ACM Template

UESTC_Dagon

Last build at July 9, 2014

Contents

1 Datastructure

1.1 KD tree

```cpp
bool Div[MaxN];
void BuildKD(int deep,int l, int r, Point p[]) {
  if (l > r) return;
  int mid = l + r >> 1;
  int minX, minY, maxX, maxY;
  minX = min_element(p + l, p + r + 1, cmpX)->x;
  minY = min_element(p + l, p + r + 1, cmpY)->y;
  maxX = max_element(p + l, p + r + 1, cmpX)->x;
  maxY = max_element(p + l, p + r + 1, cmpY)->y;
  Div[mid] = (maxX - minX >= maxY - minY);
  nth_element(p + l, p + mid, p + r + 1, Div[mid] ? cmpX : cmpY);
  BuildKD(l, mid - 1, p);
  BuildKD(mid + 1, r, p);
}
long long res;
void Find(int l, int r, Point a, Point p[]) {
  if (l > r)  return;
  int mid = l + r >> 1;
  long long dist = dist2(a, p[mid]);
  if (dist > 0)//NOTICE
    res = min(res, dist);
  long long d = Div[mid] ? (a.x - p[mid].x) : (a.y - p[mid].y);
  int l1, l2, r1, r2;
  l1 = l, l2 = mid + 1;
  r1 = mid - 1, r2 = r;
  if (d > 0)
    swap(l1, l2), swap(r1, r2);
  Find(l1, r1, a, p);
  if (d * d < res)
    Find(l2, r2, a, p);
}
```

1.2 Binary indexed tree

```cpp
int read(int k) {
  int sum = 0;
  for (; k; k^=k&-k) sum+=tree[k];
  return sum;
}
void update(int k, int v) {
  for (; k<=MaxN; k+=k&-k) tree[k]+=v;
}
int find_Kth(int k) {
  int idx = 0;
  for(int i=20; i>=0; i--) {
    idx |= 1 << i;
    if(idx <= MaxN && tree[idx] < k)
      k -= tree[idx];
    else  idx ^= 1 << i;
  }
  return idx + 1;
}
```

1.3 Splay

```cpp
//Node
struct Node {
  int size,key;
  Node *c[2], *p;
} mem[MaxN], *cur, *nil;
//Initialize functions without memory pool
Node *newNode(int v, Node *p) {
  cur->c[0] = cur->c[1] = nil, cur->p = p;
  cur->size = 1;
  cur->key = v;
  return cur++;
}
void Init() {
  cur = mem;
  nil = newNode(0, cur);
  nil->size = 0;
}
//Splay tree
struct SplayTree {
  Node *root;
  void Init() {
    root = nil;
  }
  void Pushup(Node *x) {
    if (x == nil)   return;
    Pushdown(x);
    Pushdown(x->c[0]);
    Pushdown(x->c[1]);
    x->size = x->c[0]->size + x->c[1]->size + 1;
  }
  void Pushdown(Node *x) {
    if (x == nil)   return;
    //do something
  }
  void Rotate(Node *x, int f) {
    if (x == nil)   return;
    Node *y = x->p;
    y->c[f ^ 1] = x->c[f], x->p = y->p;
    if (x->c[f] != nil)
      x->c[f]->p = y;
    if (y->p != nil)
      y->p->c[y->p->c[1] == y] = x;
    x->c[f] = y, y->p = x;
    Pushup(y);
  }
  void Splay(Node *x, Node *f) {
    static Node *stack[maxn];
    int top = 0;
    stack[top++] = x;
    for (Node *y = x; y != f; y = y->p)
      stack[top++] = y->p;
    while (top)
      Pushdown(stack[--top]);
    while (x->p != f) {
      Node *y = x->p;
      if (y->p == f)
        Rotate(x, x == y->c[0]);
      else {
        int fd = y->p->c[0] == y;
        if (y->c[fd] == x)
          Rotate(x, fd ^ 1), Rotate(x, fd);
        else
          Rotate(y, fd), Rotate(x, fd);
      }
    }
    Pushup(x);
    if (f == nil)
      root = x;
  }
  void Select(int k, Node *f) {
    Node *x = root;
    Pushdown(x);
    int tmp;
    while ((tmp = x->c[0]->size) != k) {
      if (k < tmp)  x = x->c[0];
      else
        x = x->c[1], k -= tmp + 1;
      Pushdown(x);
    }
    Splay(x, f);
  }
  void Select(int l, int r) {
    Select(l, nil), Select(r + 2, root);
  }
  Node *Make_tree(int a[], int l, int r, Node *p) {
    if (l > r)  return nil;
    int mid = l + r >> 1;
    Node *x = newNode(a[mid], p);
    x->c[0] = Make_tree(a, l, mid - 1, x);
    x->c[1] = Make_tree(a, mid + 1, r, x);
    Pushup(x);
    return x;
  }
  void Insert(int pos, int a[], int n) {
    Select(pos, nil), Select(pos + 1, root);
    root->c[1]->c[0] = Make_tree(a, 0, n - 1, root->c[1]);
    Splay(root->c[1]->c[0], nil);
  }
  void Insert(int v) {
    Node *x = root, *y = nil;
    //Need pushdown
    while (x != nil) {
      y = x;
      y->size++;
      x = x->c[v >= x->key];
    }
    y->c[v >= y->key] = x = newNode(v, y);
    Splay(x, nil);
  }
  void Remove(int l, int r) {
    Select(l, r);
    //Recycle(root->c[1]->c[0]);
    root->c[1]->c[0] = nil;
    Splay(root->c[1], nil);
  }
};
```

1.4 Dynamic tree

```
struct SplayTree {
  void Pushup(Node *x) {
    if (x == nil)   return;
    Pushdown(x);
    Pushdown(x->c[0]);
    Pushdown(x->c[1]);
    x->size = x->c[0]->size + x->c[1]->size + 1;
  }
  void Pushdown(Node *x) {
    if (x == nil)   return;
    if (x->rev) {
      x->rev = 0;
      x->c[0]->rev ^= 1;
      x->c[1]->rev ^= 1;
      swap(x->c[0], x->c[1]);
    }
  }
  bool isRoot(Node *x) {
    return (x == nil) || (x->p->c[0] != x && x->p->c[1] != x);
  }
  void Rotate(Node *x, int f) {
    if (isRoot(x))   return;
    Node *y = x->p;
    y->c[f ^ 1] = x->c[f], x->p = y->p;
    if (x->c[f] != nil)
      x->c[f]->p = y;
    if (y != nil) {
      if (y == y->p->c[1])
        y->p->c[1] = x;
      else if (y == y->p->c[0])
        y->p->c[0] = x;
    }
    x->c[f] = y, y->p = x;
    Pushup(y);
  }
  void Splay(Node *x) {
    static Node *stack[MaxN];
    int top = 0;
    stack[top++] = x;
    for (Node *y = x; !isRoot(y); y = y->p)
      stack[top++] = y->p;
    while (top)
      Pushdown(stack[--top]);
    while (!isRoot(x)) {
      Node *y = x->p;
      if (isRoot(y))
        Rotate(x, x == y->c[0]);
      else {
        int fd = y->p->c[0] == y;
        if (y->c[fd] == x)
          Rotate(x, fd ^ 1), Rotate(x, fd);
        else
          Rotate(y, fd), Rotate(x, fd);
      }
    }
    Pushup(x);
  }
  Node *Access(Node *u) {
    Node *v = nil;
    while (u != nil) {
      Splay(u);
      v->p = u;
      u->c[1] = v;
      Pushup(u);
      u = (v = u)->p;
      if (u == nil)
        return v;
    }
  }
  Node *LCA(Node *u, Node *v) {
    Access(u);
    return Access(v);
  }
  Node *Link(Node *u, Node *v) {
    Access(u);
    Splay(u);
    u->rev = true;
    u->p = v;
  }
  void ChangeRoot(Node *u) {
    Access(u)->rev ^= 1;
  }
  Node *GetRoute(Node *u, Node *v) {
    ChangeRoot(u);
    return Access(v);
  }
};
```

