NCTU_Yggdarsill

Contents

1 Building Environment

1.1 Default

```cpp
#define F(n) Fi(i,n)
#define Fi(i,n) Fl(i,0,n)
#define Fl(i,l,n) for(int i=(l);i<(int)(n);++i)
#include <bits/stdc++.h>
#include <bits/extc++.h>
// #include <ext/pb_ds/assoc_container.hpp>
// #include <ext/pb_ds/priority_queue.hpp>
using namespace std;
using namespace __gnu_pbds;
const double PI = acos(-1);
main(){
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  cout << fixed << setprecision(7) << PI << endl;
}
```

1.2 Vimrc

```
set tabstop=4
set autoindent

```

```
map <F9> :w<LF>:!g++ -O2 -std=c++11 -o %.out % && echo "----Start----" &&
    ./%.out<LF>
imap <F9> <ESC><F9>
```

2 Convolution

2.1 FFT

```
#include <cstdio>
#include <cstring>
#include <cmath>
const double PI=acos(-1.0);
typedef  struct {
    double  real;
    double  im;
} COMPLEX;
COMPLEX X[66000],Y[66000],A[66000];
COMPLEX EE(COMPLEX a,COMPLEX b)
{
    COMPLEX c;
    c.real=a.real*b.real-a.im*b.im;
    c.im=a.real*b.im+a.im*b.real;
    return c;
}
/* 1 FFT , -1 IFFT */
void fft(COMPLEX x[],int nfft,int isign)
{
    int i,j=0,k;
    COMPLEX t;
    for(i=1, j = nfft / 2;i<nfft-1;i++)
    {
        if(i<j)
        {
            t=x[j];
            x[j]=x[i];
            x[i]=t;
        }
        k=nfft/2;
        while(k<=j)
        {
            j-=k;
            k/=2;
        }
        if (j < k)
            j+=k;
    }
    int le,lei,ip;
    COMPLEX u,w, v;
    for(le=2;le<=nfft;le *= 2)
    {
        lei=le/2;
        w.real=cos(2.0*PI*isign/le);
        w.im=sin(2.0*PI*isign/le);
        for(i=0;i<nfft;i+=le)
        {
            u.real=1.0;
            u.im=0.0;
            for(j = i ; j < i + lei ; ++j)
            {
                ip=j+lei;
                v = x[j];
                t=EE(u, x[ip]);
                x[j].real=v.real+t.real;
                x[j].im=v.im+t.im;
                x[ip].real=v.real-t.real;
                x[ip].im=v.im-t.im;
                u=EE(u,w);
            }
        }
    }
}
void FFT(COMPLEX x[], int nfft)
{
    fft(x,nfft,1);
}
void IFFT(COMPLEX x[],int nfft)
{
    int i;
    fft(x,nfft,-1);

    for(i=0;i<nfft;i++)
    {
        x[i].real /= nfft;
        x[i].im /= nfft;
    }
}
int main(void) {
    int t_num;
    int i,ii,iii;
    int p_num;
    int Nx;
    int NFFT;
    int temp;
    scanf("%d",&t_num);
    for(i=0;i<t_num;i++){
        scanf("%d",&p_num);
        Nx=p_num*2-1;
        NFFT = 2 << (int)log2(Nx);
        for(ii=0;ii<p_num;++ii){
            scanf("%d",&temp);
            X[ii].real=(double)temp;
            X[ii].im=0.0;
        }
        for(iii=0;iii<p_num;++iii){

            scanf("%d",&temp);
            Y[iii].real=(double)temp;
            Y[iii].im=0.0;
```

```
        }
        for(ii=p_num;ii<NFFT;ii++)
        {
            X[ii].real=0.0;
            X[ii].im=0.0;
            Y[ii].real=0.0;
            Y[ii].im=0.0;
        }
        FFT(X,NFFT);
        FFT(Y,NFFT);
        for(ii=0;ii<NFFT;ii++){
            A[ii] = EE(X[ii], Y[ii]);
        }
        IFFT(A,NFFT);
        for(ii=0;ii<Nx;ii++){
            printf("%d ", (int)round(A[ii].real));
        }
        printf("\n");
    }
    return 0;
}
```

2.2 SunMoon FFT

```
#ifndef SUNMOON_FFT
#define SUNMOON_FFT
#include<vector>
#include<complex>
#include<algorithm>
template<typename T,typename VT=std::vector<std::complex<T> > >
struct FFT{
  const T pi;
  FFT(const T pi=acos((T)-1)):pi(pi){}
  inline unsigned int bit_reverse(unsigned int a,int len){
    a=((a&0x55555555U)<<1)|((a&0xAAAAAAAAU)>>1);
    a=((a&0x33333333U)<<2)|((a&0xCCCCCCCCU)>>2);
    a=((a&0x0F0F0F0FU)<<4)|((a&0xF0F0F0F0U)>>4);
    a=((a&0x00FF00FFU)<<8)|((a&0xFF00FF00U)>>8);
    a=((a&0x0000FFFFU)<<16)|((a&0xFFFF0000U)>>16);
    return a>>(32-len);
  }
  inline void fft(bool is_inv,VT &in,VT &out,int N){
    int bitlen=std::__lg(N),num=is_inv?-1:1;
    for(int i=0;i<N;++i)out[bit_reverse(i,bitlen)]=in[i];
    for(int step=2;step<=N;step<<=1){
      const int mh=step>>1;
      for(int i=0;i<mh;++i){
        std::complex<T> wi=exp(std::complex<T>(0,i*num*pi/mh));
        for(int j=i;j<N;j+=step){
          int k=j+mh;
          std::complex<T> u=out[j],t=wi*out[k];
          out[j]=u+t;
          out[k]=u-t;
        }
      }
    if(is_inv)for(int i=0;i<N;++i)out[i]/=N;
  }
};
#endif
```

