1, c);
            rep(j, 0, M) go[nq][j] = go[q][j];
            for (; p && go[p][c] == q; p = pre[p]) go[p][c] = nq;
            pre[nq] = pre[q], pre[T] = pre[q] = nq;
         }
      }
      num[T]++; return T; //...
   }
   inline void Go(int &p, int c) { while (p && !go[p][c]) p = pre[p]; if (p) p = go[p][c], ans += dp[p]; }
   inline int Run(const char s[], int _n = 0, int op = 0) {
      if (!_n) _n = strlen(s); int pos = S, p = op*(_n - 1), q = (op ? -1 : _n); op = 1 - 2 * op; ans = 0;
      for (int i = p; i != q; i += op) if (pos) Go(pos, h(s[i])); else break; return pos;
   }
   inline int get_dep(int x) { return (!x || dep[x]) ? dep[x] : dep[x] = get_dep(pre[x]) + 1; }
   inline void Toop() {
      rep(i, 0, n + 1) temp[i] = 0; rep(i, 1, cnt + 1) temp[get_dep(i)]++;
      rep(i, 1, n + 1) temp[i] += temp[i - 1]; per(i, 1, cnt + 1) toop[temp[dep[i]]--] = i;
   }
   inline void Count() { per(i, 1, cnt + 1) num[pre[toop[i]]] += num[toop[i]]; }
   inline ll Q() {
      /* Toop(); Count(); */ll ans = 0; rep(i, 1, cnt + 1) ans += step[i] - step[pre[i]]; return ans; //...
   }
};

const int _N = 1e5 + 5, _M = 1e5 + 5;
struct Tu {
   int head[_N], nxt[_M * 2], e[_M * 2], id[_N], n, tot; ll v[_N], w[_M * 2]; SAM M;
   inline void Init(int _n) { n = _n, mem(head, 0), tot = 0; M.Clear(); id[0] = M.S; }
   inline void I(int x, int y, ll _w = 0) { e[++tot] = y, w[tot] = _w; nxt[tot] = head[x], head[x] = tot; }
   void dfs(int x, int f) {
      id[x] = M.I(id[f], v[x]);
      for (int i = head[x]; i; i = nxt[i]) if (e[i] != f) dfs(e[i], x);
   }
   inline void Q() { /* */ }
};

// 此版本注意： N=2*M*V( 结点数 )
```

3.11 SAM_标准版

```
struct SAM {
   static const int M = 26; int go[N << 1][M], pre[N << 1], step[N << 1], rr[N << 1], temp[N << 1], toop[N << 1], num[N <<
   1], id[N << 1], cnt, S, T, n, p, q, nq;
   inline int h(int c) { return c - 'a'; }
   inline int new_node(int _s, int c) { step[++cnt] = _s, pre[cnt] = num[cnt] = 0, rr[cnt] = c; /* */mem(go[cnt], 0); return
   cnt; }
   inline void Init() { n = cnt = 0, S = T = new_node(0, 0); }
   inline void I(int c) {
      ++n, c = h(c), p = T, T = new_node(step[T] + 1, c);
      for (; p && !go[p][c]; p = pre[p]) go[p][c] = T;
      if (!p) pre[T] = S; else {
         q = go[p][c];
         if (step[p] + 1 == step[q]) pre[T] = q; else {
            nq = new_node(step[p] + 1, c);
            rep(j, 0, M) go[nq][j] = go[q][j];
            for (; p && go[p][c] == q; p = pre[p]) go[p][c] = nq;
            pre[nq] = pre[q], pre[T] = pre[q] = nq;
         }
      }
      num[id[n] = T]++; //...
   }
   inline void Insert(char s[], int _n = 0, int op = 0) {
      if (!_n) _n = strlen(s);
      if (!op) rep(i, 0, _n) I(s[i]); else per(i, 0, _n) I(s[i]);
   }
   inline void Toop() {
      rep(i, 0, n + 1) temp[i] = 0; rep(i, 1, cnt + 1) temp[step[i]]++;
      rep(i, 1, n + 1) temp[i] += temp[i - 1]; rep(i, 1, cnt + 1) toop[temp[step[i]]--] = i;
   }
   inline void Count() { per(i, 1, cnt + 1) num[pre[toop[i]]] += num[toop[i]]; }
   inline int Q() { Toop(); return 0; }//...
};

