# NCTU_Yggdarsill

## Contents

# 1 Building Environment

## 1.1 Print

```
cat -n "%s" > tmp.print
lpr tmp.print
```

## 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 <bits/stdc++.h>
using namespace std;

const double PI = 3.1415926535897932;

struct Complex{
    typedef double T;
    T x, y;
    Complex (T _x=0.0, T _y=0.0)
        :x(_x),y(_y){ }
    Complex operator + (const Complex &b) { return Complex(x+b.x,y+b.y); }
    Complex operator - (const Complex &b) { return Complex(x-b.x,y-b.y); }
    Complex operator * (const Complex &b) { return Complex(x*b.x-y*b.y,x*b.y+y*b.x); }
};

void BitReverse(Complex *a, int n){
    for (int i=1, j=0; i<n; i++){
        for (int k = n>>1; k>(j^=k); k>>=1);
        if (i<j) swap(a[i],a[j]);
    }
}

void FFT(Complex *a, int n, int rev=1){ // rev = 1 or -1
    BitReverse(a,n);
    Complex *A = a;

    for (int s=1; (1<<s)<=n; s++){
        int m = (1<<s);

        Complex wm( cos(2*PI*rev/m) , sin(2*PI*rev/m) );
        for (int k=0; k<n; k+=m){
            Complex w(1,0);
            for (int j=0; j<(m>>1); j++){
                Complex t = w * A[k+j+(m>>1)];
                Complex u = A[k+j];
                A[k+j] = u+t;
                A[k+j+(m>>1)] = u-t;
                w = w*wm;
            }
        }
    }

    if (rev==-1){
        for (int i=0; i<n; i++){
            A[i].x /= n;
            A[i].y /= n;
        }
    }
}

const int MAXN = 65536;
int n;
Complex a[MAXN], b[MAXN];

void input(){
    scanf("%d", &n);

    for (int i=0 ,ai; i<n; i++){
        scanf("%d", &ai);
        a[i] = Complex(ai,0);
    }

    for (int i=0, bi; i<n; i++){
        scanf("%d", &bi);
        b[i] = Complex(bi,0);
    }

    for (int i=n; i<MAXN; i++){
        a[i] = b[i] = Complex(0,0);
    }
}

void solve(){
    FFT(a,MAXN);
    FFT(b,MAXN);

    for (int i=0; i<MAXN; i++){
        a[i] = a[i]*b[i];
    }

    FFT(a,MAXN,-1);
    for (int i=0; i<2*n-1; i++){
        printf("%.0f%c", a[i].x, i==2*n-2?'\n':' ');
    }
}

int main(){
    int T; scanf("%d",&T);

    while (T--){
        input();
        solve();
    }
}
```

# 3 Data Structure

## 3.1 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{
```

```cpp
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];
    /*mndist=std::min(mndist,dist);*/
    if(dist<mndist){
      pQ.push(std::make_pair(dist,u->pid));
      if((int)pQ.size()==qM+1){
        mndist=pQ.top().first,pQ.pop();
      }
    }
    if(x.d[k]<u->pid.d[k]){
      nearest(u->l,(k+1)%kd,x,h,mndist);
      h[k]=std::abs(x.d[k]-u->pid.d[k]);
      nearest(u->r,(k+1)%kd,x,h,mndist);
    }else{
      nearest(u->r,(k+1)%kd,x,h,mndist);
      h[k]=std::abs(x.d[k]-u->pid.d[k]);
      nearest(u->l,(k+1)%kd,x,h,mndist);
    }
    h[k]=old;
  }
public:
  kd_tree(const T &INF,double a=0.75):root(0),alpha(a),loga(log2(1.0/a)),INF(INF){}
  inline void clear(){
    clear(root),root=0;
  }
  inline void build(int n,const point *p){
    clear(root),A.resize(n);
```

```
      for(int i=0;i<n;++i)A[i]=new node(p[i]);
      root=build(0,0,n-1);
    }
    inline void insert(const point &x){
      insert(root,0,x,std::__lg(size(root))/loga);
    }
    inline T nearest(const point &x,int k){
      qM=k;
      T mndist=INF,h[kd]={};
      nearest(root,0,x,h,mndist);
      mndist=pQ.top().first;
      pQ=std::priority_queue<std::pair<T,point > >();
      return mndist;/*回傳離x第k近的點的距離*/
    }
    inline int size(){return root?root->s:0;}
};
#endif
```