3 Geometry

3.1 Geometry

```
const double eps = 1e-10;
const double INF = 1.0/0.0;
const double SIDE = 10000;
const double PI = acos(-1.0);
const int MAXN = 500000 + 10;
struct PT{
    double x,y;
    PT(){}
    PT(double x,double y):x(x),y(y){}
    PT operator + (const PT& p)const{
        return PT(x+p.x,y+p.y);
    }
    PT operator - (const PT& p)const{
        return PT(x-p.x,y-p.y);
    }
    PT operator * (double c)const{
        return PT(x*c,y*c);
    }
    PT operator / (double c)const{
        return PT(x/c,y/c);
    }
  PT rot(double a)const{return PT(x*cos(a)-y*sin(a),x*sin(a)+y*cos(a));}
    double operator *(const PT& p)const{
        return x*p.x+y*p.y;
    }
    double operator ^(const PT& p)const{
        return x*p.y-y*p.x;
    }
    bool operator ==(const PT& p)const{
        return fabs(x-p.x)<eps&&fabs(y-p.y)<eps;
    }
  double len2()const{return x*x+y*y;}
  double len()const{return sqrt(len2());}
}poi[MAXN],stk[MAXN];
struct LINE{
    PT a,b;
    double angle;
    LINE(){}
    LINE(PT _a,PT _b):a(_a),b(_b),angle(atan2(_b.y-_a.y,_b.x-_a.x)){}
}line[MAXN],deq[MAXN];
int top;
```

```
inline int ori(const PT& p1,const PT& p2,const PT& p3){
    double a=(p2-p1)^(p3-p1);
    if(a>-eps&&a<eps)return 0;
    return a>0 ? 1:-1;
}
inline bool btw(const PT& p1,const PT& p2,const PT& p3){
    return (p2-p1)*(p3-p1)<eps;
}
//segment intersection
inline bool intersection(const PT& p1,const PT& p2,const PT& p3,const PT& p4)
    {
    int a123=ori(p1,p2,p3);
    int a124=ori(p1,p2,p4);
    int a341=ori(p3,p4,p1);
    int a342=ori(p3,p4,p2);
    if(a123==0&&a124==0)return btw(p1,p3,p4)||btw(p2,p3,p4)||btw(p3,p1,p2)||
    btw(p4,p1,p2);
    return a123*a124 <= 0 && a341*a342 <= 0;
}
inline PT intersectionPoint(const PT& p1,const PT& p2,const PT& p3,const PT&
    p4){
    double a123=(p2-p1)^(p3-p1);
    double a124=(p2-p1)^(p4-p1);
    return (p4*a123-p3*a124)/(a123-a124);
}
//line intersection
inline PT intersectionPoint(const LINE& l1,const LINE& l2){
    PT p1=l1.a,p2=l1.b,p3=l2.a,p4=l2.b;
    double a123=(p2-p1)^(p3-p1);
    double a124=(p2-p1)^(p4-p1);
    return (p4*a123-p3*a124)/(a123-a124);
}
PT foot(const LINE& l,const PT& p){
  PT m(l.b.y-l.a.y,l.a.x-l.b.x);
  return p+m*(l.a-p ^ l.b-p)/((l.b-l.a).len2());
}
PT mirror(const LINE& l,const PT& p){
  PT m(l.b.y-l.a.y,l.a.x-l.b.x);
  return p+m*(l.a-p ^ l.b-p)/((l.b-l.a).len2())*2;
}
//segment-point distance
inline double sp_dis(PT a,PT l1,PT l2){
    if((a-l1)*(l2-l1)<0) return (l1-a).len();
  else if((a-l2)*(l1-l2)<0) return (l2-a).len();
    return fabs(l1-a^l2-a)/((l2-l1).len());
}

struct cir{
    point c;
    double r;
}o[10];
double out_ang(cir a,cir b){    //a.c+(b.c-a.c).unit().rot(ang)*b.r
    return acos((a.r-b.r)/(a.c-b.c).len());
}
double in_ang(cir a,cir b){
    return acos((a.r+b.r)/(a.c-b.c).len());
}
}
int main(){
  double tmp,sum;
  if(fabs(o[i].r-o[j].r)<(o[j].c-o[i].c).len()){
    tmp = out_ang(o[i],o[j]);
    sum = ang_add(cl,tmp);
    pi=o[i].c+point(o[i].r*cos(sum),o[i].r*sin(sum));
    pj=o[j].c+point(o[j].r*cos(sum),o[j].r*sin(sum));
    sum = ang_add(cl,-tmp);
    pi=o[i].c+point(o[i].r*cos(sum),o[i].r*sin(sum));
    pj=o[j].c+point(o[j].r*cos(sum),o[j].r*sin(sum));
  }
  if(o[i].r+o[j].r<(o[j].c-o[i].c).len()){
    tmp = in_ang(o[i],o[j]);
    sum = ang_add(cl,tmp);
    pi=o[i].c+point(o[i].r*cos(sum),o[i].r*sin(sum));
    pj=o[j].c-point(o[j].r*cos(sum),o[j].r*sin(sum));
    sum = ang_add(cl,-tmp);
    pi=o[i].c+point(o[i].r*cos(sum),o[i].r*sin(sum));
    pj=o[j].c-point(o[j].r*cos(sum),o[j].r*sin(sum));
  }
}

inline double dist(const PT& p1,const PT& p2){
    return sqrt((p2-p1)*(p2-p1));
}
inline double tri(const PT& p1,const PT& p2,const PT& p3){
    return fabs((p2-p1)^(p3-p1));
}
inline double getPerimeter(){
    double res=0.0;
    poi[top++]=poi[0];
    for(int i=0;i<top-1;i++)res+=dist(poi[i],poi[i+1]);
    return res;
}
inline double getarea(){
    double res=0.0;
    for(int i=1;i<top-1;i++)res+=tri(poi[0],poi[i],poi[i+1]);
    return 0.5*res;
}

//convex hull
inline bool cmp_convex(const PT &a,const PT &b){
    if(a.x!=b.x)return a.x<b.x;
    return a.y<b.y;
}
inline void convex_hull(PT a[],int &n){
    top=0;
    sort(a,a+n,cmp_convex);
    for(int i=0;i<n;i++){
        while(top>=2&&ori(stk[top-2],stk[top-1],a[i])>=0)top--;
        stk[top++]=a[i];
    }
    for(int i=n-2,t=top+1;i>=0; i--){
        while(top>=t&&ori(stk[top-2],stk[top-1],a[i])>=0)top--;
        stk[top++]=a[i];
```

```
    }
    top--;
    for(int i=0;i<top;i++)poi[i]=stk[i];
}
//half plane intersection
inline bool cmp_half_plane(const LINE &a,const LINE &b){
    if(fabs(a.angle-b.angle)<eps)return ori(a.a,a.b,b.a)<0;
    return a.angle > b.angle;
}
inline void half_plane_intersection(LINE a[],int &n){
    int m=1,front=0,rear=1;
    sort(a,a+n,cmp_half_plane);
    for(int i=1;i<n;i++){
        if(fabs(a[i].angle-a[m-1].angle)>eps)a[m++]=a[i];
    }
    deq[0]=a[0],deq[1]=a[1];
    for(int i=2;i<m;i++){
        while(front<rear&&ori(a[i].a,a[i].b,intersectionPoint(deq[rear],deq[
    rear-1]))<0)rear--;
        while(front<rear&&ori(a[i].a,a[i].b,intersectionPoint(deq[front],deq[
    front+1]))<0)front++;
        deq[++rear]=a[i];
    }
  while(front<rear&&ori(deq[front].a,deq[front].b,intersectionPoint(deq[rear
    ],deq[rear-1]))<0)rear--;
    while(front<rear&&ori(deq[rear].a,deq[rear].b,intersectionPoint(deq[front
    ],deq[front+1]))<0)front++;
    if(front==rear)return;

    top=0;
    for(int i=front;i<rear;i++)poi[top++]=intersectionPoint(deq[i],deq[i+1]);
    if(rear>front+1)poi[top++]=intersectionPoint(deq[front],deq[rear]);
}




//smallest cover rectangle
double ans1,ans2;
void rotating_calipers(){
    ans1=ans2=INF;
    int j=1,k=1,l=1;
    poi[top]=poi[0];
    for(int i=0;i<top;i++){
        while(tri(poi[i],poi[i+1],poi[j])<tri(poi[i],poi[i+1],poi[j+1])) j=(j
    +1)%top;
        while(((poi[i+1]-poi[i])*(poi[k+1]-poi[k]))>eps)k=(k+1)%top;
        if(i==0)l=(k+1)%top;
        while(((poi[i+1]-poi[i])*(poi[l+1]-poi[l]))<-eps)l=(l+1)%top;
        double tmp1 = tri(poi[i],poi[i+1],poi[j])/dist(poi[i],poi[i+1]);
        double tmp2 = (((poi[k]-poi[i])*(poi[i+1]-poi[i]))-((poi[l]-poi[i])*(
    poi[i+1]-poi[i])))/dist(poi[i],poi[i+1]);
        if((tmp1+tmp2)*2.0<ans1)ans1=(tmp1+tmp2)*2.0;
        if(tmp1*tmp2<ans2)ans2=tmp1*tmp2;
    }
}
```

```
int main(){
    int n,m;
    while(~scanf("%d",&n)&&n){
        for(int i=0;i<n;i++)scanf("%lf%lf",&poi[i].x,&poi[i].y);
        convex_hull(poi,n);
        rotating_calipers();
        printf("%.2f %.2f\n",ans2,ans1);
    }
}

inline bool online(const LINE &L,const PT &p){
    return ori(p,L.a,L.b)==0&&btw(p,L.a,L.b);
}
inline bool on_convex(const PT& p){
    for(int i=0;i<top;i++)
        if(p==poi[i])return 1;
    poi[top]=poi[0];
    for(int i=0;i<top;i++){
        line[i].a=poi[i];
        line[i].b=poi[i+1];
    }
    for(int i=0;i<top;i++)
        if(online(line[i],p))return 1;
    return 0;
}
//originally in long long, should be modified
bool in_simple_polygon(PT b[],int k){
  bool flag=false;
  for(int j=0;j<k;j++){
    if(((p-b[j])^(p-b[(j+1)%k]))==0&&(p-b[j])*(p-b[(j+1)%k])<=0){
      flag=true;
      break;
    }
    if((b[j].y<p.y)^(b[(j+1)%k].y<p.y)){
      long long xss=(b[j]-p)^(b[(j+1)%k]-p);
      if((xss<0)^(b[j].y<b[(j+1)%k].y)){
        flag^=1;
      }
    }
  }
  return flag;
}
```

3.2 K-closet Pair

```
#define F(n) Fi(i,n)
#define Fi(i,n) Fl(i,0,n)
#define Fl(i,l,n) for(int i=(l);i<(int)(n);++i)
#include <bits/stdc++.h>
// #include <ext/pb_ds/assoc_container.hpp>
// #include <ext/pb_ds/priority_queue.hpp>
using namespace std;
// using namespace __gnu_pbds;
typedef long long ll;
```

```cpp
struct point {
  point(ll x_ = 0, ll y_ = 0): x(x_), y(y_) {}  ll x, y;
  inline bool operator<(const point &e_) const {
    return (x != e_.x ? x < e_.x : y < e_.y);
  }
  inline friend istream& operator>>(istream &is_, point& e_) {
    is_ >> e_.x >> e_.y;
    return is_;
  }
};
int k;
priority_queue<ll> PQ;
inline ll dist2(const point &e1, const point &e2) {
  ll res = (e1.x-e2.x)*(e1.x-e2.x)+(e1.y-e2.y)*(e1.y-e2.y);
  PQ.push(res);
  if (PQ.size() > k) {
    PQ.pop();
  }
  return res;
}
#define N 500005
point p[N];
queue<point> Q;
ll closet_point(int l, int m, int r, ll delta2) {
  ll xmid = p[m-1].x;
  while (!Q.empty()) {
    Q.pop();
  }
  for (int i = l, j = m ; i < m ; ++i) {
    if ((p[i].x-xmid)*(p[i].x-xmid) >= delta2) {
      continue;
    }
    while (j < r && p[j].y < p[i].y && (p[j].y-p[i].y)*(p[j].y-p[i].y) <
    delta2) {
      if ((p[j].x-xmid)*(p[j].x-xmid) < delta2) {
        Q.push(p[j]);
      }
      ++j;
    }
    while (!Q.empty() && Q.front().y < p[i].y && (Q.front().y-p[i].y)*(Q.
    front().y-p[i].y) > delta2) {
      Q.pop();
    }
    while (!Q.empty()) {
      delta2 = min(delta2, dist2(p[i], Q.front()));
      Q.pop();
    }
  }
  return delta2;
}
ll find_distance(int l, int r) {
  if (r - l <= 3000) {
    ll ans = 0x3f3f3f3f3f3f3f3f;
    for (int i = l ; i < r ; ++i)
      for (int j = i+1 ; j < r ; ++j)
        ans = min(ans, dist2(p[i], p[j]));
    return ans;
  }
  int m = (l+r)/2;
  ll delta2 = min(find_distance(l, m), find_distance(m, r));
  return min(delta2, closet_point(l, m, r, delta2));
}
int main() {
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  int n;
  cin >> n >> k;
  F(n) cin >> p[i];
  sort(p, p+n);
  find_distance(0, n);
  cout << PQ.top() << '\n';
}
```

3.3 MinimumCoveringCircle

```cpp
#define F(n) Fi(i,n)
#define Fi(i,n) Fl(i,0,n)
#define Fl(i,l,n) for(int i=(l);i<(int)(n);++i)
#include <bits/stdc++.h>
using namespace std;
const double eps = 1e-6;
#define x first
#define y second
typedef pair<double, double> point;
inline double dq(const point& p1, const point& p2) {
  return sqrt((p1.x-p2.x)*(p1.x-p2.x)+(p1.y-p2.y)*(p1.y-p2.y));
}
inline point oc(const point& pa, const point& pb, const point& pc) {
  double a, b, c, d, e, f, delta, dx, dy;
  // ax + by = c
  // dx + ey = f
  a = pa.x - pb.x;
  b = pa.y - pb.y;
  c = a*(pa.x+pb.x)/2 + b*(pa.y+pb.y)/2;
  d = pa.x - pc.x;
  e = pa.y - pc.y;
  f = d*(pa.x+pc.x)/2 + e*(pa.y+pc.y)/2;
  delta = a*e-b*d;
  dx = c*e-f*b;
  dy = a*f-d*c;
  return point(dx/delta, dy/delta);
}
inline point enc(const vector<point>& tmp) {
  random_shuffle(tmp.begin(), tmp.end());
  point O = tmp[0];
  double r = 0;
  Fl(i, 1, tmp.size()) if (dq(O, tmp[i]) - r > eps) {
    O = tmp[i], r = 0;
    Fi(j, i) if (dq(O, tmp[j]) - r > eps) {
      O = point((tmp[i].x+tmp[j].x)/2, (tmp[i].y+tmp[j].y)/2);
```

```
      r = dq(O, tmp[j]);
      Fi(k, j) if (dq(O, tmp[k]) - r > eps)
        O = oc(tmp[i], tmp[j], tmp[k]), r = dq(O, tmp[k]);
    }
  }
  return O;
}
int n;
vector<point> v;
int main() {
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  while (cin >> n) {
    if (!n) break;
    v.clear();
    F(n) {
      point tp;
      cin >> tp.x >> tp.y;
      v.push_back(tp);
    }
    point ct = enc(v);
    cout << setprecision(2) << fixed << ct.x << ' ' << ct.y << ' ' << dq(ct,
    v[0]) << '\n';
  }
}
```