1.5 Partition tree

```
int n,m;
struct elem {
  int v,index;
} a[120000];
int d[30][120000];
int s[30][120000];
bool cmp(elem a,elem b) {
  if (a.v == b.v)
    return a.index <= b.index;
  return a.v < b.v;
}
void build(int depth,int l,int r) {
  if (l == r)
    return;
  int mid = (l+r)/2;
  int tl,tr;
  tl = tr = 0;
  for (int i = l; i <= r; i++) {
    if (cmp(a[d[depth][i]],a[mid])) {
      d[depth+1][l+tl] = d[depth][i];
      tl++;
    } else {
      d[depth+1][mid+1+tr] = d[depth][i];
      tr++;
    }
    s[depth][i] = tl;
  }
  build(depth+1,l,mid);
  build(depth+1,mid+1,r);
}
int find(int depth,int dl,int dr,int fl,int fr,int k) {
  if (fl == fr)
    return a[d[depth][fl]].v;
  int ls,rs;
  int mid = (dl+dr)/2;
  ls = (fl == dl)? 0 : s[depth][fl-1];
  rs = s[depth][fr];
  return (rs-ls < k)?
         find(depth+1,mid+1,dr,mid+fl-dl-ls+1,mid+fr-dl-rs+1,k-(rs-ls))
         : find(depth+1,dl,mid,dl+ls,dl+rs-1,k);
}
int main() {
  while (scanf("%d%d",&n,&m) != EOF) {
    for (int i = 1; i <= n; i++) {
      scanf("%d",&a[i].v);
      a[i].index = i;
    }
    sort(a+1,a+n+1,cmp);
    for (int i = 1; i <= n; i++)
      d[0][a[i].index] = i;
    build(0,1,n);
    int l,r,k;
    for (int i = 1; i <= m; i++) {
      scanf("%d%d%d",&l,&r,&k);
      printf("%d\n",find(0,1,n,l,r,k));
    }
  }
  return 0;
}
```

2 Dynamic programming

2.1 RMQ

```
void init() {
  int i,j;
  int n=N,k=1,l=0;
  for (i=0; i<n; i++) {
    f[i][0]=ele[i].num;
    if (i+1>k*2) {
      k*=2;
      l++;
    }
    lent[i+1]=l;
  }
  for (j=1; (1<<j)-1<n; j++)
    for (i=0; i+(1<<j)-1<n; i++)
      f[i][j]=max(f[i][j-1],f[i+(1<<(j-1))][j-1]);
}
int fint(int x,int y) {
  int k=lent[y-x+1];
  return max(f[x][k],f[y-(1<<k)+1][k]);
}
```

2.2 2D-LIS

```
#include<cstdio>
#include<map>
using namespace std;
map<int,int> mp[100001];
bool check(int idx,int x,int y) {
  if (!idx) return 1;
  if (mp[idx].begin()->first>=x) return 0;
  map<int,int> ::iterator it=mp[idx].lower_bound(x);
  it--;
  if (it->second<y) return 1;
  else return 0;
}
int main() {
  int n;
  scanf("%d",&n);
  int l=0,r=0;
  for (int i=0; i<n; i++) {
    int x,y;
    scanf("%d%d",&x,&y);
    int tl=l,tr=r;
    while (tl<tr) {
      int mid=(tl+tr+1)/2;
      if (check(mid,x,y))
        tl=mid;
      else
        tr=mid-1;
    }
    if (tl==r) r++;
    int idx=tl+1;
    map<int,int> ::iterator itl=mp[idx].lower_bound(x),itr=itl;
    while (itr!=mp[idx].end() && itr->second>y) itr++;
    if (mp[idx].find(x)!=mp[idx].end())
      y=min(y,mp[idx][x]);
    if (itl!=itr) mp[idx].erase(itl,itr);
    if (mp[idx].find(x)==mp[idx].end() || mp[idx][x]>y)
      mp[idx][x]=y;
  }
  printf("%d\n",r);
  return 0;
}
```

3 Geometry

3.1 2D

3.1.1 Point

```
//Use cross product instead of atan2
bool cmp(const Point& a,const Point& b) {
  if (a.y*b.y <= 0) {
    if (a.y > 0 || b.y > 0) return a.y < b.y;
    if (a.y == 0 && b.y == 0) return a.x < b.x;
  }
  return a*b > 0;
}
```

3.1.2 Line

```
Point operator &(const Line& b) const {
  Point res = s;
  double t = ((s - b.s) * (b.s - b.e)) / ((s - e) * (b.s - b.e));
  res.x += (e.x - s.x) * t;
  res.y += (e.y - s.y) * t;
  return res;
}
```

3.1.3 Functions

```
Point nearestPointToLine(Point P, Line L) {
  Point result;
  double a, b, t;
  a = L.e.x-L.s.x;
  b = L.e.y-L.s.y;
  t = ( (P.x-L.s.x)*a+(P.y-L.s.y)*b )/(a*a+b*b);
  if (t >= 0 && t <= 1) {
    result.x = L.s.x+a*t;
    result.y = L.s.y+b*t;
  }
  return result;
}
//Segment
bool inter(Line l1,Line l2) {
  return
    max(l1.s.x,l1.e.x) >= min(l2.s.x,l2.e.x) &&
    max(l2.s.x,l2.e.x) >= min(l1.s.x,l1.e.x) &&
    max(l1.s.y,l1.e.y) >= min(l2.s.y,l2.e.y) &&
    max(l2.s.y,l2.e.y) >= min(l1.s.y,l1.e.y) &&
    sgn((l2.s-l1.s)*(l1.e-l1.s))*sgn((l2.e-l1.s)*(l1.e-l1.s)) <= 0 &&
    sgn((l1.s-l2.s)*(l2.e-l2.s))*sgn((l1.e-l2.s)*(l2.e-l2.s)) <= 0;
}
bool onSeg(Line a,Point b) {
  return ((a.s-b)*(a.e-b) == 0 &&
          (b.x-a.s.x)*(b.x-a.e.x) <= 0 &&
          (b.y-a.s.y)*(b.y-a.e.y) <= 0);
}
int inPoly(Point p,Point poly[], int n) {
  int i, count;
  Line ray, side;
  count = 0;
  ray.s = p;
  ray.e.y  = p.y;
  ray.e.x  = -1;//-∞
  for (i = 0; i < n; i++) {
    side.s = poly[i];
    side.e = poly[(i+1)%n];
    if(OnSeg(p, side))
      return 1;
    if (side.s.y == side.e.y)
      continue;
    if (OnSeg(side.s, ray)) {
      if (side.s.y > side.e.y) count++;
    } else if (OnSeg(side.e, ray)) {
      if (side.e.y > side.s.y) count++;
    } else if (inter(ray, side)) {
      count++;
    }
  }
  return ((count % 2 == 1) ? 0 : 2);
}
Point centerOfPolygon(Point poly[],int n) {
  Point p, p0, p1, p2, p3;
  double m, m0;
  p1 = poly[0];
  p2 = poly[1];
  p.x = p.y = m = 0;
  for (int i = 2; i < n; i++) {
    p3 = poly[i];
    p0.x = (p1.x + p2.x + p3.x) / 3.0;
    p0.y = (p1.y + p2.y + p3.y) / 3.0;
    m0 = p1.x*p2.y+p2.x*p3.y+p3.x*p1.y-p1.y*p2.x-p2.y*p3.x-p3.y*p1.x;
    if (cmp(m + m0,0.0) == 0)
      m0 += eps;
    p.x = (m * p.x + m0 * p0.x) / (m + m0);
    p.y = (m * p.y + m0 * p0.y) / (m + m0);
    m = m + m0;
    p2 = p3;
  }
  return p;
}
```