## 3.2 Segment Tree (Lazy)

```
/* ⊠于⊠⊠求和 */
void push_up(int rt) {
    tree[rt] = tree[rt << 1] + tree[rt << 1 | 1];
}

/* ⊠于⊠⊠求最大值 */
void push_up(int rt) {
    tree[rt] = max(tree[rt << 1], tree[rt << 1 | 1]);
}

void push_down(int rt, int len) {
    tree[rt << 1] += lazy[rt] * (len - (len >> 1));
    lazy[rt << 1] += lazy[rt];
    tree[rt << 1 | 1] += lazy[rt] * (len >> 1);
    lazy[rt << 1 | 1] += lazy[rt];
    lazy[rt] = 0;
}

#define lchild rt << 1, l, m
#define rchild rt << 1 | 1, m + 1, r
void build(int rt = 1, int l = 1, int r = N) {
    if (l == r) { std::cin >> tree[rt]; return; }
    int m = (l + r) >> 1;
    build(lchild); build(rchild);
    push_up(rt);
}

#define lchild rt << 1, l, m
#define rchild rt << 1 | 1, m + 1, r
void update(int L, int R, int delta, int rt = 1, int l = 1, int r = N) {
    if (L <= l && r <= R) {
        tree[rt] += delta * (r - l + 1);
        lazy[rt] += delta;
        return;
    }
    if (lazy[rt]) push_down(rt, r - l + 1);
    int m = (l + r) >> 1;
    if (L <= m) update(L, R, delta, lchild);
    if (R > m)  update(L, R, delta, rchild);
    push_up(rt);
}

#define lchild rt << 1, l, m
#define rchild rt << 1 | 1, m + 1, r
int query(int L, int R, int rt = 1, int l = 1, int r = N) {
    if (L <= l && r <= R) return tree[rt];
    if (lazy[rt]) push_down(rt, r - l + 1);
    int m = (l + r) >> 1, ret = 0;
    if (L <= m) ret += query(L, R, lchild);
    if (R > m)  ret += query(L, R, rchild);
    return ret;
}
```

## 3.3 Treap

```
struct Treap{
  Treap *l,*r;
  int pri,sz,val,add;
  Treap(int _val):pri(rand()),sz(1),val(_val),add(0),l(NULL),r(NULL){}
};

int size(Treap *t){
  return t?t->sz:0;
}
void pull(Treap *t){
  t->sz=size(t->l)+size(t->r)+1;
}
void push(Treap *t){
  t->val+=t->add;
  if(t->l) t->l->add+=t->add;
  if(t->r) t->r->add+=t->add;
  t->add=0;
}
Treap* merge(Treap *a,Treap *b){
  if(!a||!b) return a?a:b;
  if(a->pri > b->pri){
    push(a);
    a->r = merge(a->r,b);
    pull(a);
    return a;
  }
  else{
    push(b);
    b->l = merge(a,b->l);
    pull(b);
    return b;
  }
}
```

```
void split(Treap *t,int k,Treap *&a,Treap *&b){
  if(!t) a=b=NULL;
  else{
    push(t);
    if(size(t->l) < k){
      a=t;
      split(t->r,k-size(t->l)-1,a->r,b);
      pull(a);
    }
    else{
      b=t;
      split(t->l,k,a,b->l);
      pull(b);
    }
  }
}
```