// rr[] 表示 right 集合 , 用来维护需要量
```

4 Zec

4.1 划分数

```
const int N = 1e6 + 7, P = 998244353;
int n, fv[N];
ll f[N];
void init(int n) {
   f[0] = f[1] = 1;
   int m = sqrt(n) + 1;
   rep(i, 1, m+1) fv[i] = i * (3 * i - 1) / 2;
   rep(i, 2, n+1) {
      for(int j = 1; fv[j] <= i; j++) {
         f[i] += j & 1 ? f[i - fv[j]] : -f[i - fv[j]];
         if (fv[j] + j <= i)
            f[i] += j & 1 ? f[i - fv[j] - j] : -f[i - fv[j] - j];
      }
      f[i] = (f[i] % P + P) % P;
   }
}
```

4.2 多项式相关

```
const int M = 1 << 17 << 1;
int rev[M], ta[M], tb[M], tmp[M];

struct NTT{
   static const int G = 3, P = 998244353; //P = C*2^k + 1
   int n, N, na, nb, w[2][M], rev[M];
   void FFT(int *a, int f){
      rep(i, 0, N) if (i < rev[i]) swap(a[i], a[rev[i]]);
      for (int i = 1; i < N; i <<= 1)
         for (int j = 0, t = N / (i << 1); j < N; j += i << 1)
            for (int k = 0, l = 0, x, y; k < i; k++, l += t)
               x = (ll) w[f][l] * a[j+k+i] % P, y = a[j+k], a[j+k] = (y+x) % P, a[j+k+i] = (y-x+P) % P;
      if (f) for (int i = 0, x = kpow(N, P-2); i < N; i++) a[i] = (ll)a[i] * x % P;
   }
   int work(int len){
      for (N = 1; N < len; N <<= 1);
      if (n == N) return n; n = N;
      int d = __builtin_ctz(N);
      w[0][0] = w[1][0] = 1;
      for (int i = 1, x = kpow(G, (P-1) / N); i < N; i++) {
         rev[i] = (rev[i>>1] >> 1) | ((i&1) << (d-1));
         w[0][i] = w[1][N - i] = (ll)x * w[0][i-1] % P;
      }
      return N;
   }
   void doit(int *a, int *b, int na, int nb){ // [0, na)
      work(na + nb - 1);
      rep(i, 0, na) if (a[i] < 0) a[i] += P; rep(i, na, N) a[i] = 0;
      rep(i, 0, nb) if (b[i] < 0) b[i] += P; rep(i, nb, N) b[i] = 0;
      FFT(a, 0), FFT(b, 0);
      rep(i, 0, N) a[i] = (ll)a[i] * b[i] % P;
      FFT(a, 1);
      //rep(i, 0, N) cout << a[i] << endl;
   }
} ntt;



struct poly {
   vi a;
   inline int cs() { return sz(a) - 1; }
   inline int& operator [] (int x) { return a[x]; }
   inline void sc(int x) { a.resize(x+1); }
   inline void clr() {
      int p = cs() + 1;
      while(p && !a[p-1]) --p;
      sc(p-1);
   }
   inline void rev() { reverse(all(a)); }
   inline void dbg() {
      bool fi=0;
      for(int i=cs(); i>=0; i--) {
         a[i]=(a[i]%P+P)%P;
         if(!a[i]) continue;
         if(a[i]>P/2) a[i]-=P;
         if(fi) {
            if(i==0) printf("%+d",a[i]);
            else if(a[i]==1) printf("+");
```