4 GNU Black Magic

4.1 GNU Bitwise Operation

```
int __builtin_ffs (unsigned int x)
int __builtin_ffsl (unsigned long)
int __builtin_ffsll (unsigned long long)
// 返回右起第一個1的位置
// Returns one plus the index of the least significant 1-bit of x, or if x is
     zero, returns zero.

int __builtin_clz (unsigned int x)
int __builtin_clzl (unsigned long)
int __builtin_clzll (unsigned long long)
// 返回左起第一個1之前0的個數
// Returns the number of leading 0-bits in x, starting at the most
    significant bit position. If x is 0, the result is undefined.

int __builtin_ctz (unsigned int x)
int __builtin_ctzl (unsigned long)
int __builtin_ctzll (unsigned long long)
// 返回右起第一個1之後的0的個數
// Returns the number of trailing 0-bits in x, starting at the least
    significant bit position. If x is 0, the result is undefined.

int __builtin_popcount (unsigned int x)
int __builtin_popcountl (unsigned long)
int __builtin_popcountll (unsigned long long)
// 返回1的個數
// Returns the number of 1-bits in x.

int __builtin_parity (unsigned int x)
int __builtin_parityl (unsigned long)
int __builtin_parityll (unsigned long long)
// 返回1的個數的奇偶性(1的個數 mod 2的值)
// Returns the parity of x, i.e. the number of 1-bits in x modulo 2.
```

5 Graph

5.1 2-SAT

```
const int MAXN = 2020;

struct TwoSAT{
    static const int MAXv = 2*MAXN;
    vector<int> GO[MAXv],BK[MAXv],stk;
    bool vis[MAXv];
    int SC[MAXv];

    void imply(int u,int v){ // u imply v
        GO[u].push_back(v);
        BK[v].push_back(u);
    }
    int dfs(int u,vector<int>*G,int sc){
        vis[u]=1, SC[u]=sc;
        for (int v:G[u])if (!vis[v])
            dfs(v,G,sc);
        if (G==GO)stk.push_back(u);
    }
    int scc(int n=MAXv){
        memset(vis,0,sizeof(vis));
        for (int i=0; i<n; i++)if (!vis[i])
            dfs(i,GO,-1);
        memset(vis,0,sizeof(vis));
        int sc=0;
        while (!stk.empty()){
            if (!vis[stk.back()])
                dfs(stk.back(),BK,sc++);
            stk.pop_back();
        }
    }
}SAT;

int main(){
    SAT.scc(2*n);
    bool ok=1;
    for (int i=0; i<n; i++){
        if (SAT.SC[2*i]==SAT.SC[2*i+1])ok=0;
```

```
    }
    if (ok){
        for (int i=0; i<n; i++){
            if (SAT.SC[2*i]>SAT.SC[2*i+1]){
                cout << i << endl;
            }
        }
    }
    else puts("NO");
}
```

5.2 Articulation Point

```
void tarjan(int u, int p)
{   // u 為當前點, p 為當前點之母節點
  // cnt 為 DFS 次序
    low[u] = dfn[u] = ++cnt;
    int i, v;
    for (i = 0 ; i < G[u].size() ; ++i) {
        v = G[u][i];
        if (u == rt && !dfn[v]) ++c;
        if (!dfn[v]){
          // (u, v) 為 Tree Edge
            tarjan(v, u);
            low[u] = min(low[u], low[v]);
            // To check if u is AP or not.
            if (dfn[u] <= low[v] && u != rt) ge[u] = 1;
        }
        // 注意不可以同一條邊走兩次，且根節點特判
        if (v != p && p != -1) low[u] = min(low[u], dfn[v]);
    }
}
```

5.3 BCC

```
#include <bits/stdc++.h>
using namespace std;
const int MAXN = 10000;
vector <int> adja[MAXN];
int gcnt, top, timeStamp, dfn[MAXN], low[MAXN], depth[MAXN];
pair<int, int> stk[MAXN],ans[MAXN];
set <int> group[MAXN];
bool cut[MAXN];
void BCC(int now, int nextv){
    int sf, st;
    group[gcnt].clear();
    do{
        sf = stk[top-1].first, st = stk[top-1].second;
        group[gcnt].insert(sf);
        group[gcnt].insert(st);
        --top;
    }while(sf != now || st != nextv);
    ++gcnt;
}
void tarjan(int now, int parent, int d){
    int child = 0;
    dfn[now] = low[now] = ++timeStamp, depth[now] = d;
    for(int i = 0; i < adja[now].size(); i++){
        int nextv = adja[now][i];
        if(nextv == parent) continue;
        if(dfn[nextv] == 0){
            stk[top++] = make_pair(now, nextv);
            tarjan(nextv, now, d+1);
            low[now] = min(low[now], low[nextv]);
            ++child;
            if( (parent != -1 && low[nextv] >= dfn[now]) || (parent == -1 &&
    child >= 2)){
                cut[now] = true;
                if(parent != -1) BCC(now, nextv);
            }
            if(parent == -1) BCC(now, nextv);
        }
        else if(depth[nextv] < depth[now]-1){
            stk[top++] = make_pair(now, nextv);
            low[now] = min(low[now], dfn[nextv]);
        }
    }
}
int main(){
    int n,m,x,y,cnt=0;
    while(~scanf("%d",&n)){
        cnt=timeStamp=top=gcnt=0;
        memset(cut, 0, sizeof(cut));
        memset(dfn, 0, sizeof(dfn));
        for(int i=0;i<n;i++)adja[i].clear();
        for(int i=0;i<n;i++){
            scanf("%d ",&x);
            scanf("(%d)",&m);
            while(m--){
                scanf("%d",&y);
                adja[x].push_back(y);
            }
        }
        for(int i=0;i<n;i++)
            if(dfn[i]==0)tarjan(i, -1, 1);
        for(int i=0;i<gcnt;i++){
            if(group[i].size()==2){
                //critical links
            }
        }
    }
}
```

5.4 Heavy Light Decomposition

```cpp
//with set value && query sum, 1-based with n points
//remove vis in DFS, add it back if something weird happen(I don't think it 's required)
using namespace std;
int sz[N],top[N],up[N],dep[N];
int lightval[N];   //value on light edge
struct node{
  node(){}
  node(int _l,int _r):val(1),l(_l),r(_r),lc(NULL),rc(NULL){}
  int l,r;
  node *lc,*rc;
  int sum;
  int val;
  int qsum(){return val>=0?val*(r-l):sum;}
  void push(){
    if(val>=0){
      sum=val*(r-l);
      lc->val=rc->val=val;
      val=-1;
    }
  }
  void pull(){
    sum=lc->qsum()+rc->qsum();
  }
};
node* tr[N];
node* build(int l,int r){
  node *now=new node(l,r);
  if(r-l>1){
    now->lc=build(l,(l+r)/2);
    now->rc=build((l+r)/2,r);
  }
  return now;
}
//partial
int qry(node* now,int l,int r){
  if(l>=r) return 0;
  if(l==now->l&&r==now->r){
    return now->qsum();
  }
  int m=(now->l+now->r)/2;
  now->push();
  if(l>=m){
    return qry(now->rc,l,r);
  }
  else if(r<=m){
    return qry(now->lc,l,r);
  }
  else return qry(now->lc,l,m)+qry(now->rc,m,r);
}
void set0(node *now,int l,int r){
  if(l>=r) return;
  if(l==now->l&&r==now->r){
    now->val=0;
    return;
  }
  int m=(now->l+now->r)/2;
  now->push();
  if(l>=m){
    set0(now->rc,l,r);
  }
  else if(r<=m){
    set0(now->lc,l,r);
  }
  else{
    set0(now->lc,l,m);
    set0(now->rc,m,r);
  }
  now->pull();
}
vector<int> g[N];
void DFS(int u,int p,int d){
  dep[u]=d;
  sz[u]=1;
  for(int i=0;i<g[u].size();i++){
    int v=g[u][i];
    if(v==p) continue;
    DFS(v,u,d+1);
    sz[u]+=sz[v];
}
}
void decom(int u,int p,bool istop){
  bool ed=true;
  if(istop) top[u]=u,up[u]=p,lightval[u]=1;
  else top[u]=top[p],up[u]=up[p];
  for(int i=0;i<g[u].size();i++){
    int v=g[u][i];
    if(v==p) continue;
    if(sz[v]>=sz[u]-sz[v]){
      decom(v,u,false);
      ed=false;
    }
    else decom(v,u,true);
  }
  if(ed){
    tr[top[u]]=build(dep[top[u]],dep[u]);
  }
}
//global
int qry(int u,int v){
  int res=0;
  while(top[u]!=top[v]){
    if(dep[top[u]]>dep[top[v]]) swap(u,v);
    res+=qry(tr[top[v]],dep[top[v]],dep[v]);
    res+=lightval[top[v]];
    v=up[top[v]];
  }
  if(dep[u]>dep[v]) swap(u,v);
  res+=qry(tr[top[v]],dep[u],dep[v]);
  return res;
}
void set0(int u,int v){
```

```
  while(top[u]!=top[v]){
    if(dep[top[u]]>dep[top[v]]) swap(u,v);
    set0(tr[top[v]],dep[top[v]],dep[v]);
    lightval[top[v]]=0;
    v=up[top[v]];
  }
  if(dep[u]>dep[v]) swap(u,v);
  set0(tr[top[v]],dep[u],dep[v]);
}
int main(){
  DFS(1,0,0);
  decom(1,0,true);
}
```