3.1.4 Half plane intersection

```
bool HPIcmp(Line a, Line b) {
  if (fabs(a.k - b.k) > EPS)  return a.k < b.k;
  return ((a.s - b.s) * (b.e - b.s)) < 0;
}
Line Q[MAXN];
void HPI(Line line[], int n, Point res[], int &resn) {
  int tot = n;
  sort(line, line + n, HPIcmp);
  tot = 1;
  for (int i = 1; i < n; i++)
    if (fabs(line[i].k - line[i - 1].k) > EPS)
      line[tot++] = line[i];
  int head = 0, tail = 1;
  Q[0] = line[0];
  Q[1] = line[1];
  resn = 0;
  for (int i = 2; i < tot; i++) {
    if (fabs((Q[tail].e - Q[tail].s) * (Q[tail - 1].e - Q[tail - 1].s)) < EPS ||
        fabs((Q[head].e - Q[head].s) * (Q[head + 1].e - Q[head + 1].s)) < EPS)
      return;
    while (head < tail && (((Q[tail] & Q[tail - 1]) - line[i].s) * (line[i].e - line[i].s)) > EPS)
      tail--;
    while (head < tail && (((Q[head] & Q[head + 1]) - line[i].s) * (line[i].e - line[i].s)) > EPS)
      head++;
    Q[++tail] = line[i];
  }
  while (head < tail && (((Q[tail] & Q[tail - 1]) - Q[head].s) * (Q[head].e - Q[head].s)) > EPS)
    tail--;
  while (head < tail && (((Q[head] & Q[head + 1]) - Q[tail].s) * (Q[tail].e - Q[tail].s)) > EPS)
    head++;
  if (tail <= head + 1)   return;
  for (int i = head; i < tail; i++)
    res[resn++] = Q[i] & Q[i + 1];
  if (head < tail + 1)
    res[resn++] = Q[head] & Q[tail];
}
```

3.1.5 Convex hull

```
bool GScmp(Point a, Point b) {
  if (fabs(a.x - b.x) < eps)
    return a.y < b.y - eps;
  return a.x < b.x - eps;
}
void GS(Point p[],int n,Point res[],int &resn) {
  resn = 0;
  int top = 0;
  sort(p,p+n,GScmp);
  if (conPoint(p,n)) {
    res[resn++] = p[0];
    return;
  }
  if (conLine(p,n)) {
    res[resn++] = p[0];
    res[resn++] = p[n-1];
    return;
  }
  for (int i = 0; i < n;)
    if (resn < 2 ||
        (res[resn-1]-res[resn-2])*(p[i]-res[resn-1]) > 0)
      res[resn++] = p[i++];
    else
```

```
      --resn;
  top = resn-1;
  for (int i = n-2; i >= 0;)
    if (resn < top+2 ||
        (res[resn-1]-res[resn-2])*(p[i]-res[resn-1]) > 0)
      res[resn++] = p[i--];
    else
      --resn;
  resn--;
}
```

3.1.6 Intersections of line and polygon

```
//Intersecting segment between $[la,lb]$
int Gao(int la,int lb,Line line) {
  if (la > lb)
    lb += n;
  int l = la,r = lb,mid;
  while (l < r) {
    mid = l+r+1>>1;
    if (cmp((line.e-line.s)*(p[la]-line.s),0)*cmp((line.e-line.s)*(p[mid]-line.s),0) >= 0)
      l = mid;
    else
      r = mid-1;
  }
  return l%n;
}
double theta[maxn];
void Gettheta() {
  for (int i = 0; i < n; i++) {
    Point v = p[(i+1)%n]-p[i];
    theta[i] = atan2(v.y,v.x);
  }
  for (int i = 1; i < n; i++)
    if (theta[i-1] > theta[i]+eps)
      theta[i] += 2*pi;
}
void Calc(Line l) {
  double tnow;
  Point v = l.e-l.s;
  tnow = atan2(v.y,v.x);
  if (cmp(tnow,theta[0]) < 0) tnow += 2*pi;
  int pl = lower_bound(theta,theta+n,tnow)-theta;
  tnow = atan2(-v.y,-v.x);
  if (cmp(tnow,theta[0]) < 0) tnow += 2*pi;
  int pr = lower_bound(theta,theta+n,tnow)-theta;
  //Farest points with $l$ on polygon
  pl = pl%n;
  pr = pr%n;
  if (cmp(v*(p[pl]-l.s),0)*cmp(v*(p[pr]-l.s),0) >= 0)
    return 0.0;
  int xa = Gao(pl,pr,l);
  int xb = Gao(pr,pl,l);
  if (xa > xb)  swap(xa,xb);
  //Intersecting with line $P_{xa} \rightarrow P_{xa+1}$ and $P_{xb} \rightarrow P_{xb+1}$
  if (cmp(v*(p[xa+1]-p[xa]),0) == 0)  return 0.0;
  if (cmp(v*(p[xb+1]-p[xb]),0) == 0)  return 0.0;
  Point pa,pb;
  //Intersections
  pa = Line(p[xa],p[xa+1])&l;
  pb = Line(p[xb],p[xb+1])&l;
}
```

3.2 3D

3.2.1 Point

```
Point3D operator *(const Point3D& b)const {
  return Point3D(y*b.z-z*b.y,z*b.x-x*b.z,x*b.y-y*b.x);
}
//Rotate around $V$, notice that $|V| = 1$
Point3D Trans(Point3D pa,Point3D V,double theta) {
  double s = sin(theta);
  double c = cos(theta);
  double x,y,z;
  x = V.x;
  y = V.y;
  z = V.z;
  Point3D pp =
    Point3D(
      (x*x*(1-c)+c)*pa.x+(x*y*(1-c)-z*s)*pa.y+(x*z*(1-c)+y*s)*pa.z,
      (y*x*(1-c)+z*s)*pa.x+(y*y*(1-c)+c)*pa.y+(y*z*(1-c)-x*s)*pa.z,
      (x*z*(1-c)-y*s)*pa.x+(y*z*(1-c)+x*s)*pa.y+(z*z*(1-c)+c)*pa.z);
  return pp;
}
```