```
            else if(a[i]==-1) printf("-");
            else printf("%+d",a[i]);
         } else {
            if(i==0) printf("%d",a[i]);
            else if(a[i]==1);
            else if(a[i]==-1) printf("-");
            else printf("%d",a[i]);
         }
         if(i>1) printf("x^%d",i);
         else if(i==1) printf("x");
         fi=1;
      }
      if(!fi) printf("0");
      putchar(10);
   }
};

inline poly operator * (poly a, poly b) {
   if(a.cs() < 180 || b.cs() < 180) {
      poly g;
      g.sc(a.cs()+b.cs());
      int *G = &g[0], *A = &a[0], *B = &b[0];
      for(int i=0; i<=a.cs(); i++) {
         register int*h=G+i,j=0;
         register ll x=A[i];
         for(; j<=b.cs(); ++j) h[j]=(h[j]+x*(ll)B[j])%P;
      }
      return g;
   }
   int na = a.cs()+1, nb = b.cs()+1;
   rep(i, 0, na) ta[i] = a[i];
   rep(i, 0, nb) tb[i] = b[i];
   ntt.doit(ta, tb, na, nb);
   a.sc(na+nb-2); rep(i, 0, na+nb-1) a[i] = ta[i]; a.clr();
   return a;
}

inline void ginv(poly &a, int t) {
   if(t == 1) { tb[0] = kpow(a[0], P - 2); return; }
   ginv(a, (t+1)>>1);
   int n = ntt.work(2 * t);
   rep(i, t, n) ta[i] = tb[i] = 0;
   rep(i, 0, t) ta[i] = a[i];
   ntt.FFT(ta, 0);
   ntt.FFT(tb, 0);
   rep(i, 0, n) tb[i] = mul(tb[i], P + 2 - mul(ta[i], tb[i]));
   ntt.FFT(tb, 1);
   rep(i, t, n) tb[i] = 0;
}

inline poly inv(poly a) {
   ginv(a, a.cs()+1);
   per(i, 0, a.cs()+1) a[i] = tb[i]; a.clr();
   return a;
}
inline poly operator + (poly a, poly b) {
   a.sc(max(a.cs(), b.cs()));
   per(i, 0, b.cs()+1) a[i] = add(a[i], b[i]);
   return a;
}
inline poly operator - (poly a, poly b) {
   a.sc(max(a.cs(), b.cs()));
   per(i, 0, b.cs()+1) a[i] = add(a[i], -b[i]); a.clr();
   return a;
}
inline void div(poly a, poly b, poly &d, poly &r) {
   int n = a.cs(), m = b.cs();
   if(n < m) { d.sc(0); d[0] = 0; r = a; return; }
   poly aa = a, bb = b; aa.rev(), bb.rev(), bb.sc(n - m);
   d = aa * inv(bb), d.sc(n - m), d.rev();
   r = a - b * d, r.clr();
}
inline poly operator / (poly a, poly b) {
   poly d, r;
   div(a, b, d, r);
   return d;
}
inline poly operator % (poly a, poly b) {
   a.clr(); b.clr();
   if(a.cs() < b.cs()) return a;
   poly d, r;
   div(a, b, d, r);
```

```
   return r;
}
inline poly dev(poly a) {
   rep(i, 1, a.cs()+1) a[i-1] = mul(a[i], i);
   a.sc(a.cs()-1);
   return a;
}
inline poly inte(poly a) {
   a.sc(a.cs()+1);
   per(i, 1, a.cs()+1) a[i] = mul(a[i-1], rev[i]); a[0] = 0;
   return a;
}
inline ll qz(poly &a, ll x) {
   ll ans = 0;
   per(i, 0, a.cs()+1) ans = (ans * x + a[i]) % P;
   return ans;
}

poly vvs[M];

inline void gvs(int m, int *x, int id) {
   if(m == 1) {
      vvs[id].sc(1), vvs[id][1] = 1, vvs[id][0] = add(0, -*x);
      return;
   }
   int hf = m >> 1;
   gvs(hf, x, id*2);
   gvs(m-hf, x+hf, id*2+1);
   vvs[id] = vvs[id*2] * vvs[id*2+1];
}

inline void gv(poly f, int m, int *x, int *ans, int id) {
   if(f.cs() <= 300) {
      rep(i, 0, m) ans[i] = qz(f, x[i]);
      return;
   }
   int hf = m >> 1;
   gv(f % vvs[id*2], hf, x, ans, id*2);
   gv(f % vvs[id*2+1], m-hf, x+hf, ans+hf, id*2+1);
}