5.5 K-D Tree (Insert)

```
#ifndef SUNMOON_DYNEMIC_KD_TREE
#define SUNMOON_DYNEMIC_KD_TREE
#include<algorithm>
#include<vector>
#include<queue>
#include<cmath>
template<typename T,size_t kd>//kd表示有幾個維度
class kd_tree{
  public:
    struct point{
      T d[kd];
      inline T dist(const point &x)const{
        T ret=0;
        for(size_t i=0;i<kd;++i)ret+=std::abs(d[i]-x.d[i]);
        return ret;
      }
      inline bool operator<(const point &b)const{
        return d[0]<b.d[0];
      }
    };
  private:
    struct node{
      node *l,*r;
      point pid;
      int s;
      node(const point &p):l(0),r(0),pid(p),s(1){}
      inline void up(){
        s=(l?l->s:0)+1+(r?r->s:0);
      }
    }*root;
    const double alpha,loga;
    const T INF;//記得要給INF，表示極大值
    std::vector<node*> A;
    int qM;
    std::priority_queue<std::pair<T,point > >pQ;
    struct __cmp{
      int sort_id;
      inline bool operator()(const node*x,const node*y)const{
        return x->pid.d[sort_id]<y->pid.d[sort_id];
      }
    }cmp;
    void clear(node *o){
      if(!o)return;
      clear(o->l);
      clear(o->r);
      delete o;
    }
    inline int size(node *o){
      return o?o->s:0;
    }
    node* build(int k,int l,int r){
      if(l>r)return 0;
      if(k==kd)k=0;
      int mid=(l+r)/2;
      cmp.sort_id=k;
      std::nth_element(A.begin()+l,A.begin()+mid,A.begin()+r+1,cmp);
      node *ret=A[mid];
      ret->l=build(k+1,l,mid-1);
      ret->r=build(k+1,mid+1,r);
      ret->up();
      return ret;
    }
    inline bool isbad(node*o){
      return size(o->l)>alpha*o->s||size(o->r)>alpha*o->s;
    }
    void flatten(node *u,typename std::vector<node*>::iterator &it){
      if(!u)return;
      flatten(u->l,it);
      *it=u;
      flatten(u->r,++it);
    }
    bool insert(node*&u,int k,const point &x,int dep){
      if(!u){
        u=new node(x);
        return dep<=0;
      }
      ++u->s;
      if(insert(x.d[k]<u->pid.d[k]?u->l:u->r,(k+1)%kd,x,dep-1)){
        if(!isbad(u))return 1;
        if((int)A.size()<u->s)A.resize(u->s);
        typename std::vector<node*>::iterator it=A.begin();
        flatten(u,it);
        u=build(k,0,u->s-1);
      }
      return 0;
    }
    inline T heuristic(const T h[])const{
      T ret=0;
      for(size_t i=0;i<kd;++i)ret+=h[i];
      return ret;
    }
    void nearest(node *u,int k,const point &x,T *h,T &mndist){
      if(u==0||heuristic(h)>=mndist)return;
      T dist=u->pid.dist(x),old=h[k];
```

```
95        /*mndist=std::min(mndist,dist);*/
96        if(dist<mndist){
97          pQ.push(std::make_pair(dist,u->pid));
98          if((int)pQ.size()==qM+1){
99            mndist=pQ.top().first,pQ.pop();
100         }
101       }
102       if(x.d[k]<u->pid.d[k]){
103         nearest(u->l,(k+1)%kd,x,h,mndist);
104         h[k]=std::abs(x.d[k]-u->pid.d[k]);
105         nearest(u->r,(k+1)%kd,x,h,mndist);
106       }else{
107         nearest(u->r,(k+1)%kd,x,h,mndist);
108         h[k]=std::abs(x.d[k]-u->pid.d[k]);
109         nearest(u->l,(k+1)%kd,x,h,mndist);
110       }
111       h[k]=old;
112     }
113   public:
114     kd_tree(const T &INF,double a=0.75):root(0),alpha(a),loga(log2(1.0/a)),
      INF(INF){}
115     inline void clear(){
116       clear(root),root=0;
117     }
118     inline void build(int n,const point *p){
119       clear(root),A.resize(n);
120       for(int i=0;i<n;++i)A[i]=new node(p[i]);
121       root=build(0,0,n-1);
122     }
123     inline void insert(const point &x){
124       insert(root,0,x,std::__lg(size(root))/loga);
125     }
126     inline T nearest(const point &x,int k){
127       qM=k;
128       T mndist=INF,h[kd]={};
129       nearest(root,0,x,h,mndist);
130       mndist=pQ.top().first;
131       pQ=std::priority_queue<std::pair<T,point > >();
132       return mndist;/*回傳離x第k近的點的距離*/
133     }
134     inline int size(){return root?root->s:0;}
135 };
136 #endif
```

5.6 SCC

```
1  // Kosaraju - Find SCC by twice dfs, and the SCC DAG is in the Topology
2  // ordering.
3  // Owner: samsam2310
4  //
5  #include <bits/stdc++.h>
6  #define N 300002 // Maximum number of vertices
7  using namespace std;
8  vector<int> forward_graph[N];  // original graph
9  vector<int> backward_graph[N]; // reverse graph
10 vector<int> dag_graph[N];      // result dag graph(graph of scc)
11 int scc[N];                   // SCC index of a vertex
12 bool visit[N];
13 void init() {
14     fill(forward_graph, forward_graph + N, vector<int>());
15     fill(backward_graph, backward_graph + N, vector<int>());
16     fill(dag_graph, dag_graph + N, vector<int>());
17 }
18 void dfs(vector<int> &graph, int now, int scc_id,
19          stack<int> *leave_order = NULL) {
20     visit[now] = true;
21     if (scc != -1) {
22         scc[now] = scc_id;
23     }
24     for (int v : graph[now]) {
25         if (!visit[v]) {
26             dfs(graph, v, scc_id, leave_order);
27         }
28     }
29     if (leave_order) {
30         leave_order->push(now);
31     }
32 }
33 int main(int argc, char *argv[]) {
34     ios_base::sync_with_stdio(false);
35     cin.tie(0);
36     init();
37     cin >> n;
38     for (int i = 0; i < n; ++i) {
39         int a, b; // edge of a -> b
40         cin >> a >> b;
41         forward_graph[a].push_back(b);
42         backward_graph[b].push_back(a);
43     }
44     // Find the SCC.
45     memset(visit, 0, sizeof(visit));
46     stack<int> leave_order;
47     for (int i = 0; i < n; ++i) {
48         if (!visit[i]) {
49             dfs(forward_graph, i, -1, &leave_order);
50         }
51     }
52     memset(visit, 0, sizeof(visit));
53     int scc_id = 0;
54     while (!leave_order.empty()) {
55         int v = leave_order.top();
56         leave_order.pop();
57         if (!visit[v]) {
58             dfs(backward_graph, i, scc_id, NULL);
59             ++scc_id;
60         }
61     }
62     // Build the SCC DAG.
63     for (int i = 0; i < n; ++i) {
64         for (int v : forward_graph[i]) {
```

```
65              if (scc[i] != scc[v]) {
66                  dag_graph[scc[i]].push_back(scc[v]);
67              }
68          }
69      }
70      return 0;
71  }
```

5.7 Treap

```
1  struct Treap{
2    Treap *l,*r;
3    int pri,sz,val,add;
4    Treap(int _val):pri(rand()),sz(1),val(_val),add(0),l(NULL),r(NULL){}
5  };
6
7  int size(Treap *t){
8    return t?t->sz:0;
9  }
10 void pull(Treap *t){
11   t->sz=size(t->l)+size(t->r)+1;
12 }
13 void push(Treap *t){
14   t->val+=t->add;
15   if(t->l) t->l->add+=t->add;
16   if(t->r) t->r->add+=t->add;
17   t->add=0;
18 }
19 Treap* merge(Treap *a,Treap *b){
20   if(!a||!b) return a?a:b;
21   if(a->pri > b->pri){
22     push(a);
23     a->r = merge(a->r,b);
24     pull(a);
25     return a;
26   }
27   else{
28     push(b);
29     b->l = merge(a,b->l);
30     pull(b);
31     return b;
32   }
33 }
34 void split(Treap *t,int k,Treap *&a,Treap *&b){
35   if(!t) a=b=NULL;
36   else{
37     push(t);
38     if(size(t->l) < k){
39       a=t;
40       split(t->r,k-size(t->l)-1,a->r,b);
41       pull(a);
42     }
43     else{
44       b=t;
45       split(t->l,k,a,b->l);
46       pull(b);
47     }
48   }
49 }
```

6 Java

6.1 Big Integer

```
1  import java.math.*;
2  import java.io.*;
3  import java.util.*;
4  public class Main{
5      public static void main(String []argv){
6          c[0][0]=BigInteger.ONE;
7          for(int i=1;i<3001;i++){
8              c[i][0]=BigInteger.ONE;
9              c[i][i]=BigInteger.ONE;
10             for(int j=1;j<i;j++)c[i][j]=c[i-1][j].add(c[i-1][j-1]);
11         }
12         Scanner scanner = new Scanner(System.in);
13         int T = scanner.nextInt();
14         BigInteger x;
15         BigInteger ans;
16         while(T-- > 0){
17             ans = BigInteger.ZERO;
18             int n = scanner.nextInt();
19             for(int i=0;i<n;i++){
20                 x = new BigInteger(scanner.next());
21                 if(i%2 == 1)ans=ans.subtract(c[n-1][i].multiply(x));
22                 else ans=ans.add(c[n-1][i].multiply(x));
23             }
24             if(n%2 == 0)ans=BigInteger.ZERO.subtract(ans);
25             System.out.println(ans);
26         }
27     }
28 }
```

6.2 Prime

```
1  import java.math.*;
2  import java.io.*;
3  import java.util.*;
4  public class Main{
5      public static void main(String []argv){
6          Scanner scanner = new Scanner(System.in);
7          int T = scanner.nextInt();
8          for (int cs = 0 ; cs < T ; cs++){
```

```
            if (cs != 0) {
                System.out.println("");
            }
            int a = scanner.nextInt();
            int b = scanner.nextInt();
            for (int i = a ; i <= b ; i++) {
                BigInteger x = BigInteger.valueOf(i);
                if (x.isProbablePrime(5) == true) {
                    System.out.println(x);
                }
            }

        }
    }
}
```