3.2.2 Functions

```
bool lineIntersect(Line3D L1, Line3D L2) {
  Point3D s = L1.s-L1.e;
  Point3D e = L2.s-L2.e;
  Point3D p  = s*e;
  if (ZERO(p)) return false;    //Parallel
  p = (L2.s-L1.e)*(L1.s-L1.e);
  return ZERO(p&L2.e);          //Common face
}
//Please check whether $a$, $b$, $c$, $d$ on a plane first
bool segmentIntersect(Point a,Point b,Point c,Point d) {
  Point ret = (a-b)*(c-d);
  Point t1 = (b-a)*(c-a);
  Point t2 = (b-a)*(d-a);
  Point t3 = (d-c)*(a-c);
  Point t4 = (d-c)*(b-c);
  return sgn(t1&ret)*sgn(t2&ret) < 0 &&
         sgn(t3&ret)*sgn(t4&ret) < 0;
}
//Distance from point $p$ to line $L$
double distance(Point3D p, Line3D L) {
  return (Norm((p-L.s)*(L.e-L.s))/Norm(L.e-L.s));
}
//Angle between line $L_1$ and $L_2$, $\theta \in [0, \pi]$
double calcTheta(Line3D L1, Line3D L2) {
  Point3D u = L1.e - L1.s;
  Point3D v = L2.e - L2.s;
  return acos( (u & v) / (Norm(u)*Norm(v)) );
}
```

3.2.3 Convex hull

Don't forget **Randomshuffle**!

```
struct pt {
  double x, y, z;
  pt() {}
  pt(double _x, double _y, double _z): x(_x), y(_y), z(_z) {}
  pt operator - (const pt p1) {}
  pt operator * (pt p) {}
  double operator ^ (pt p) {}
};
struct _3DCH {
  struct fac {
    int a, b, c;
    bool ok;
  };
  int n;
  pt P[MAXV];
  int cnt;
  fac F[MAXV*8];
  int to[MAXV][MAXV];
  double vlen(pt a) {
    return sqrt(a.x*a.x+a.y*a.y+a.z*a.z);
  }
  double area(pt a, pt b, pt c) {
    return vlen((b-a)*(c-a));
  }
  double volume(pt a, pt b, pt c, pt d) {
    return (b-a)*(c-a)^(d-a);
  }
  double ptof(pt &p, fac &f) {
    pt m = P[f.b]-P[f.a], n = P[f.c]-P[f.a], t = p-P[f.a];
    return (m * n) ^ t;
  }
  void deal(int p, int a, int b) {
    int f = to[a][b];
    fac add;
    if (F[f].ok) {
      if (ptof(P[p], F[f]) > eps)
        dfs(p, f);
      else {
        add.a = b, add.b = a, add.c = p, add.ok = 1;
        to[p][b] = to[a][p] = to[b][a] = cnt;
        F[cnt++] = add;
      }
    }
  }
  void dfs(int p, int cur) {
    F[cur].ok = 0;
    deal(p, F[cur].b, F[cur].a);
    deal(p, F[cur].c, F[cur].b);
    deal(p, F[cur].a, F[cur].c);
  }
  bool same(int s, int t) {
    pt &a = P[F[s].a], &b = P[F[s].b], &c = P[F[s].c];
    return fabs(volume(a, b, c, P[F[t].a])) < eps && fabs(volume(a, b, c,
           P[F[t].b])) < eps && fabs(volume(a, b, c, P[F[t].c])) < eps;
  }
  void construct() {
    cnt = 0;
    if (n < 4)
      return;
    bool sb = 1;
    for (int i = 1; i < n; i++) {
      if (vlen(P[0] - P[i]) > eps) {
        swap(P[1], P[i]);
        sb = 0;
        break;
      }
    }
    if (sb)return;
    sb = 1;
    for (int i = 2; i < n; i++) {
      if (vlen((P[0] - P[1]) * (P[1] - P[i])) > eps) {
        swap(P[2], P[i]);
        sb = 0;
        break;
      }
    }
    if (sb)return;
    sb = 1;
    for (int i = 3; i < n; i++) {
      if (fabs((P[0] - P[1]) * (P[1] - P[2]) ^ (P[0] - P[i])) > eps) {
        swap(P[3], P[i]);
        sb = 0;
        break;
      }
    }
    if (sb)return;
    fac add;
    for (int i = 0; i < 4; i++) {
      add.a = (i+1)%4, add.b = (i+2)%4, add.c = (i+3)%4, add.ok = 1;
      if (ptof(P[i], add) > 0)
        swap(add.b, add.c);
      to[add.a][add.b] = to[add.b][add.c] = to[add.c][add.a] = cnt;
      F[cnt++] = add;
    }
    for (int i = 4; i < n; i++) {
      for (int j = 0; j < cnt; j++) {
        if (F[j].ok && ptof(P[i], F[j]) > eps) {
          dfs(i, j);
          break;
        }
      }
    }
    int tmp = cnt;
    cnt = 0;
    for (int i = 0; i < tmp; i++) {
      if (F[i].ok) {
        F[cnt++] = F[i];
      }
    }
  }
  double area() {
    double ret = 0.0;
    for (int i = 0; i < cnt; i++) {
      ret += area(P[F[i].a], P[F[i].b], P[F[i].c]);
    }
    return ret / 2.0;
  }
  double volume() {
    pt O(0, 0, 0);
    double ret = 0.0;
    for (int i = 0; i < cnt; i++) {
      ret += volume(O, P[F[i].a], P[F[i].b], P[F[i].c]);
    }
    return fabs(ret / 6.0);
  }
  int facetCnt_tri() {
    return cnt;
  }
  int facetCnt() {
    int ans = 0;
    for (int i = 0; i < cnt; i++) {
      bool nb = 1;
      for (int j = 0; j < i; j++) {
        if (same(i, j)) {
          nb = 0;
          break;
        }
      }
      ans += nb;
    }
    return ans;
  }
  pt Fc[MAXV*8];
  double V[MAXV*8];
  pt Center() {
    pt O(0,0,0);
    for (int i = 0; i < cnt; i++) {
      Fc[i].x = (O.x+P[F[i].a].x+P[F[i].b].x+P[F[i].c].x)/4.0;
      Fc[i].y = (O.y+P[F[i].a].y+P[F[i].b].y+P[F[i].c].y)/4.0;
      Fc[i].z = (O.z+P[F[i].a].z+P[F[i].b].z+P[F[i].c].z)/4.0;
      V[i] = volume(O,P[F[i].a],P[F[i].b],P[F[i].c]);
    }
    pt res = Fc[0],tmp;
    double m = V[0];
    for (int i = 1; i < cnt; i++) {
      if (fabs(m+V[i]) < eps)
        V[i] += eps;
      tmp.x = (m*res.x+V[i]*Fc[i].x)/(m+V[i]);
      tmp.y = (m*res.y+V[i]*Fc[i].y)/(m+V[i]);
      tmp.z = (m*res.z+V[i]*Fc[i].z)/(m+V[i]);
      m += V[i];
      res = tmp;
    }
    return res;
  }
};
```

3.3 Circle

3.3.1 Functions

```
//Common area of two circle
double area(int x1,int y1,int x2,int y2,double r1,double r2) {
  double s=dis(x2-x1,y2-y1);
  if(r1+r2<s) return 0;
  else if(r2-r1>s) return PI*r1*r1;
  else if(r1-r2>s) return PI*r2*r2;
  double q1=acos((r1*r1+s*s-r2*r2)/(2*r1*s));
  double q2=acos((r2*r2+s*s-r1*r1)/(2*r2*s));
  return (r1*r1*q1+r2*r2*q2-r1*s*sin(q1));
}
```