# 4 Geometry

## 4.1 Geometry

```
typedef double Double;
struct Point {
  Double x,y;

  bool operator < (const Point &b)const{
    //return tie(x,y) < tie(b.x,b.y);
    //return atan2(y,x) < atan2(b.y,b.x);
    assert(0 && "choose compare");
  }
  Point operator + (const Point &b)const{
    return (Point){x+b.x,y+b.y};
  }
  Point operator - (const Point &b)const{
    return (Point){x-b.x,y-b.y};
  }
  Point operator * (const Double &d)const{
    return Point(d*x,d*y);
  }
  Double operator * (const Point &b)const{
    return x*b.x + y*b.y;
  }
  Double operator % (const Point &b)const{
    return x*b.y - y*b.x;
  }
  friend Double abs2(const Point &p){
    return p.x*p.x + p.y*p.y;
  }
  friend Double abs(const Point &p){
    return sqrt( abs2(p) );
  }
};
typedef Point Vector;

struct Line{
  Point P; Vector v;
  bool operator < (const Line &b)const{
    return atan2(v.y,v.x) < atan2(b.v.y,b.v.x);
  }
};
```

## 4.2 Half Plane Intersection

```
bool OnLeft(const Line& L,const Point& p){
  return Cross(L.v,p-L.P)>0;
}
Point GetIntersection(Line a,Line b){
  Vector u = a.P-b.P;
  Double t = Cross(b.v,u)/Cross(a.v,b.v);
  return a.P + a.v*t;
}
int HalfplaneIntersection(Line* L,int n,Point* poly){
  sort(L,L+n);

  int first,last;
  Point *p = new Point[n];
  Line *q = new Line[n];
  q[first=last=0] = L[0];
  for(int i=1;i<n;i++){
    while(first < last && !OnLeft(L[i],p[last-1])) last--;
    while(first < last && !OnLeft(L[i],p[first])) first++;
    q[++last]=L[i];
    if(fabs(Cross(q[last].v,q[last-1].v))<EPS){
      last--;
      if(OnLeft(q[last],L[i].P)) q[last]=L[i];
    }
    if(first < last) p[last-1]=GetIntersection(q[last-1],q[last]);
  }
  while(first<last && !OnLeft(q[first],p[last-1])) last--;
  if(last-first<=1) return 0;
  p[last]=GetIntersection(q[last],q[first]);

  int m=0;
  for(int i=first;i<=last;i++) poly[m++]=p[i];
  return m;
}
```

## 4.3 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;
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';
}
```

## 4.4 Minimum Covering Circle

```
#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';
  }
}
```

# 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

```cpp
#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
// N: 10010, LOG: 15, INF: 1e9
// val[]: array that stores initial values
int n;
// ed: store input edges
struct edge ed[N];
vector<int> g[N];
int sz[N], dep[N];
int ts, tin[N], tout[N]; // timestamp
int par[N][LOG+1], head[N];
// head: head of the chain that contains u

void dfssz(int u, int p) {
  // precompute the size of each subtree
  par[u][0] = p;
  sz[u][1] = 1;
  head[u] = u;
  for (int v: g[u]) if (v != p) {
    dep[v] = dep[u] + 1;
    dfssz(v, u);
    sz[u] += sz[v];
  }
}

void dfshl(int u) {
  tin[u] = tout[u] = ++ts;
  sort(g[u].begin(), g[u].end(),
    [&](int a, int b) { return sz[a] > sz[b]; });
  bool flag = 1;
  for (int v: g[u]) if (v != par[u][0]) {
    if (flag) head[v] = head[u], flag = 0;
    dfshl(v);
  }
  tout[u] = ts;
}

inline bool anc(int a, int b) {
  return tin[a] <= tin[b] && tout[b] <= tout[a];
}
```