7 Matching

7.1 Bipartite Matching

```
#include<bits/stdc++.h>
#define V 20100
#define inf 0x3f3f3f3f
int mx[V],my[V],dis[V],que[V];
bool vis[V];
vector<int> g[V];
bool DFS(int u){
  vis[u]=true;
  for(int i=0;i<g[u].size();i++){
    int v=my[g[u][i]];
    if(v==-1||!vis[v]&&dis[v]==dis[u]+1&&DFS(v)){
      mx[u]=g[u][i];
      my[g[u][i]]=u;
      return true;
    }
  }
  return false;
}
// n is the size of left hand side
int Hopcroft_Karp(int n){
  int matching=0,qt,qf,sp,i,u,v;
  bool flag=true;
  memset(mx,-1,sizeof(mx));
  memset(my,-1,sizeof(my));
  while(flag){
    flag=false;
    qt=qf=0;
    sp=inf;
    for(i=0;i<n;i++){
      if(mx[i]==-1){
        dis[i]=0;
        que[qt++]=i;
      }
      else dis[i]=inf;
    }
    while(qf<qt){
      u=que[qf++];
      if(dis[u]>=sp) continue;
      for(i=0;i<g[u].size();i++){
        v=my[g[u][i]];
        if(v==-1){
          if(dis[u]+1<sp){
            sp=dis[u]+1;
            flag=true;
          }
        }
        else if(dis[u]+1<dis[v]){
          dis[v]=dis[u]+1;
          que[qt++]=v;
        }
      }
    }
    if(flag){
      memset(vis,0,sizeof(vis));
      for(i=0;i<n;i++){
        if(dis[i]==0&&DFS(i)) matching++;
      }
    }
  }
  return matching;
}
```

7.2 Blossom

```
#define MAXN 505
vector<int>g[MAXN];//用vector存圖
int pa[MAXN],match[MAXN],st[MAXN],S[MAXN],vis[MAXN];
int t,n;
inline int lca(int u,int v){//找花的花托
  for(++t;;swap(u,v)){
    if(u==0)continue;
    if(vis[u]==t)return u;
    vis[u]=t;//這種方法可以不用清空vis陣列
    u=st[pa[match[u]]];
  }
}
#define qpush(u) q.push(u),S[u]=0
inline void flower(int u,int v,int l,queue<int> &q){
  while(st[u]!=l){
    pa[u]=v;//所有未匹配邊的pa都是雙向的
    if(S[v=match[u]]==1)qpush(v);//所有奇點變偶點
    st[u]=st[v]=l,u=pa[v];
  }
}
inline bool bfs(int u){
  for(int i=1;i<=n;++i)st[i]=i;//st[i]表示第i個點的集合
```

```
  memset(S+1,-1,sizeof(int)*n);//-1:沒走過 0:偶點 1:奇點
  queue<int>q;qpush(u);
  while(q.size()){
    u=q.front(),q.pop();
    for(size_t i=0;i<g[u].size();++i){
      int v=g[u][i];
      if(S[v]==-1){
        pa[v]=u,S[v]=1;
        if(!match[v]){//有增廣路直接擴充
          for(int lst;u;v=lst,u=pa[v])
            lst=match[u],match[u]=v,match[v]=u;
          return 1;
        }
        qpush(match[v]);
      }else if(!S[v]&&st[v]!=st[u]){
        int l=lca(st[v],st[u]);//遇到花，做花的處理
        flower(v,u,l,q),flower(u,v,l,q);
      }
    }
  }
  return 0;
}
inline int blossom(){
  memset(pa+1,0,sizeof(int)*n);
  memset(match+1,0,sizeof(int)*n);
  int ans=0;
  for(int i=1;i<=n;++i)
    if(!match[i]&&bfs(i))++ans;
  return ans;
}
```

7.3 Dinic

```
//Dinic
#define V 1000
struct edge{
    edge(){}
    edge(int a,int b,int c):to(a),cap(b),rev(c){}
    int to,cap,rev;
};
vector<edge> g[V];
int level[V];
int iter[V];
void add_edge(int from,int to,int cap){
    g[from].push_back(edge(to,cap,g[to].size()));
    g[to].push_back(edge(from,0,g[from].size()-1));
}
void bfs(int s){
    memset(level,-1,sizeof(level));
    queue<int>que;
    level[s]=0;
    que.push(s);
    while(!que.empty()){
        int v=que.front();
        que.pop();
        for(int q=0;q<g[v].size();q++){
            edge &e=g[v][q];
            if(e.cap>0&&level[e.to]<0){
                level[e.to]=level[v]+1;
                que.push(e.to);
            }
        }
    }
}
int dfs(int v,int t,int f){
    if(v==t)return f;
    for(int &q=iter[v];q<g[v].size();++q){
        edge &e=g[v][q];
        if(e.cap>0&&level[v]<level[e.to]){
            int d=dfs(e.to,t,min(f,e.cap));
            if(d>0){
                e.cap-=d;
                g[e.to][e.rev].cap+=d;
                return d;
            }
        }
    }
    return 0;
}
int max_flow(int s,int t){
    int flow=0;
    for(;;){
        bfs(s);
        if(level[t]<0)return flow;
        memset(iter,0,sizeof(iter));
        int f;
        while((f=dfs(s,t,1e9))>0)
              flow+=f;
    }
}
```