3.3.2 Union

```
for (int i = 1; i <= n; i++)
  ans[i] = 0.0;
for (int i = 0; i < n; i++) {
  tote = 0;
  e[tote++] = Event(-pi,1);
  e[tote++] = Event(pi,-1);
  for (int j = 0; j < n; j++)
    if (j != i) {
      lab = Point(c[j].c.x-c[i].c.x,c[j].c.y-c[i].c.y);
      AB = lab.Length();
      AC = c[i].r;
      BC = c[j].r;
      if (cmp(AB+AC,BC) <= 0) {
        e[tote++] = Event(-pi,1);
        e[tote++] = Event(pi,-1);
        continue;
      }
      if (cmp(AB+BC,AC) <= 0) continue;
      if (cmp(AB,AC+BC) > 0)  continue;
      theta = atan2(lab.y,lab.x);
      fai = acos((AC*AC+AB*AB-BC*BC)/(2.0*AC*AB));
      a0 = theta-fai;
      if (cmp(a0,-pi) < 0)  a0 += 2*pi;
      a1 = theta+fai;
      if (cmp(a1,pi) > 0)  a1 -= 2*pi;
      if (cmp(a0,a1) > 0) {
        e[tote++] = Event(a0,1);
        e[tote++] = Event(pi,-1);
        e[tote++] = Event(-pi,1);
        e[tote++] = Event(a1,-1);
      } else {
        e[tote++] = Event(a0,1);
        e[tote++] = Event(a1,-1);
      }
    }
  sort(e,e+tote,Eventcmp);
  cur = 0;
  for (int j = 0; j < tote; j++) {
    if (cur != 0 && cmp(e[j].tim,pre[cur]) != 0) {
      ans[cur] += Area(e[j].tim-pre[cur],c[i].r);
      ans[cur] += xmult(Point(c[i].c.x+c[i].r*cos(pre[cur]),c[i].c.y+c[i].r*sin(pre[cur])),
                        Point(c[i].c.x+c[i].r*cos(e[j].tim),c[i].c.y+c[i].r*sin(e[j].tim)))/2.0;
    }
    cur += e[j].typ;
    pre[cur] = e[j].tim;
  }
}
for (int i = 1; i < n; i++)
  ans[i] -= ans[i+1];
```

3.3.3 Area of intersection part with polygon

```
bool InCircle(Point a,double r) {
  return cmp(a.x*a.x+a.y*a.y,r*r) <= 0;
  //ϵ should big enough
}
double CalcArea(Point a,Point b,double r) {
  Point p[4];
  int tot = 0;
  p[tot++] = a;
  Point tv = Point(a,b);
  Line tmp = Line(Point(0,0),Point(tv.y,-tv.x));
  Point near = LineToLine(Line(a,b),tmp);
  if (cmp(near.x*near.x+near.y*near.y,r*r) <= 0) {
    double A,B,C;
    A = near.x*near.x+near.y*near.y;
    C = r;
    B = C*C-A;
    double tvl = tv.x*tv.x+tv.y*tv.y;
    double tmp = sqrt(B/tvl);
    p[tot] = Point(near.x+tmp*tv.x,near.y+tmp*tv.y);
    if (OnSeg(Line(a,b),p[tot]) == true)  tot++;
    p[tot] = Point(near.x-tmp*tv.x,near.y-tmp*tv.y);
    if (OnSeg(Line(a,b),p[tot]) == true)  tot++;
  }
  if (tot == 3) {
    if (cmp(Point(p[0],p[1]).Length(),Point(p[0],p[2]).Length()) > 0)
      swap(p[1],p[2]);
  }
  p[tot++] = b;
  double res = 0.0,theta,a0,a1,sgn;
  for (int i = 0; i < tot-1; i++) {
    if (InCircle(p[i],r) == true && InCircle(p[i+1],r) == true) {
      res += 0.5*xmult(p[i],p[i+1]);
    } else {
      a0 = atan2(p[i+1].y,p[i+1].x);
      a1 = atan2(p[i].y,p[i].x);
      if (a0 < a1)  a0 += 2*pi;
      theta = a0-a1;
      if (cmp(theta,pi) >= 0) theta = 2*pi-theta;
      sgn = xmult(p[i],p[i+1])/2.0;
      if (cmp(sgn,0) < 0) theta = -theta;
      res += 0.5*r*r*theta;
    }
  }
  return res;
}
area2 = 0.0;
for (int i = 0; i < resn; i++) //counterclockwise
  area2 += CalcArea(p[i],p[(i+1)%resn],r);
```

3.4 Matrix

3.4.1 基本矩阵

按向量 $\overrightarrow{(x,y,z)}$ 平移：

$$\begin{pmatrix} 1 & 0 & 0 & x \\ 0 & 1 & 0 & y \\ 0 & 0 & 1 & z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

按比例 (x,y,z) 缩放：

$$\begin{pmatrix} x & 0 & 0 & 0 \\ 0 & y & 0 & 0 \\ 0 & 0 & z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

绕单位向量 $\overrightarrow{(x,y,z)}$ 旋转 $angle$ 角度：

$$\begin{pmatrix} x^2\times(1-c)+c & x\times y\times(1-c)-z\times s & x\times z\times(1-c)+y\times s & 0 \\ y\times x\times(1-c)+z\times s & y^2\times(1-c)+c & y\times z\times(1-c)-x\times s & 0 \\ x\times z\times(1-c)-y\times s & y\times z\times(1-c)+x\times s & z^2\times(1-c)+c & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{cases} s=sin(angle) \\ c=cos(angle) \end{cases}$$

以上矩阵变换都把点当作列向量，旋转角度的正负由右手定则决定

4 Graph

4.1 Sap

```
const int MAXEDGE=50000;
const int MAXN=3000;
const int inf=0x3fffffff;
struct edges {
  int cap,to,next,flow;
} edge[MAXEDGE+100];
struct nodes {
  int head,label,pre,cur;
} node[MAXN+100];
int L,N;
int gap[MAXN+100];
void init(int n) {
  L=0;
  N=n;
  for (int i=0; i<N; i++)
    node[i].head=-1;
}
void add_edge(int x,int y,int z,int w) {
  edge[L].cap=z;
  edge[L].flow=0;
  edge[L].to=y;
  edge[L].next=node[x].head;
  node[x].head=L++;
  edge[L].cap=w;
  edge[L].flow=0;
  edge[L].to=x;
  edge[L].next=node[y].head;
  node[y].head=L++;
}
int maxflow(int s,int t) {
  memset(gap,0,sizeof(gap));
  gap[0]=N;
  int u,ans=0;
  for (int i=0; i<N; i++) {
    node[i].cur=node[i].head;
    node[i].label=0;
  }
  u=s;
  node[u].pre=-1;
  while (node[s].label<N) {
    if (u==t) {
      int min=inf;
      for (int i=node[u].pre; i!=-1; i=node[edge[i^1].to].pre)
        if (min>edge[i].cap-edge[i].flow)
          min=edge[i].cap-edge[i].flow;
      for (int i=node[u].pre; i!=-1; i=node[edge[i^1].to].pre) {
        edge[i].flow+=min;
        edge[i^1].flow-=min;
      }
      u=s;
      ans+=min;
      continue;
    }
    bool flag=false;
    int v;
    for (int i=node[u].cur; i!=-1; i=edge[i].next) {
      v=edge[i].to;
      if (edge[i].cap-edge[i].flow &&
          node[v].label+1==node[u].label) {
```

```
        flag=true;
        node[u].cur=node[v].pre=i;
        break;
      }
    }
    if (flag) {
      u=v;
      continue;
    }
    node[u].cur=node[u].head;
    int min=N;
    for (int i=node[u].head; i!=-1; i=edge[i].next)
      if (edge[i].cap-edge[i].flow && node[edge[i].to].label<min)
        min=node[edge[i].to].label;
    gap[node[u].label]--;
    if (!gap[node[u].label]) return ans;
    node[u].label=min+1;
    gap[node[u].label]++;
    if (u!=s) u=edge[node[u].pre^1].to;
  }
  return ans;
}
```