```

inline bool lca(int a, int b) {
  if (anc(b, a)) return b;
  for (int j = LOG ; j >= 0 ; --j)
    if (!anc(par[b][j], a))
      b = par[b][j];
  return par[b][0];
}
vector<pii> getPath(int u, int v) {
  // u must be ancestor of v
  // return a list of intervals from u to v
  vector<pii> res;
  while (tin[u] < tin[head[v]]) {
    res.push_back(pii(tin[head[v]], tin[v]));
    v = par[head[v]][0];
  }
  if (tin[u] + 1 <= tin[v])
    res.push_back(pii(tin[u]+1, tin[v]));
  return res;
}
void init() {
  cin >> n;
  for (int i = 1 ; i < n ; ++i) {
    int u, v, vl;
    cin >> u >> v >> vl;
    ed[i] = edge(u, v, vl);
    g[u].push_back(v);
    g[v].push_back(u);
  }
  // do Heavy-Light Decomp.
  int root = 1; // set root node
  dep[root] = 1;
  dfssz(root, root);
  ts = 0;
  dfshl(root);
  for (int k = 1 ; k <= LOG ; ++k)
    for (int i = 1 ; i <= n ; ++i)
      par[i][k] = par[par[i][k-1]][k-1];
  // set initial values
  for (int i = 1 ; i < n ; ++i) {
    if (dep[ed[i].u] < dep[ed[i].v])
      swap(ed[i].u, ed[i].v);
    val[tin[ed[i].u]] = ed[i].vl;
  }
}
```

## 5.5 SCC

```
// Kosaraju - Find SCC by twice dfs, and the SCC DAG is in the Topology
// ordering.
// Owner: samsam2310
//
#include <bits/stdc++.h>
#define N 300002 // Maximum number of vertices
using namespace std;
vector<int> forward_graph[N];  // original graph
vector<int> backward_graph[N]; // reverse graph
vector<int> dag_graph[N];      // result dag graph(graph of scc)
int scc[N];                    // SCC index of a vertex
bool visit[N];
void init() {
    fill(forward_graph, forward_graph + N, vector<int>());
    fill(backward_graph, backward_graph + N, vector<int>());
    fill(dag_graph, dag_graph + N, vector<int>());
}
void dfs(vector<int> &graph, int now, int scc_id,
         stack<int> *leave_order = NULL) {
    visit[now] = true;
    if (scc != -1) {
        scc[now] = scc_id;
    }
    for (int v : graph[now]) {
        if (!visit[v]) {
            dfs(graph, v, scc_id, leave_order);
        }
    }
    if (leave_order) {
        leave_order->push(now);
    }
}
int main(int argc, char *argv[]) {
    ios_base::sync_with_stdio(false);
    cin.tie(0);
    init();
    cin >> n;
    for (int i = 0; i < n; ++i) {
        int a, b; // edge of a -> b
        cin >> a >> b;
        forward_graph[a].push_back(b);
        backward_graph[b].push_back(a);
    }
    // Find the SCC.
    memset(visit, 0, sizeof(visit));
    stack<int> leave_order;
    for (int i = 0; i < n; ++i) {
        if (!visit[i]) {
            dfs(forward_graph, i, -1, &leave_order);
        }
    }
    memset(visit, 0, sizeof(visit));
    int scc_id = 0;
    while (!leave_order.empty()) {
        int v = leave_order.top();
        leave_order.pop();
        if (!visit[v]) {
            dfs(backward_graph, i, scc_id, NULL);
            ++scc_id;
        }
    }
    // Build the SCC DAG.
    for (int i = 0; i < n; ++i) {
        for (int v : forward_graph[i]) {
            if (scc[i] != scc[v]) {
                dag_graph[scc[i]].push_back(scc[v]);
            }
        }
    }
    return 0;
}
```

# 6 Java

## 6.1 Big Integer