7.4 General Weighted Matching

```
#include <iostream>
#include <cstdio>
#include <algorithm>
#include <vector>
using namespace std;

typedef long long s64;

const int INF = 2147483647;

const int MaxN = 400;
const int MaxM = 79800;

template <class T>
inline void tension(T &a, const T &b)
```

```
{
  if (b < a)
    a = b;
}
template <class T>
inline void relax(T &a, const T &b)
{
  if (b > a)
    a = b;
}
template <class T>
inline int size(const T &a)
{
  return (int)a.size();
}

inline int getint()
{
  char c;
  while (c = getchar(), '0' > c || c > '9');

  int res = c - '0';
  while (c = getchar(), '0' <= c && c <= '9')
    res = res * 10 + c - '0';
  return res;
}

const int MaxNX = MaxN + MaxN;

struct edge
{
  int v, u, w;

  edge(){}
  edge(const int &_v, const int &_u, const int &_w)
    : v(_v), u(_u), w(_w){}
};

int n, m;
edge mat[MaxNX + 1][MaxNX + 1];

int n_matches;
s64 tot_weight;
int mate[MaxNX + 1];
int lab[MaxNX + 1];

int q_n, q[MaxN];
int fa[MaxNX + 1], col[MaxNX + 1];
int slackv[MaxNX + 1];

int n_x;
int bel[MaxNX + 1], blofrom[MaxNX + 1][MaxN + 1];
vector<int> bloch[MaxNX + 1];

inline int e_delta(const edge &e) // does not work inside blossoms
{
  return lab[e.v] + lab[e.u] - mat[e.v][e.u].w * 2;
}
inline void update_slackv(int v, int x)
{
  if (!slackv[x] || e_delta(mat[v][x]) < e_delta(mat[slackv[x]][x]))
    slackv[x] = v;
}
inline void calc_slackv(int x)
{
  slackv[x] = 0;
  for (int v = 1; v <= n; v++)
    if (mat[v][x].w > 0 && bel[v] != x && col[bel[v]] == 0)
      update_slackv(v, x);
}

inline void q_push(int x)
{
  if (x <= n)
    q[q_n++] = x;
  else
  {
    for (int i = 0; i < size(bloch[x]); i++)
      q_push(bloch[x][i]);
  }
}
inline void set_mate(int xv, int xu)
{
  mate[xv] = mat[xv][xu].u;
  if (xv > n)
  {
    edge e = mat[xv][xu];
    int xr = blofrom[xv][e.v];
    int pr = find(bloch[xv].begin(), bloch[xv].end(), xr) - bloch[xv].begin()
    ;
    if (pr % 2 == 1)
    {
      reverse(bloch[xv].begin() + 1, bloch[xv].end());
      pr = size(bloch[xv]) - pr;
    }

    for (int i = 0; i < pr; i++)
      set_mate(bloch[xv][i], bloch[xv][i ^ 1]);
    set_mate(xr, xu);

    rotate(bloch[xv].begin(), bloch[xv].begin() + pr, bloch[xv].end());
  }
}
inline void set_bel(int x, int b)
{
  bel[x] = b;
  if (x > n)
  {
    for (int i = 0; i < size(bloch[x]); i++)
      set_bel(bloch[x][i], b);
  }
}
```

```
127
128 inline void augment(int xv, int xu)
129 {
130   while (true)
131   {
132     int xnu = bel[mate[xv]];
133     set_mate(xv, xu);
134     if (!xnu)
135       return;
136     set_mate(xnu, bel[fa[xnu]]);
137     xv = bel[fa[xnu]], xu = xnu;
138   }
139 }
140 inline int get_lca(int xv, int xu)
141 {
142   static bool book[MaxNX + 1];
143   for (int x = 1; x <= n_x; x++)
144     book[x] = false;
145   while (xv || xu)
146   {
147     if (xv)
148     {
149       if (book[xv])
150         return xv;
151       book[xv] = true;
152       xv = bel[mate[xv]];
153       if (xv)
154         xv = bel[fa[xv]];
155     }
156     swap(xv, xu);
157   }
158   return 0;
159 }
160
161 inline void add_blossom(int xv, int xa, int xu)
162 {
163   int b = n + 1;
164   while (b <= n_x && bel[b])
165     b++;
166   if (b > n_x)
167     n_x++;
168
169   lab[b] = 0;
170   col[b] = 0;
171
172   mate[b] = mate[xa];
173
174   bloch[b].clear();
175   bloch[b].push_back(xa);
176   for (int x = xv; x != xa; x = bel[fa[bel[mate[x]]]])
177     bloch[b].push_back(x), bloch[b].push_back(bel[mate[x]]), q_push(bel[mate[
      x]]);
178   reverse(bloch[b].begin() + 1, bloch[b].end());
179   for (int x = xu; x != xa; x = bel[fa[bel[mate[x]]]])
180     bloch[b].push_back(x), bloch[b].push_back(bel[mate[x]]), q_push(bel[mate[
      x]]);
181
182   set_bel(b, b);
183
184   for (int x = 1; x <= n_x; x++)
185   {
186     mat[b][x].w = mat[x][b].w = 0;
187     blofrom[b][x] = 0;
188   }
189   for (int i = 0; i < size(bloch[b]); i++)
190   {
191     int xs = bloch[b][i];
192     for (int x = 1; x <= n_x; x++)
193       if (mat[b][x].w == 0 || e_delta(mat[xs][x]) < e_delta(mat[b][x]))
194         mat[b][x] = mat[xs][x], mat[x][b] = mat[x][xs];
195     for (int x = 1; x <= n_x; x++)
196       if (blofrom[xs][x])
197         blofrom[b][x] = xs;
198   }
199   calc_slackv(b);
200 }
201 inline void expand_blossom1(int b) // lab[b] == 1
202 {
203   for (int i = 0; i < size(bloch[b]); i++)
204     set_bel(bloch[b][i], bloch[b][i]);
205
206   int xr = blofrom[b][mat[b][fa[b]].v];
207   int pr = find(bloch[b].begin(), bloch[b].end(), xr) - bloch[b].begin();
208   if (pr % 2 == 1)
209   {
210     reverse(bloch[b].begin() + 1, bloch[b].end());
211     pr = size(bloch[b]) - pr;
212   }
213
214   for (int i = 0; i < pr; i += 2)
215   {
216     int xs = bloch[b][i], xns = bloch[b][i + 1];
217     fa[xs] = mat[xns][xs].v;
218     col[xs] = 1, col[xns] = 0;
219     slackv[xs] = 0, calc_slackv(xns);
220     q_push(xns);
221   }
222   col[xr] = 1;
223   fa[xr] = fa[b];
224   for (int i = pr + 1; i < size(bloch[b]); i++)
225   {
226     int xs = bloch[b][i];
227     col[xs] = -1;
228     calc_slackv(xs);
229   }
230
231   bel[b] = 0;
232 }
233 inline void expand_blossom_final(int b) // at the final stage
234 {
235   for (int i = 0; i < size(bloch[b]); i++)
236   {
```

```
    if (bloch[b][i] > n && lab[bloch[b][i]] == 0)
      expand_blossom_final(bloch[b][i]);
    else
      set_bel(bloch[b][i], bloch[b][i]);
  }
  bel[b] = 0;
}

inline bool on_found_edge(const edge &e)
{
  int xv = bel[e.v], xu = bel[e.u];
  if (col[xu] == -1)
  {
    int nv = bel[mate[xu]];
    fa[xu] = e.v;
    col[xu] = 1, col[nv] = 0;
    slackv[xu] = slackv[nv] = 0;
    q_push(nv);
  }
  else if (col[xu] == 0)
  {
    int xa = get_lca(xv, xu);
    if (!xa)
    {
      augment(xv, xu), augment(xu, xv);
      for (int b = n + 1; b <= n_x; b++)
        if (bel[b] == b && lab[b] == 0)
          expand_blossom_final(b);
      return true;
    }
    else
      add_blossom(xv, xa, xu);
  }
  return false;
}

bool match()
{
  for (int x = 1; x <= n_x; x++)
    col[x] = -1, slackv[x] = 0;

  q_n = 0;
  for (int x = 1; x <= n_x; x++)
    if (bel[x] == x && !mate[x])
      fa[x] = 0, col[x] = 0, slackv[x] = 0, q_push(x);
  if (q_n == 0)
    return false;

  while (true)
  {
    for (int i = 0; i < q_n; i++)
    {
      int v = q[i];
      for (int u = 1; u <= n; u++)
        if (mat[v][u].w > 0 && bel[v] != bel[u])
        {
          int d = e_delta(mat[v][u]);
          if (d == 0)
          {
            if (on_found_edge(mat[v][u]))
              return true;
          }
          else if (col[bel[u]] == -1 || col[bel[u]] == 0)
            update_slackv(v, bel[u]);
        }
    }

    int d = INF;
    for (int v = 1; v <= n; v++)
      if (col[bel[v]] == 0)
        tension(d, lab[v]);
    for (int b = n + 1; b <= n_x; b++)
      if (bel[b] == b && col[b] == 1)
        tension(d, lab[b] / 2);
    for (int x = 1; x <= n_x; x++)
      if (bel[x] == x && slackv[x])
      {
        if (col[x] == -1)
          tension(d, e_delta(mat[slackv[x]][x]));
        else if (col[x] == 0)
          tension(d, e_delta(mat[slackv[x]][x]) / 2);
      }

    for (int v = 1; v <= n; v++)
    {
      if (col[bel[v]] == 0)
        lab[v] -= d;
      else if (col[bel[v]] == 1)
        lab[v] += d;
    }
    for (int b = n + 1; b <= n_x; b++)
      if (bel[b] == b)
      {
        if (col[bel[b]] == 0)
          lab[b] += d * 2;
        else if (col[bel[b]] == 1)
          lab[b] -= d * 2;
      }

    q_n = 0;
    for (int v = 1; v <= n; v++)
      if (lab[v] == 0) // all unmatched vertices' labels are zero! cheers!
        return false;
    for (int x = 1; x <= n_x; x++)
      if (bel[x] == x && slackv[x] && bel[slackv[x]] != x && e_delta(mat[
    slackv[x]][x]) == 0)
      {
        if (on_found_edge(mat[slackv[x]][x]))
          return true;
      }
    for (int b = n + 1; b <= n_x; b++)
      if (bel[b] == b && col[b] == 1 && lab[b] == 0)
```

```
348         expand_blossom1(b);
349   }
350   return false;
351 }
352
353 void calc_max_weight_match()
354 {
355   for (int v = 1; v <= n; v++)
356     mate[v] = 0;
357
358   n_x = n;
359   n_matches = 0;
360   tot_weight = 0;
361
362   bel[0] = 0;
363   for (int v = 1; v <= n; v++)
364     bel[v] = v, bloch[v].clear();
365   for (int v = 1; v <= n; v++)
366     for (int u = 1; u <= n; u++)
367       blofrom[v][u] = v == u ? v : 0;
368
369   int w_max = 0;
370   for (int v = 1; v <= n; v++)
371     for (int u = 1; u <= n; u++)
372       relax(w_max, mat[v][u].w);
373   for (int v = 1; v <= n; v++)
374     lab[v] = w_max;
375
376   while (match())
377     n_matches++;
378
379   for (int v = 1; v <= n; v++)
380     if (mate[v] && mate[v] < v)
381       tot_weight += mat[v][mate[v]].w;
382 }
383
384 int main()
385 {
386   n = getint(), m = getint();
387
388   for (int v = 1; v <= n; v++)
389     for (int u = 1; u <= n; u++)
390       mat[v][u] = edge(v, u, 0);
391
392   for (int i = 0; i < m; i++)
393   {
394     int v = getint(), u = getint(), w = getint();
395     mat[v][u].w = mat[u][v].w = w;
396   }
397
398   calc_max_weight_match();
399
400   printf("%lld\n", tot_weight);
401   for (int v = 1; v <= n; v++)
402     printf("%d ", mate[v]);
403   printf("\n");
404
405   return 0;
406 }
```

7.5 KM

```
1  #define MAXN 100
2  #define INF INT_MAX
3  int g[MAXN][MAXN],lx[MAXN],ly[MAXN],slack_y[MAXN];
4  int px[MAXN],py[MAXN],match_y[MAXN],par[MAXN];
5  int n;
6  void adjust(int y){//把增廣路上所有邊反轉
7    match_y[y]=py[y];
8    if(px[match_y[y]]!=-2)
9      adjust(px[match_y[y]]);
10 }
11 bool dfs(int x){//DFS找增廣路
12   for(int y=0;y<n;++y){
13     if(py[y]!=-1)continue;
14     int t=lx[x]+ly[y]-g[x][y];
15     if(t==0){
16       py[y]=x;
17       if(match_y[y]==-1){
18         adjust(y);
19         return 1;
20       }
21       if(px[match_y[y]]!=-1)continue;
22       px[match_y[y]]=y;
23       if(dfs(match_y[y]))return 1;
24     }else if(slack_y[y]>t){
25       slack_y[y]=t;
26       par[y]=x;
27     }
28   }
29   return 0;
30 }
31 inline int km(){
32   memset(ly,0,sizeof(int)*n);
33   memset(match_y,-1,sizeof(int)*n);
34   for(int x=0;x<n;++x){
35     lx[x]=-INF;
36     for(int y=0;y<n;++y){
37       lx[x]=max(lx[x],g[x][y]);
38     }
39   }
40   for(int x=0;x<n;++x){
41     for(int y=0;y<n;++y)slack_y[y]=INF;
42     memset(px,-1,sizeof(int)*n);
43     memset(py,-1,sizeof(int)*n);
44     px[x]=-2;
45     if(dfs(x))continue;
46     bool flag=1;
47     while(flag){
48       int cut=INF;
```

```
      for(int y=0;y<n;++y)
        if(py[y]==-1&&cut>slack_y[y])cut=slack_y[y];
      for(int j=0;j<n;++j){
        if(px[j]!=-1)lx[j]-=cut;
        if(py[j]!=-1)ly[j]+=cut;
        else slack_y[j]-=cut;
      }
      for(int y=0;y<n;++y){
        if(py[y]==-1&&slack_y[y]==0){
          py[y]=par[y];
          if(match_y[y]==-1){
            adjust(y);
            flag=0;
            break;
          }
          px[match_y[y]]=y;
          if(dfs(match_y[y])){
            flag=0;
            break;
          }
        }
      }
    }
  }
  int ans=0;
  for(int y=0;y<n;++y)if(g[match_y[y]][y]!=-INF)ans+=g[match_y[y]][y];
  return ans;
}
```

7.6 Min Cost Flow

```
#define maxnode (1000+10)
#define maxedge (40000+10)
#define INF 1023456789
#include<bits/stdc++.h>
using namespace std;
int node, src, dest, nedge;
int head[maxnode], point[maxedge], nxt[maxedge], flow[maxedge], capa[maxedge], wt[maxedge];
int dist[maxnode], in[maxnode], from[maxnode], mf[maxnode];
//set number of node, source, and destination (one base)
void init(int _node, int _src, int _dest) {
    node = _node;
    src = _src;
    dest = _dest;
    nedge = 0;
    memset(point, -1, sizeof(point));
    for (int i = 1; i <= node; i++) head[i] = -1;
    nedge = 0;
}
void add_edge(int u, int v, int c1, int w) {
    point[nedge] = v, capa[nedge] = c1, flow[nedge] = 0, nxt[nedge] = head[u], wt[nedge]=w, head[u] = (nedge++);
    point[nedge] = u, capa[nedge] = 0, flow[nedge] = 0, nxt[nedge] = head[v], wt[nedge]=-w, head[v] = (nedge++);
}
int sp(int &left){
  for(int i=1;i<=node;i++) dist[i]=INF;
  queue<int> que;
  que.push(src);
  in[src]=1;
  mf[src]=left;
  dist[src]=0;
  while(!que.empty()){
    int u=que.front();
    que.pop();
    in[u]=0;
    if(dist[u]>=dist[dest]) continue;
    for(int v=head[u];v!=-1;v=nxt[v]){
      if(flow[v]==capa[v]) continue;
      if(dist[u]+wt[v]<dist[point[v]]){
        dist[point[v]]=dist[u]+wt[v];
        from[point[v]]=v;
        mf[point[v]]=min(mf[u],capa[v]-flow[v]);
        if(!in[point[v]]){
          in[point[v]]=1;
          que.push(point[v]);
        }
      }
    }
  }
  left-=mf[dest];
  if(dist[dest]<INF){
    for(int u=dest;u!=src;u=point[from[u]^1]){
      flow[from[u]]+=mf[dest];
      flow[from[u]^1]-=mf[dest];
    }
  }
  return dist[dest];
}
int min_cost_flow(){
  int res=0,tmp,maxflow=2;
  while(maxflow&&(tmp=sp(maxflow))<INF) res+=tmp;
  return res;
}
int main(){
  int n,m,x,y,z;
  while(scanf("%d%d",&n,&m)==2){
    init(n,1,n);
    for(int i=0;i<m;i++){
      scanf("%d%d%d",&x,&y,&z);
      add_edge(x,y,1,z);
      add_edge(y,x,1,z); //undirected
    }
    printf("%d\n",min_cost_flow());
  }
  return 0;
}
```