4.2 Minimal cost maximal flow

```
//Use stack instead of queue when get TLE
int L,N;
int K;
struct edges {
  int to,next,cap,flow,cost;
} edge[MAXM];
struct nodes {
  int dis,pre,head;
  bool visit;
} node[MAXN];
void init(int n) {
  N=n;
  L=0;
  for (int i=0; i<N; i++)
    node[i].head=-1;
}
void add_edge(int x,int y,int cap,int cost) {
  edge[L].to=y;
  edge[L].cap=cap;
  edge[L].cost=cost;
  edge[L].flow=0;
  edge[L].next=node[x].head;
  node[x].head=L++;
  edge[L].to=x;
  edge[L].cap=0;
  edge[L].cost=-cost;
  edge[L].flow=0;
  edge[L].next=node[y].head;
  node[y].head=L++;
}
bool spfa(int s,int t) {
  queue <int> q;
  for (int i=0; i<N; i++) {
    node[i].dis=0x3fffffff;
    node[i].pre=-1;
    node[i].visit=0;
  }
  node[s].dis=0;
  node[s].visit=1;
  q.push(s);
  while (!q.empty()) {
    int u=q.front();
    node[u].visit=0;
    for (int i=node[u].head; i!=-1; i=edge[i].next) {
      int v=edge[i].to;
      if (edge[i].cap>edge[i].flow &&
          node[v].dis>node[u].dis+edge[i].cost) {
        node[v].dis=node[u].dis+edge[i].cost;
        node[v].pre=i;
        if (!node[v].visit) {
          node[v].visit=1;
          q.push(v);
        }
      }
    }
    q.pop();
  }
  if (node[t].pre==-1)
    return 0;
  else
    return 1;
}
int mcmf(int s,int t,int &cost) {
  int flow=0;
  while (spfa(s,t)) {
    int max=inf;
    for (int i=node[t].pre; i!=-1; i=node[edge[i^1].to].pre) {
      if (max>edge[i].cap-edge[i].flow)
        max=edge[i].cap-edge[i].flow;
    }
    for (int i=node[t].pre; i!=-1; i=node[edge[i^1].to].pre) {
      edge[i].flow+=max;
      edge[i^1].flow-=max;
      cost+=edge[i].cost*max;
    }
    flow+=max;
  }
  return flow;
}
```

4.3 Johnson Minimal cost flow

```
#include <cstdio>
#include <cstring>
#include <algorithm>
#include <queue>
#include <stack>
using namespace std;
const int MAXN = 2003;
const int MAXM = 2000 * 1999 / 2 + 2000 * 3;
int N, L;
int head[MAXN];
struct Edge {
  int to, next, flow, cost;
} edge[MAXM * 2];
int h[MAXN], dis[MAXN], pre[MAXN];
struct Heap {
  int value[MAXN + 1], id[MAXN + 1];
  int pos[MAXN];
  int size;
  void init() {
    size = 1;
  }
  void swap2(int p, int q) {
    swap(value[p], value[q]);
    swap(id[p], id[q]);
    pos[id[p]] = p;
    pos[id[q]] = q;
  }
  void push_up(int p) {
    while (p > 1 && value[p / 2] > value[p]) {
      swap2(p, p / 2);
      p /= 2;
    }
  }
  void push_down(int p) {
    while (p * 2 < size) {
      int best = p;
      if (p * 2 < size && value[p] > value[p * 2])
        best = p * 2;
      if (p * 2 + 1 < size && value[best] > value[p * 2 + 1])
        best = p * 2 + 1;
      if (p == best)
        break;
      swap2(p, best);
      p = best;
    }
  }
  void push(int _value, int _id) {
    value[size] = _value;
    id[size] = _id;
    pos[_id] = size;
    push_up(size++);
  }
  int top() {
    return id[1];
  }
  void pop() {
    value[1] = value[size - 1];
    id[1] = id[--size];
    pos[id[1]] = 1;
    push_down(1);
  }
  void update(int _value, int _id) {
    int p = pos[_id];
    value[p] = _value;
    push_up(p);
  }
} heap;
bool inque[MAXN];
void init(int n) {
  N = n;
  L = 0;
  memset(head, -1, 4 * n);
}
void add_edge(int u, int v, int flow, int cost) {
  edge[L].to = v;
  edge[L].flow = flow;
  edge[L].cost = cost;
  edge[L].next = head[u];
  head[u] = L++;
  edge[L].to = u;
  edge[L].flow = 0;
  edge[L].cost = -cost;
  edge[L].next = head[v];
  head[v] = L++;
}
void spfa(int s) {
  memset(dis, 63, 4 * N);
  memset(inque, 0, N);
  memset(pre, -1, 4 * N);
  dis[s] = 0;
  queue <int> que;
  que.push(s);
  while (!que.empty()) {
    int u = que.front();
    inque[u] = 0;
    que.pop();
    for (int i = head[u]; i != -1; i = edge[i].next)
      if (edge[i].flow) {
        int v = edge[i].to;
        if (dis[v] > dis[u] + edge[i].cost) {
          dis[v] = dis[u] + edge[i].cost;
          pre[v] = i;
          if (!inque[v]) {
            inque[v] = 1;
            que.push(v);
          }
        }
      }
  }
}
void dijkstra(int s) {
  for (int i = 0; i < N; ++i)
    h[i] += dis[i];
  memset(dis, 63, 4 * N);
  memset(pre, -1, 4 * N);
  memset(inque, 0, N);
  dis[s] = 0;
  inque[s] = 1;
  heap.init();
  heap.push(0, s);
  while (heap.size > 1) {
    int u = heap.top();
    heap.pop();
    for (int i = head[u]; i != -1; i = edge[i].next)
      if (edge[i].flow) {
        int v = edge[i].to;
        if (dis[v] > dis[u] + edge[i].cost + h[u] - h[v]) {
          dis[v] = dis[u] + edge[i].cost + h[u] - h[v];
          pre[v] = i;
          if (!inque[v]) {
            heap.push(dis[v], v);
            inque[v] = 1;
          } else
            heap.update(dis[v], v);
        }
      }
  }
}
int minimumCostFlow(int s, int t, int &cost) {
  int flow = 0;
  memset(h, 0, 4 * N);
  for (spfa(s); pre[t] != -1; dijkstra(s)) {
    int maxs = edge[pre[t]].flow;
    for (int i = pre[t]; i != -1; i = pre[edge[i ^ 1].to])
      maxs = min(maxs, edge[i].flow);
    for (int i = pre[t]; i != -1; i = pre[edge[i ^ 1].to]) {
      edge[i].flow -= maxs;
      edge[i ^ 1].flow += maxs;
      cost += edge[i].cost * maxs;
    }
    flow += maxs;
  }
  return flow;
}
int main() {
  return 0;
}
```