inline vi getv(poly a, vi x) {
   int m = sz(x);
   if(!m) return vi();
   a.clr(); vi ans(m);
   gvs(m, &x[0], 1);
   gv(a % vvs[1], m, &x[0], &ans[0], 1);
   return ans;
}

inline poly comb(int m, int *v, int id) {
   if(m == 1) {
      poly s; s.sc(0); s[0] = *v;
      return s;
   }
   int hf = m >> 1;
   return comb(hf, v, id*2) * vvs[id*2+1] + comb(m-hf, v+hf, id*2+1) * vvs[id*2];
}

inline poly intp(vi x, vi y) {
   int m = sz(x);
   gvs(m, &x[0], 1);
   gv(dev(vvs[1]), m, &x[0], tmp, 1);
   rep(i, 0, m) tmp[i] = mul(kpow(tmp[i], P - 2), y[i]);
   return comb(m, tmp, 1);
}
```

NTT_2D使用说明
接口说明
Poly _ 扩展版 _ 使用说明
【重要数组变量和接口说明】
【不可操作序列修正方法和其他备注】
PAM使用说明
【数组约定】
【函数说明】
【使用方法】
【注意点】

NTT_2D使用说明

接口说明

1）FT部分：

void Init(int _K,int p)：两个参数为用户所期望的结果最大次数，和模数

void init_w(int m)：预处理长度为1<<m的w数组和rev数组

void FFT(vector& A,int m,int op)：变换接口，长度为1<<m，op为0表示普通和卷积，op为1表示差卷积

void multiply(const vector& A,const vector& B,vector *C)：

乘法接口，注意第三个参数传入指针

2）Matrix部分：

void Set_m(int _m,int x=0)：将每列resize为m，不足元素填充为x

void Set_n(int _n)：将行扩充为n

void Set(int _n,int _m,int x=0)：设置矩阵规模

void Transpose()：转置接口

void Reverse()：翻转接口，不光行翻转，列也翻转

void Shift(int x,int y)：移位接口，将(x,y)设置为矩阵左上角元素，整体移位，不足部分填充0

void FFT(FT &T,int len,int op)：批量行变换，对每行进行正/逆变换

void print() const：打印接口

void Normalize(int _p)：规范化接口，保证矩阵中每个元素都严格非负且已模

void Random()：产生随机数，填充此矩阵

==：矩阵可以直接比较判等

3）Calculator部分：

void Init(int p)：预处理模数

void Multiply(const Matrix &A,const Matrix &B,Matrix &C,int op=0):

乘法接口，op=0为和卷积，op=1为差卷积

void Multiply_B(const Matrix &A,const Matrix &B,Matrix &C):

暴力和卷积接口

void Multiply_B_sub(const Matrix &A,const Matrix &B,Matrix &C):

暴力差卷积接口

Poly _ 扩展版 _ 使用说明

【重要数组变量和接口说明】

- inv[],fac[],fac_inv[]：逆元，阶乘和阶乘逆
- init_inv(int n)：预处理逆元
- init_fac(int n)：预处理阶乘和阶乘逆元
- struct FT< V >{}：ntt结构体，此处可以换成fft或者其他模板，但是要求适配接口
  - init_len(int _n)：预处理FT内部长度，产生一个严格大于 _n的2的幂给n
  - Init(int _n)：FT预处理，调用之后才可以用FFT()接口
  - void FFT(V A[],int op)：op=0，把A转化为点值；op=1，为其逆
- **void Fill(V a[],V b[],int n,int len)**：标准填充接口，把b的前n个元素赋值给a的前n个元素，且将a中[n,len)清0
- void Add(V a[],int n,V b[],int m,V c[],int t=1)：长度为n(下标[0,n))的多项式a和长度为m的多项式b，计算c[i]=a[i]+t*b[i]；**[注意]**不保证不对齐位置清0