7.7 Stable Marriage

```
#define F(n) Fi(i, n)
#define Fi(i, n) Fl(i, 0, n)
#define Fl(i, l, n) for(int i = l ; i < n ; ++i)
#include <bits/stdc++.h>
using namespace std;
int D, quota[205], weight[205][5];
int S, scoretodep[12005][205], score[5];
int P, prefer[12005][85], iter[12005];
int ans[12005];
typedef pair<int, int> PII;
map<int, int> samescore[205];
typedef priority_queue<PII, vector<PII>, greater<PII>> QQQ;
QQQ pri[205];
void check(int d) {
  PII t = pri[d].top();
  int v;
  if (pri[d].size() - samescore[d][t.first] + 1 <= quota[d]) return;
  while (pri[d].top().first == t.first) {
    v = pri[d].top().second;
    ans[v] = -1;
    --samescore[d][t.first];
    pri[d].pop();
  }
}
void push(int s, int d) {
  if (pri[d].size() < quota[d]) {
    pri[d].push(PII(scoretodep[s][d], s));
    ans[s] = d;
    ++samescore[s][scoretodep[s][d]];
  } else if (scoretodep[s][d] >= pri[d].top().first) {
    pri[d].push(PII(scoretodep[s][d], s));
    ans[s] = d;
    ++samescore[s][scoretodep[s][d]];
    check(d);
  }
}
void f() {
  int over;
  while (true) {
    over = 1;
    Fi (q, S) {
      if (ans[q] != -1 || iter[q] >= P) continue;
      push(q, prefer[q][iter[q]++]);
      over = 0;
    }
    if (over) break;
  }
}
main() {
  ios::sync_with_stdio(false);
  cin.tie(NULL);
  int sadmit, stof, dexceed, dfew;
  while (cin >> D, D) { // Beware of the input format or judge may troll us.
    sadmit = stof = dexceed = dfew = 0;
    memset(iter, 0, sizeof(iter));
    memset(ans, 0, sizeof(ans));
    Fi (q, 205) {
      pri[q] = QQQ();
      samescore[q].clear();
    }
    cin >> S >> P;
    Fi (q, D) {
      cin >> quota[q];
      Fi (w, 5) cin >> weight[q][w];
    }
    Fi (q, S) {
      Fi (w, 5) cin >> score[w];
      Fi (w, D) {
        scoretodep[q][w] = 0;
        F (5) scoretodep[q][w] += weight[w][i] * score[i];
      }
    }
    Fi (q, S) Fi (w, P) {
      cin >> prefer[q][w];
      --prefer[q][w];
    }
    f();
    Fi (q, D) sadmit += pri[q].size();
    Fi (q, S) if (ans[q] == prefer[q][0]) ++stof;
    Fi (q, D) if (pri[q].size() > quota[q]) ++dexceed;
    Fi (q, D) if (pri[q].size() < quota[q]) ++dfew;
    cout << sadmit << ' ' << stof << ' ' << dexceed << ' ' << dfew << '\n';
  }
}
```

8 Mathematics

8.1 Extended GCD

```
long long extgcd(long long a,long long b,long long &x,long long &y){
    long long d=a;
    if(b!=0){
        d=extgcd(b,a%b,y,x);
        y-=(a/b)*x;
    }
    else x=1,y=0;
    return d;
}
int main(){
    int T;
    long long a,b,m,GCD,x,y;
    while(~scanf("%d",&T))
        while(T--){
            scanf("%lld%lld%lld",&m,&a,&b);
            GCD=extgcd(a,m,x,y);
            if(GCD!=1)printf("No inverse, gcd(a,m)=%lld\n",GCD);
```

```
            else{
                b=((-b*x)%m+m)%m;
                printf("%lld %lld\n",(x%m+m)%m,b);
            }
        }
}
```

8.2 Lucas's Theorem

```
bigM = int(1e9+7)
fac = [1]*10001
for i in range(1, 10001):
  fac[i] = fac[i-1]*i
ifac = [pow(fac[i], bigM-2, bigM) for i in range(10001)]
def f(a, b, M):
  if b == 0 or b == a:
    return 1
  elif a < b:
    return 0
  elif a < M:
    return fac[a]*ifac[b]*ifac[a-b]%bigM
  else:
    return f(a//M, b//M, M) * f(a%M, b%M, M) % bigM
t = int(input())
for cases in range(t):
  a, b, M = [int(x) for x in input().split()]
  print(f(a, b, M))
```

8.3 Miller-Rabin

```
inline long long mod_mul(long long a,long long b,long long m){
  a%=m,b%=m;
  long long y=(long long)((double)a*b/m+0.5);/* fast for m < 2^58 */
  long long r=(a*b-y*m)%m;
  return r<0?r+m:r;
}
template<typename T>
inline T pow(T a,T b,T mod){//a^b%mod
  T ans=1;
  for(;b;a=mod_mul(a,a,mod),b>>=1)
    if(b&1)ans=mod_mul(ans,a,mod);
  return ans;
}
int sprp[3]={2,7,61};//int範圍可解
int llsprp[7]={2,325,9375,28178,450775,9780504,1795265022};//至少unsigned
    long long範圍
template<typename T>
inline bool isprime(T n,int *sprp,int num){
  if(n==2)return 1;
  if(n<2||n%2==0)return 0;
  int t=0;
  T u=n-1;
  for(;u%2==0;++t)u>>=1;
  for(int i=0;i<num;++i){
    T a=sprp[i]%n;
    if(a==0||a==1||a==n-1)continue;
    T x=pow(a,u,n);
    if(x==1||x==n-1)continue;
    for(int j=0;j<t;++j){
      x=mod_mul(x,x,n);
      if(x==1)return 0;
      if(x==n-1)break;
    }
    if(x==n-1)continue;
    return 0;
  }
  return 1;
}
```

8.4 Tonelli Shanks

```
#include<cstdio>
#include<cassert>
#include<cstdlib>
using namespace std;
int pow_mod(int a,int p,int q){ //a^p mod q
  int r=1;
  while(p){
    if(p&1) r=1LL*r*a%q;
    a=1LL*a*a%q;
    p>>=1;
  }
  return r;
}
int Jacobi(int q,int p){ //q/p
  if(p==1) return 1;
  q%=p;
  int c2=0,m2;
  while(!(q&1)){
    q>>=1;
    c2^=1;
  }
  if((p&7)==7||(p&7)==1||!c2) m2=1;
  else m2=-1;
  if((p&2)&&(q&2)) m2*=-1;
  return m2*Jacobi(p,q);
}
int Tonelli_Shanks(int a,int p){ //p is prime,gcd(a,p)=1
  if(p==2) return 1;
  if(Jacobi(a,p)==-1) return -1;
  int s=0,q=p-1,z=2;
  while(!(q&1)) q>>=1,s++;
  while(Jacobi(z,p)==1) z++;
  z = pow_mod(z, q, p);
  int zp[30]={z};
```

```
  for(int i=1;i<s;i++) zp[i]=1LL*zp[i-1]*zp[i-1]%p;
  int r = pow_mod(a,(q+1)>>1, p), t = pow_mod(a, q, p);
  while(t!=1){
    int m=0;
    for(int i=t;i!=1;i=1LL*i*i%p) m++;
    r=1LL*r*zp[s-m-1]%p;
    t=1LL*t*zp[s-m]%p;
  }
  return r;
}
int main(){
  for(int i=0;i<37;i++){

  }
  return 0;
}
```