```
import java.math.*;
import java.io.*;
import java.util.*;
public class Main{
    public static void main(String []argv){
        c[0][0]=BigInteger.ONE;
        for(int i=1;i<3001;i++){
            c[i][0]=BigInteger.ONE;
            c[i][i]=BigInteger.ONE;
            for(int j=1;j<i;j++)c[i][j]=c[i-1][j].add(c[i-1][j-1]);
        }
        Scanner scanner = new Scanner(System.in);
        int T = scanner.nextInt();
        BigInteger x;
        BigInteger ans;
        while(T-- > 0){
            ans = BigInteger.ZERO;
            int n = scanner.nextInt();
            for(int i=0;i<n;i++){
                x = new BigInteger(scanner.next());
                if(i%2 == 1)ans=ans.subtract(c[n-1][i].multiply(x));
                else ans=ans.add(c[n-1][i].multiply(x));
            }
            if(n%2 == 0)ans=BigInteger.ZERO.subtract(ans);
            System.out.println(ans);
        }
    }
}
```

## 6.2 Prime

```
import java.math.*;
import java.io.*;
import java.util.*;
public class Main{
    public static void main(String []argv){
        Scanner scanner = new Scanner(System.in);
        int T = scanner.nextInt();
        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 KM

```
#define MAXN 100
#define INF INT_MAX
int g[MAXN][MAXN],lx[MAXN],ly[MAXN],slack_y[MAXN];
int px[MAXN],py[MAXN],match_y[MAXN],par[MAXN];
int n;
void adjust(int y){//把增廣路上所有邊反轉
  match_y[y]=py[y];
  if(px[match_y[y]]!=-2)
    adjust(px[match_y[y]]);
}
bool dfs(int x){//DFS找增廣路
  for(int y=0;y<n;++y){
    if(py[y]!=-1)continue;
    int t=lx[x]+ly[y]-g[x][y];
    if(t==0){
      py[y]=x;
      if(match_y[y]==-1){
        adjust(y);
        return 1;
      }
      if(px[match_y[y]]!=-1)continue;
      px[match_y[y]]=y;
      if(dfs(match_y[y]))return 1;
    }else if(slack_y[y]>t){
      slack_y[y]=t;
      par[y]=x;
    }
  }
  return 0;
}
inline int km(){
  memset(ly,0,sizeof(int)*n);
  memset(match_y,-1,sizeof(int)*n);
  for(int x=0;x<n;++x){
    lx[x]=-INF;
    for(int y=0;y<n;++y){
      lx[x]=max(lx[x],g[x][y]);
    }
  }
  for(int x=0;x<n;++x){
    for(int y=0;y<n;++y)slack_y[y]=INF;
    memset(px,-1,sizeof(int)*n);
    memset(py,-1,sizeof(int)*n);
    px[x]=-2;
    if(dfs(x))continue;
    bool flag=1;
    while(flag){
      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;
}
```