- void Dot_Mul(V a[],V b[],int len,V c[])：长度相等的两个数组，对应元素点乘给c[]
- void Dot_Mul(V a[],int len,V v,V c[])：每个元素乘v
- **void Mul(V a[],int n,V b[],int m,V c[])**：多项式乘法，用户不需要考虑传入数组非法位置的元素
- void Int(V a[],int n,V b[])：多项式积分，只保留结果的前n项
- void Der(V a[],int n,V b[])：求导
- **void Inv(V a[],int n,V b[])**：多项式求逆，也就是倒数；[注意]常数项必须可逆
- **void Log(V a[],int n,V b[])**：多项式对数，[注意]由于结果常数项为0，所以要求多项式常数项必须为1
- **void Exp(V a[],int n,V b[])**：多项式exp，[注意]ntt版本常数项必须为0(否则正确结果不在模意义整数范围内)，其他版本没有要求，double或者复数版本
- **void Sqrt(V a[],int n,V b[])**：多项式开根，[注意]常数项要可以被开根，如果是ntt，注意边界要求a[0]的模意义二次剩余，如果不存在就无解；double版本a[0]不能为负
- **void Power(V a[],int n,ll k,V b[])**：多项式乘方，[注意]由于依赖log和exp，多项式a[]要求常数项为1，如果以一串0开头，左移之后常数为1，则可以使用POW接口
- **V Lagrange(V a[],int n,int k)**：求多项式a[]的拉格朗日逆或者复合逆(也就是f(g(x))=x，已知一个，求另一个）的x^k前系数，要求常数项可逆

- void Div(V a[],int n,V b[],int m,V d[],V r[])：多项式除法，d是商，r是余数
- void Sinh(V a[],int n,V b[])，void Cosh(V a[],int n,V b[])：双曲正余弦函数
- **void Dirichlet_Mul(V a[],int n,V b[],int m,V c[],int L)**：两个序列的狄利克雷卷积，保留前L项
- void Der_k(V a[],int n,int k,V b[])：k阶导
- void Int_k(V a[],int n,int k,V b[])：k重积分
- void Grow(V a[],int n,V b[])：多项式生长操作，a[i] * = i
- void Grow_k(V a[],int n,int k,V b[])：k次生长
- void Shl(V a[],int n,int k,V b[])，void Shr(V a[],int n,int k,V b[])：左右移k位，传入传出均是前n项有效
- void To_egf(V a[],int n,V b[])：普通型生成函数转化为指数型生成函数，a[i] * = fac[i]
- void To_ogf(V a[],int n,V b[])：前面的逆操作
- **void Bin_Mul(V a[],int n,V b[],int m,V c[])**：求两个序列的二项卷积
- void POW(V a[],int n,ll k,V b[],int t=0)：允许前面有连续t个0的且以1开头的序列传入
- void Reverse(V a[],int n,V b[])：位置翻转操作
- void Init_Com_Num_H_B(V a[],int n,ll k)：预处理组合数第k行前n项，k<=1e18
- void Init_Com_Num_L_B(V a[],int n,ll k)：预处理组合数第k列前n项，k<=1e18，[注意]前面多余的0去掉了，也就是从k行k列开始
- void Pre_Sum(V a[],int n,V b[])：求前缀和
- **void Pre_Sum_k(V a[],int n,ll k,V b[])**：k次前缀和，k<=1e18
- void Fly(V a[],int n,ll k,V b[])：起飞操作，a[i] * = k^i
- void Crossify(V a[],int n)：序列交错化，奇数项符号取反
- void Diff(V a[],int n,V b[])：向前差分，b[i] = a[i+1] - a[i]
- **void Diff_k(V a[],int n,int k,V b[])**：k次差分，有效范围每次减少1个
- void Get_all_one(V a[],int n)：获得全1序列