9 String

9.1 AC Automaton

```
#ifndef SUNMOON_AHO_CORASICK_AUTOMATON
#define SUNMOON_AHO_CORASICK_AUTOMATON
#include<queue>
#include<vector>
template<char L='a',char R='z'>
class ac_automaton{
  private:
    struct joe{
      int next[R-L+1],fail,efl,ed,cnt_dp,vis;
      joe():ed(0),cnt_dp(0),vis(0){
        for(int i=0;i<=R-L;++i)next[i]=0;
      }
    };
  public:
    std::vector<joe> S;
    std::vector<int> q;
    int qs,qe,vt;
    ac_automaton():S(1),qs(0),qe(0),vt(0){}
    inline void clear(){
      q.clear();
      S.resize(1);
      for(int i=0;i<=R-L;++i)S[0].next[i]=0;
      S[0].cnt_dp=S[0].vis=qs=qe=vt=0;
    }
    inline void insert(const char *s){
      int o=0;
      for(int i=0,id;s[i];++i){
        id=s[i]-L;
        if(!S[o].next[id]){
          S.push_back(joe());
          S[o].next[id]=S.size()-1;
        }
        o=S[o].next[id];
      }
      ++S[o].ed;
    }
    inline void build_fail(){
      S[0].fail=S[0].efl=-1;
      q.clear();
      q.push_back(0);
      ++qe;
      while(qs!=qe){
        int pa=q[qs++],id,t;
        for(int i=0;i<=R-L;++i){
          t=S[pa].next[i];
          if(!t)continue;
          id=S[pa].fail;
          while(~id&&!S[id].next[i])id=S[id].fail;
          S[t].fail=~id?S[id].next[i]:0;
          S[t].efl=S[S[t].fail].ed?S[t].fail:S[S[t].fail].efl;
          q.push_back(t);
          ++qe;
        }
      }
    }
    /*DP出每個前綴在字串s出現的次數並傳回所有字串被s匹配成功的次數O(N+M)*/
    inline int match_0(const char *s){
      int ans=0,id,p=0,i;
      for(i=0;s[i];++i){
        id=s[i]-L;
        while(!S[p].next[id]&&p)p=S[p].fail;
        if(!S[p].next[id])continue;
        p=S[p].next[id];
        ++S[p].cnt_dp;/*匹配成功則它所有後綴都可以被匹配(DP計算)*/
      }
      for(i=qe-1;i>=0;--i){
        ans+=S[q[i]].cnt_dp*S[q[i]].ed;
        if(~S[q[i]].fail)S[S[q[i]].fail].cnt_dp+=S[q[i]].cnt_dp;
      }
      return ans;
    }
    /*多串匹配走efl邊並傳回所有字串被s匹配成功的次數O(N*M^1.5)*/
    inline int match_1(const char *s)const{
      int ans=0,id,p=0,t;
      for(int i=0;s[i];++i){
        id=s[i]-L;
        while(!S[p].next[id]&&p)p=S[p].fail;
        if(!S[p].next[id])continue;
        p=S[p].next[id];
        if(S[p].ed)ans+=S[p].ed;
        for(t=S[p].efl;~t;t=S[t].efl){
          ans+=S[t].ed;/*因為都走efl邊所以保證匹配成功*/
        }
      }
      return ans;
    }
    /*枚舉(s的子字串∩A)的所有相異字串各恰一次並傳回次數O(N*M^(1/3))*/
```

```
    inline int match_2(const char *s){
      int ans=0,id,p=0,t;
      ++vt;
      /*把戳記vt+=1，只要vt沒溢位，所有S[p].vis==vt就會變成false
      這種利用vt的方法可以O(1)歸零vis陣列*/
      for(int i=0;s[i];++i){
        id=s[i]-L;
        while(!S[p].next[id]&&p)p=S[p].fail;
        if(!S[p].next[id])continue;
        p=S[p].next[id];
        if(S[p].ed&&S[p].vis!=vt){
          S[p].vis=vt;
          ans+=S[p].ed;
        }
        for(t=S[p].efl;~t&&S[t].vis!=vt;t=S[t].efl){
          S[t].vis=vt;
          ans+=S[t].ed;/*因為都走efl邊所以保證匹配成功*/
        }
      }
      return ans;
    }
    /*把AC自動機變成真的自動機*/
    inline void evolution(){
      for(qs=1;qs!=qe;){
        int p=q[qs++];
        for(int i=0;i<=R-L;++i)
          if(S[p].next[i]==0)S[p].next[i]=S[S[p].fail].next[i];
      }
    }
};
#endif
```

9.2 BWT

```
// BWT
const int N = 8;              // 字串長度
int s[N+N+1] = "suffixes";    // 字串，後面預留一倍空間。
int sa[N];                    // 後綴陣列
int pivot;

int cmp(const void* i, const void* j)
{
    return strncmp(s+*(int*)i, s+*(int*)j, N);
}

// 此處便宜行事，採用 O(N²logN) 的後綴陣列演算法。
void BWT()
{
    strncpy(s + N, s, N);
    for (int i=0; i<N; ++i) sa[i] = i;
    qsort(sa, N, sizeof(int), cmp);
    // 當輸入字串的所有字元都相同，必須當作特例處理。
    // 或者改用stable sort。

    for (int i=0; i<N; ++i)
        cout << s[(sa[i] + N-1) % N];

    for (int i=0; i<N; ++i)
        if (sa[i] == 0)
        {
            pivot = i;
            break;
        }
}

// Inverse BWT
const int N = 8;              // 字串長度
char t[N+1] = "xuffessi";     // 字串
int pivot;
int next[N];

void IBWT()
{
    vector<int> index[256];
    for (int i=0; i<N; ++i)
        index[t[i]].push_back(i);

    for (int i=0, n=0; i<256; ++i)
        for (int j=0; j<index[i].size(); ++j)
            next[n++] = index[i][j];

    int p = pivot;
    for (int i=0; i<N; ++i)
        cout << t[p = next[p]];
}
```

9.3 Suffix Array

```
//should initialize s and n first
#define N 301000
using namespace std;
char s[N]; //string=s,suffix array=sar,longest common prefix=lcp
int rk[2][N],id[2][N];
int n,p;
int cnt[N];
int len[N],od[N],sar[N];
inline int sr(int i,int t){ //rank of shifted position
  return i+t<n?rk[p][i+t]:-1;
}
inline bool check_same(int i,int j,int t){
  return rk[p][i]==rk[p][j]&&sr(i,t)==sr(j,t);
}
bool cmp(int i,int j){
  return s[i]<s[j];
}
void sa(){ //length of array s
  int i,t,now,pre;
```

```
  memset(cnt,0,sizeof(cnt));
  for(i=0;i<n;i++){
    id[p][i]=i;
    rk[p][i]=s[i];
    cnt[s[i]]++;
  }
  for(i=1;i<128;i++) cnt[i]+=cnt[i-1];
  sort(id[p],id[p]+n,cmp);
  for(t=1;t<n;t<<=1){
        //least significant bit is already sorted
    for(i=n-1;i>=0;i--){
            now=id[p][i]-t;
      if(now>=0) id[p^1][--cnt[rk[p][now]]]=now;
    }
    for(i=n-t;i<n;i++){
            id[p^1][--cnt[rk[p][i]]]=i;
    }
    memset(cnt,0,sizeof(cnt));
    now=id[p^1][0];
    rk[p^1][now]=0;
    cnt[0]++;
    for(i=1;i<n;i++){
      pre=now;
      now=id[p^1][i];
      if(check_same(pre,now,t)){
        rk[p^1][now]=rk[p^1][pre];
      }
      else{
        rk[p^1][now]=rk[p^1][pre]+1;
      }
      cnt[rk[p^1][now]]++;
    }
    p^=1;
    if(rk[p][now]==n-1) break;
    for(i=1;i<n;i++) cnt[i]+=cnt[i-1];
  }
  memcpy(sar,id[p],sizeof(sar));
}
void lcp(){
    int i,l,pre;
    for(i=0;i<n;i++) od[sar[i]]=i;
    for(i=0;i<n;i++){
        if(i) l=len[od[i-1]]?len[od[i-1]]-1:0;
        else l=0;
        if(od[i]){
            pre=sar[od[i]-1];
            while(pre+l<n&&i+l<n&&s[pre+l]==s[i+l]) l++;
            len[od[i]]=l;
        }
        else len[0]=0;
    }
}
```

9.4 Suffix Automaton

```
#include<bits/stdc++.h>
#define C 96
#define N 200100
using namespace std;
struct SAM{
  struct node{
    node *nxt[C],*pre;
    int len;
    vector<int> pos;
  };
  node mem[N*2],*root,*ed;
  int top;
  SAM(){
    top = 0;
    root = new_node(0);
    ed = root;
  }
  node *new_node(int l){
    for(int i=0;i<C;i++) mem[top].nxt[i]=NULL;
    mem[top].pre=NULL;
    mem[top].len=l;
    mem[top].pos.clear();
    return mem+(top++);
  }
  node *split_node(int l,node *p){
    for(int i=0;i<C;i++) mem[top].nxt[i]=p->nxt[i];
    mem[top].pre = p->pre;
    mem[top].len = l;
    mem[top].pos.assign()
    p->pre = mem+top;
    return mem+(top++);
  }
  void push(char c){
    node *nw = new_node(ed->len+1),*ptr=ed->pre;
    ed->nxt[c] = nw;
    nw->pos.push_back(ed->len);
    for(;ptr;ptr=ptr->pre){
      if(ptr->nxt[c]){
        if(ptr->nxt[c]->len==ptr->len+1){
          nw->pre = ptr->nxt[c];
        }
        else{
          node *tmp=ptr->nxt[c];
          nw->pre = split_node(ptr->len+1,tmp);
          while(ptr && ptr->nxt[c]==tmp){
            ptr->nxt[c] = nw->pre;
            ptr = ptr->pre;
          }
        }
        break;
      }
      else{
        ptr->nxt[c] = nw;
      }
```

```
55      }
56      if(!nw->pre) nw->pre = root;
57      ed = ed->nxt[c];
58    }
59    void init(){
60      while(top){
61        mem[--top].pos.clear();
62      }
63      root = new_node(0);
64      ed = root;
65    }
66    void push(char *s){
67      for(int i=0;s[i];i++) push(s[i]-32);
68    }
69    long long count(){
70      long long ans=0;
71      for(int i=1;i<top;i++){
72        ans+=mem[i].len-mem[i].pre->len;
73      }
74      return ans;
75    }
76  }sam;
77  char S[N];
78  int main(){
79    int T;
80    scanf("%d",&T);
81    while(T--){
82      scanf("%s",S);
83      sam.build(S);
84      printf("%lld\n",sam.count());
85    }
86    return 0;
87  }
```

9.5 Z Algorithm

```
1  void Zalg(char *s, int *z, int n) {
2    z[0]=n;
3    for(int L=0, R=0, i=1; i<n; i++) {
4      if(i<=R && z[i-L]<=R-i) z[i]=z[i-L];
5      else {
6        L=i;
7        if(i>R) R=i;
8        while(R<n && s[R-L]==s[R]) R++;
9        z[i]=(R--)-L;
10     }
11   }
12 }
```

10 無權邊的生成樹個數 Kirchhoff's Theorem

1. 定義 $n \times m$ 矩陣 $E = (a_{i,j})$，n 為點數，m 為邊數，若 i 點在 j 邊上，i 為小點 $a_{i,j} = 1$，i 為大點 $a_{i,j} = -1$，否則 $a_{i,j} = 0$。
(證明省略)
4. 令 $E(E^T) = Q$，他是一種有負號的 kirchhoff 的矩陣，取 Q 的子矩陣即為 $F(F^T)$
結論：做 Q 取子矩陣算 det 即為所求。(除去第一行第一列 by mz)

11 monge

$i \le i' < j \le j'$
$m(i,j) + m(i',j') \le m(i',j) + m(i,j')$
$k(i,j-1) <= k(i,j) <= k(i+1,j)$

12 四心

$\frac{sa*A+sb*B+sc*C}{sa+sb+sc}$
外心 sin 2A : sin 2B : sin 2C
內心 sin A : sin B : sin C
垂心 tan A : tan B : tan C
重心 1 : 1 : 1

13 Runge-Kutta

$y_{n+1} = y_n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4)$
$k_1 = f(t_n, y_n)$
$k_2 = f(t_n + \frac{h}{2}, y_n + \frac{h}{2}k_2)$
$k_3 = f(t_n + \frac{h}{2}, y_n + \frac{h}{2}k_3)$
$k_2 = f(t_n + h, y_n + hk_3)$

14 Householder Matrix

$I - 2\frac{vv^T}{v^Tv}$