4.4 Bi-connect

```
struct edges {
  int to,next;
  bool cut,visit;
} edge[MAXM<<1];
int head[MAXN],low[MAXN],dpt[MAXN],L;
bool visit[MAXN],cut[MAXN];
void init(int n) {
  L=0;
  memset(head,-1,4*n);
  memset(visit,0,n);
}
void add_edge(int u,int v) {
  edge[L].cut=edge[L].visit=0;
  edge[L].to=v;
  edge[L].next=head[u];
  head[u]=L++;
}
int idx;
stack<int> st;
int bcc[MAXM];
void dfs(int u,int fu,int deg) {
  cut[u]=0;
  visit[u]=1;
  low[u]=dpt[u]=deg;
  int tot=0;
  for (int i=head[u]; i!=-1; i=edge[i].next) {
    int v=edge[i].to;
    if (edge[i].visit)
      continue;
    st.push(i/2);
    edge[i].visit=edge[i^1].visit=1;
    if (visit[v]) {
      low[u]=dpt[v]>low[u]?low[u]:dpt[v];
      continue;
    }
    dfs(v,u,deg+1);
    edge[i].cut=edge[i^1].cut=(low[v]>dpt[u] || edge[i].cut);
    if (u!=fu) cut[u]=low[v]>=dpt[u]?1:cut[u];
    if (low[v]>=dpt[u] || u==fu) {
      while (st.top()!=i/2) {
        int x=st.top()*2,y=st.top()*2+1;
        bcc[st.top()]=idx;
        st.pop();
      }
      bcc[i/2]=idx++;
      st.pop();
    }
    low[u]=low[v]>low[u]?low[u]:low[v];
    tot++;
  }
  if (u==fu && tot>1) cut[u]=1;
}
int main() {
  int n,m;
  while (scanf("%d%d",&n,&m)!=EOF) {
    init(n);
    for (int i=0; i<m; i++) {
      int u,v;
      scanf("%d%d",&u,&v);
      add_edge(u,v);
      add_edge(v,u);
    }
    idx=0;
    for (int i=0; i<n; i++)
      if (!visit[i])
        dfs(i,i,0);
  }
  return 0;
}
```

4.5 Cut and bridge

```
struct edges {
  int to,next;
  bool cut,visit;
  int from;
} edge[MAXN-1<<1];
int head[MAXN],low[MAXN],dfn[MAXN],L;
bool visit[MAXN],cut[MAXN];
void init(int n) {
  L=0;
  memset(head,-1,4*n);
  memset(cut,0,4*n);
  memset(visit,0,4*n);
}
void add_edge(int u,int v) {
  edge[L].from=u;
  edge[L].cut=edge[L].visit=0;
  edge[L].to=v;
  edge[L].next=head[u];
  head[u]=L++;
}
int idx;
void dfs(int u,int fu) {
  visit[u]=1;
  low[u]=dfn[u]=idx++;
  int tot=0;
  for (int i=head[u]; i!=-1; i=edge[i].next) {
    int v=edge[i].to;
    if (edge[i].visit)
      continue;
    edge[i].visit=edge[i^1].visit=1;
    if (visit[v]) {
      low[u]=dfn[v]>low[u]?low[u]:dfn[v];
      continue;
    }
    dfs(v,u);
    edge[i].cut=edge[i^1].cut=low[v]>dfn[u] || edge[i].cut;
    if (u!=fu) cut[u]=low[v]>=dfn[u]?1:cut[u];
    low[u]=low[v]>low[u]?low[u]:low[v];
    tot++;
  }
  if (u==fu && tot>1) cut[u]=1;
}
int main() {
  int t;
  scanf("%d",&t);
  while (t--) {
    int n,m;
    scanf("%d%d",&n,&m);
    init(n);
    for (int i=0; i<m; i++) {
      int u,v;
      scanf("%d%d",&u,&v);
      add_edge(--u,--v);
      add_edge(v,u);
    }
    for (int i=0; i<n; i++)
      if (!visit[i]) {
        idx=0;
        dfs(i,i);
      }
  }
  return 0;
}
```

4.6 Stoer-Wagner

```
int map[maxn][maxn];
int n;
void contract(int x,int y) {
  int i,j;
  for (i=0; i<n; i++)
    if (i!=x) map[x][i]+=map[y][i],map[i][x]+=map[i][y];
  for (i=y+1; i<n; i++) for (j=0; j<n; j++) {
      map[i-1][j]=map[i][j];
      map[j][i-1]=map[j][i];
    }
  n--;
}
int w[maxn],c[maxn];
int sx,tx;
int mincut() {
  int i,j,k,t;
  memset(c,0,sizeof(c));
  c[0]=1;
  for (i=0; i<n; i++) w[i]=map[0][i];
  for (i=1; i+1<n; i++) {
    t=k=-1;
    for (j=0; j<n; j++) if (c[j]==0&&w[j]>k)
        k=w[t=j];
    c[sx=t]=1;
    for (j=0; j<n; j++) w[j]+=map[t][j];
  }
  for (i=0; i<n; i++) if (c[i]==0) return w[tx=i];
}
int main() {
  int i,j,k,m;
  while (scanf("%d%d",&n,&m)!=EOF) {
    memset(map,0,sizeof(map));
    while (m--) {
      scanf("%d%d%d",&i,&j,&k);
      map[i][j]+=k;
      map[j][i]+=k;
    }
    int mint=999999999;
    while (n>1) {
      k=mincut();
      if (k<mint) mint=k;
      contract(sx,tx);
    }
    printf("%d\n",mint);
  }
  return 0;
}
```

4.7 Euler path

```
//Directed graph
void solve(int x) {
  int i;
  if (!match[x]) {
    path[++l]=x;
    return ;
  }
  for (i=1; i<=n; i++)
    if (b[x][i]) {
      b[x][i]--;
      match[x]--;
      solve(i);
    }
  path[++l]=x;
}
//Undirected graph
void solve(int x) {
  int i;
  if (!match[x]) {
    path[++l]=x;
    return ;
  }
  for (i=1; i<=n; i++)
    if (b[x][i]) {
      b[x][i]--;
      b[i][x]--;
      match[x]--;
      match[i]--;
      solve(i);
    }
  path[++l]=x;
}
```

4.8 Strongly connected component

```
int dfsnum[2000];
int low[2000];
int stack[2000];
int top;
int ans;
int an;
int be[2000];
int flag[2000];
void dfs(int x) {
  dfsnum[x] = low[x] = ans++;
  stack[++top] = x;
  flag[x] = 1;
  for (int i = head[x]; i != -1; i = edge[i].next) {
    int y = edge[i].to;
```

```
      if (dfsnum[y] == -1) {
        dfs(y);
        low[x] = min(low[x],low[y]);
      } else if (flag[y] == 1)
        low[x] = min(low[x],dfsnum[y]);
    }
    if (dfsnum[x] == low[x]) {
      while (stack[top] != x) {
        flag[stack[top]] = 0;
        be[stack[top]] = an;
        top--;
      }
      flag[x] = 0;
      be[x] = an++;
      top--;
    }
}
void SC() {
  memset(dfsnum,-1,sizeof(dfsnum));
  memset(flag,0,sizeof(flag));
  top = 0;
  an = 0;
  ans = 0;
  for (int i = 0; i < n; i++)
    if (dfsnum[i] == -1)
      dfs(i);
}
```