- **void Get_exp_x(V a[],int n)**：获得e^x展开式
- void Get_log_1_add_x(V a[],int n)：获得log(1+x)展开式
- void Init_Bell_Num(V a[],int n)：预处理Bell数
- void Init_Bernoulli_Num(V a[],int n)：预处理Bernoulli数
- **Get_Num_Power_Sum(ll n,int k)**：获得自然数等幂和S(n,k)=1^k^+2^k^+...n^k^ ， n<=1e18
- **void Init_Stiriling_Num_2_H_B(V a[],int n,ll k)**：预处理第二类Stiriling数第k行前n项(0..n-1),k<=1e18
- void Init_Stiriling_Num_2_L(V a[],int n,int k)：预处理第二类Stiriling数第k列前n项(去掉连续0)
- void Init_Stiriling_Num_1_L(V a[],int n,int k)：预处理第一类Stiriling数第k列前n项
- void Mod_p(V a[],int n)：序列取模，转化为可输出的格式

【不可操作序列修正方法和其他备注】

- 对序列提取一个常数，比如：常数项为-5，不能开根，整个提取-5，再开根，最后乘上根号-5
- 对多项式提取x ^ t，常数项为0，不能pow，那么g(x)^k^ =x^kt^ * f(x)^k^
- 求逆函数，有时候需要结合二次剩余的模板

- 求第一类Stiriling数的一行是x(x+1)..(x+n-1)=x的n次上升阶乘幂展开，需要分治fft/ntt模板或者启发式合并或者倍增
- 分离一个常数，exp中多项式常数要为0，可以把g(x)写成f(x)+c，f(x)常数项为0，求f(x)的exp最后乘上e^c

PAM使用说明

【数组约定】

1.s[]表示当前已经插入到自动机的串，s[0]=-1，真实的字符从s[1]开始，s[]的活动范围是[0,n]

在多串模式中，中间的间隔符，是从-2开始递减的负数，会完全隔绝串间的回文匹配

2.len[i]表示i这个节点表示的回文子串的长度

3.next[i][c] i这个节点，在字符c方向的转移

4.fail[] 失配指针

5.cnt[i] 表示节点i以最长回文后缀出现的前缀下标的个数，通过count()调用，求出每个节点表示的回文串出现的次数

6.dep[i] i这个节点在parent树中的深度，其实际意义是：以i为终止下标的回文后缀的个数

7.id[i] 表示i这个下标（指的是插入串s[]）,所代表的前缀的最长回文后缀在自动机中的节点编号

8.no[i] i这个节点，在插入串s[]出现的最末下标(此处指的是右端点)，用于获取具体的回文串内容

9.last 当前插入的字符生效后，指向最长回文后缀节点，在当前必然是parent树上的叶子节点

10.n 插入的串，字符数0...n

p 任何时刻都表示自动机中的自动机中的最大节点标号+1，节点标号：0..p-1

p-2 为任意时刻本质不同的回文串的个数

str_cnt 多串模式下表示插入的串的个数，串编号从0开始

d0 当前间隔符 M 字符集大小，0..25，默认小写字母

【函数说明】

1.int new_node(int *l) 新建节点，回文长度为*l

2.void Init() 每组数据初始化自动机，O（1）

3.int get_fail(int x) 沿着失配指针，获取最长匹配节点

4.void I(int c) 插入字符c，注意是默认小写字母

5.void Insert(char s[],int _n=0) 用户接口，插入串，多串意义下无须考虑间隔符

6.void count() 树dp，可以计算很多内容，默认计算回文串出现的次数

7.ll Q() 用户接口，主操作，或者称主询问

8.单链表中：查询next[x][y] 等价于 next[x].F(y)；修改next[x][y]=z 等价于 next[x].I(y,z)

【使用方法】

1.每组数据都要首先Init()

2.多串模式下，直接Insert(s，长度)

3.求全局回文子串的个数，cnt[]求和即可；也可以不进行count(),直接每个节点cnt[]*dep[]求和

求任意前缀回文子串的个数，相当于动态即时查询，需要动态维护答案；

考虑每个节点每次的贡献，其实就是深度，所以在cnt[x]++的时候，将深度dep[x]加入答案，即可回答

也可以这样考虑，叠加每个新插入字符新增的回文后缀的数目，这个结构上等价于dep[x]

【注意点】

1.老生常谈，注意多串插入时，N的大小调整，要比总串长大一点

2.count() 调用时，务必要注意：树dp的数组，是否需要开long long；一般，回文串总数目需要long long