## 7.5 Min Cost Flow

```
#include<bits/stdc++.h>
using namespace std;
#define int long long
typedef pair<int,int> P;
struct edge{
    edge(){}
    edge(int a,int b,int c,int d):to(a),cap(b),cost(c),rev(d){}
    int to,cap,cost,rev;
};
#define V 210
#define inf 1000000000000000
vector<edge> g[V];
int h[V],dist[V],prev_v[V],prev_e[V];
void add_edge(int from,int to,int cap,int cost){
    g[from].push_back(edge(to,cap,cost,g[to].size()));
    g[to].push_back(edge(from,0,-cost,g[from].size()-1));
}
int min_costflow(int s,int t,int f){
    int res=0;
    memset(h,0,sizeof(h));
    while(f>0){
        priority_queue<P,vector<P>,greater<P> >que;
        fill(dist,dist+V,inf);
        dist[s]=0;
        que.push(P(dist[s],s));
        while(!que.empty()){
            P p=que.top();
            que.pop();
            int v=p.second;
            if(dist[v]<p.first)continue;
            for(int i=0;i<g[v].size();++i){
                edge &e=g[v][i];
                if(e.cap>0&&dist[e.to]>dist[v]+e.cost+h[v]-h[e.to]){
                    dist[e.to]=dist[v]+e.cost+h[v]-h[e.to];
                    prev_v[e.to]=v;
                    prev_e[e.to]=i;
                    que.push(P(dist[e.to],e.to));
                }
            }
        }
        if(dist[t]==inf) return -1;
        for(int v=0;v<V;++v)h[v]+=dist[v];
        int d=f;
        for(int v=t;v!=s;v=prev_v[v]) d=min(d,g[prev_v[v]][prev_e[v]].cap);
        f-=d;
        res+=d*h[t];
        for(int v=t;v!=s;v=prev_v[v]){
            edge &e=g[prev_v[v]][prev_e[v]];
            e.cap-=d;
            g[v][e.rev].cap+=d;
        }
    }
    return res;
}
#undef int
int main()
{
#define int long long
    int T,n,m,cost,l,s,t,ans;
    cin>>T;
    while(T--){
      cin>>n>>m;
        for(int q=0;q<V;++q)g[q].clear();
        s=m+n;
        t=m+n+1;
        for(int i=0;i<n;++i)
          for(int j=0;j<m;++j){
            cin>>cost;
            if(cost>0)
              add_edge(n+j,i,1,cost);
          }
        for(int i=0;i<m;++i){
          cin>>l;
          add_edge(s,n+i,1,0);
        }
        for(int i=0;i<n;++i)
          add_edge(i,t,1,0);
        ans=min_costflow(s,t,n);
        cout<<ans<<endl;
    }
    return 0;
}
```

## 7.6 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

```
template <typename T>
T extgcd(T a, T b, T &x, T &y){
    // g = a * x + b * y
    T g = a;
    if (b != 0) {
        g = extgcd(b, a % b, y, x);
        y -= (a / b) * x;
    }
    else x = 1, y = 0;
    return g;
}
```

## 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;
}
```

# 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 KMP

```
template<typename T>
void build_KMP(int n, T *s, int *f){ // 1 base
  f[0]=-1, f[1]=0;
  for (int i=2; i<=n; i++){
    int w = f[i-1];
    while (w>=0 && s[w+1]!=s[i])w = f[w];
    f[i]=w+1;
  }
}

template<typename T>
int KMP(int n, T *a, int m, T *b){
  build_KMP(m,b,f);
  int ans=0;

  for (int i=1, w=0; i<=n; i++){
    while ( w>=0 && b[w+1]!=a[i] )w = f[w];
    w++;
    if (w==m){
      ans++;
      w=f[w];
    }
  }
  return ans;
}
```

## 9.4 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;
    }
}
```

## 9.5 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;
      }
    }
    if(!nw->pre) nw->pre = root;
    ed = ed->nxt[c];
  }
  void init(){
    while(top){
      mem[--top].pos.clear();
    }
    root = new_node(0);
    ed = root;
  }
  void push(char *s){
    for(int i=0;s[i];i++) push(s[i]-32);
  }
  long long count(){
    long long ans=0;
    for(int i=1;i<top;i++){
      ans+=mem[i].len-mem[i].pre->len;
    }
    return ans;
  }
}sam;
char S[N];
int main(){
  int T;
  scanf("%d",&T);
  while(T--){
    scanf("%s",S);
    sam.build(S);
    printf("%lld\n",sam.count());
  }
  return 0;
}
```

## 9.6 Z Algorithm

```
void Zalg(char *s, int *z, int n) {
  z[0]=n;
  for(int L=0, R=0, i=1; i<n; i++) {
    if(i<=R && z[i-L]<=R-i) z[i]=z[i-L];
    else {
      L=i;
      if(i>R) R=i;
      while(R<n && s[R-L]==s[R]) R++;
      z[i]=(R--)-L;
    }
  }
}
```

## 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

$i \leq i^{'} < j \leq j^{'}$

$m(i,j) + m(i^{'}, j^{'}) \leq m(i^{'}, j) + m(i, j^{'})$

$k(i, j-1) <= k(i,j) <= k(i+1, j)$

## 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

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