4.9 Match

4.9.1 Bipartite graph

```
bool check(int u) {
  for (int i=head[u]; i!=-1; i=edge[i].next) {
    int v=edge[i].to;
    if (!use[v]) {
      use[v]=1;
      if (pre[v]==-1 || check(pre[v])) {
        pre[v]=u;
        return 1;
      }
    }
  }
  return 0;
}
int match() {
  int ret=0;
  memset(pre,-1,sizeof(pre));
  for (int u=1; u<=N; u++) {
    memset(use,0,sizeof(use));
    if (check(u))
      ret++;
  }
  return ret;
}
```

4.9.2 Edmonds

```
int N;
bool Graph[MaxN+1][MaxN+1];
int Match[MaxN+1];
bool InQueue[MaxN+1],InPath[MaxN+1],InBlossom[MaxN+1];
```

```
int Head,Tail;
int Queue[MaxN+1];
int Start,Finish;
int NewBase;
int Father[MaxN+1],Base[MaxN+1];
int Count;
void CreateGraph() {}
void Push(int u) {
  Queue[Tail] = u;
  Tail++;
  InQueue[u] = true;
}
int Pop() {
  int res = Queue[Head];
  Head++;
  return res;
}
int FindCommonAncestor(int u,int v) {
  memset(InPath,false,sizeof(InPath));
  while (true) {
    u = Base[u];
    InPath[u] = true;
    if (u == Start) break;
    u = Father[Match[u]];
  }
  while (true) {
    v = Base[v];
    if (InPath[v]) break;
    v = Father[Match[v]];
  }
  return v;
}
void ResetTrace(int u) {
  int v;
  while (Base[u] != NewBase) {
    v = Match[u];
    InBlossom[Base[u]] = InBlossom[Base[v]] = true;
    u = Father[v];
    if (Base[u] != NewBase) Father[u] = v;
  }
}
void BlossomContract(int u,int v) {
  NewBase = FindCommonAncestor(u,v);
  memset(InBlossom,false,sizeof(InBlossom));
  ResetTrace(u);
  ResetTrace(v);
  if (Base[u] != NewBase) Father[u] = v;
  if (Base[v] != NewBase) Father[v] = u;
  for (int tu = 1; tu <= N; tu++)
    if (InBlossom[Base[tu]]) {
      Base[tu] = NewBase;
      if (!InQueue[tu]) Push(tu);
    }
}
void FindAugmentingPath() {
  memset(InQueue,false,sizeof(InQueue));
  memset(Father,0,sizeof(Father));
  for (int i = 1; i <= N; i++)
    Base[i] = i;
  Head = Tail = 1;
  Push(Start);
  Finish = 0;
  while (Head < Tail) {
    int u = Pop();
    for (int v = 1; v <= N; v++)
      if (Graph[u][v] && (Base[u] != Base[v]) && (Match[u] != v)) {
        if ((v == Start) ||
            ((Match[v] > 0) && (Father[Match[v]] > 0)))
          BlossomContract(u,v);
        else if (Father[v] == 0) {
          Father[v] = u;
          if (Match[v] > 0)
            Push(Match[v]);
          else {
            Finish = v;
            return;
          }
        }
      }
  }
}
void AugmentPath() {
  int u,v,w;
  u = Finish;
  while (u > 0) {
    v = Father[u];
    w = Match[v];
    Match[v] = u;
    Match[u] = v;
    u = w;
  }
}
void Edmonds() {
  memset(Match,0,sizeof(Match));
  for (int u = 1; u <= N; u++)
    if (Match[u] == 0) {
      Start = u;
      FindAugmentingPath();
      if (Finish > 0) AugmentPath();
    }
}
void PrintMatch() {}
int main() {
  CreateGraph();
  Edmonds();
  PrintMatch();
}
```

4.9.3 KM

```
bool visx[N],visy[N];
int lx[N],ly[N];
int matchy[N];
int map[N][N];
bool find(int x) {
  visx[x]=true;
  int t;
  for (int y=0; y<ycnt; y++) {
    if (!visy[y]) {
      t=lx[x]+ly[y]-map[x][y];
      if (t==0) {
        visy[y]=true;
        if (matchy[y]==-1 || find(matchy[y])) {
          matchy[y]=x;
          return true;
        }
      } else if (lack>t) lack=t;
    }
  }
  return false;
}
void KM() {
  memset(lx,0,sizeof(lx));
  memset(ly,0,sizeof(ly));
  memset(matchy,-1,sizeof(matchy));
  for (int i=0; i<xcnt; i++)
    for (int j=0; j<ycnt; j++)
      if (map[i][j]>lx[i])
        lx[i]=map[i][j];
  for (int x=0; x<xcnt; x++) {
    while (true) {
      memset(visx,false,sizeof(visx));
      memset(visy,false,sizeof(visy));
      lack=INFI;
      if (find(x)) break;
      for (int i=0; i<xcnt; i++) {
        if (visx[i]) lx[i]-=lack;
        if (visy[i]) ly[i]+=lack;
      }
    }
  }
  int cost=0;
  for (int i=0; i<ycnt; i++)
    cost+=map[matchy[i]][i];
}
```

4.10 Clique

```
bool am[100][100];
int ans;
int c[100];
int U[100][100];
int n;
bool dfs(int rest,int num) {
  if (!rest) {
    if (num>=ans)
      return 1;
    else
      return 0;
  }
  int pre=-1;
  for (int i=0; i<rest && rest-i+num>=ans; i++) {
    int idx=U[num][i];
    if (num+c[idx]<ans)
      return 0;
    int nrest=0;
    for (int j=i+1; j<rest; j++)
      if (am[idx][U[num][j]])
        U[num+1][nrest++]=U[num][j];
    if (dfs(nrest,num+1))
      return 1;
  }
  return 0;
}
int main() {
  while (scanf("%d",&n),n) {
    for (int i=0; i<n; i++)
      for (int j=0; j<n; j++)
        scanf("%d",&am[i][j]);
    ans=0;
    for (int i=n-1; i>=0; i--) {
      int rest=0;
      for (int j=i+1; j<n; j++)
        if (am[i][j])
          U[0][rest++]=j;
      ans+=dfs(rest,0);
      c[i]=ans;
    }
    printf("%d\n",ans);
  }
  return 0;
}
```

4.11 Spanning tree

4.11.1 Count the number of spanning tree

```
Matrix laplacian;
laplacian.clear();
for (int i = 0; i < n; i++)
  for (int j = 0; j < n; j++)
    if (i != j && G[i][j]) {
      laplacian.a[i][j] = -1;
      laplacian.a[i][i]++;
    }
printf("%d\n",laplacian.det(n-1));
```

4.11.2 Spanning tree on directed graph

```
struct Edge {
  int u,v,cost;
};
Edge e[1001*1001];
int pre[1001],id[1001],visit[1001],in[1001];
int zhuliu(int root,int n,int m,Edge e[]) {
  int res = 0,u,v;
  while (true) {
    for (int i = 0; i < n; i++)
      in[i] = inf;
    for (int i = 0; i < m; i++)
      if (e[i].u != e[i].v && e[i].cost < in[e[i].v]) {
        pre[e[i].v] = e[i].u;
        in[e[i].v] = e[i].cost;
      }
    for (int i = 0; i < n; i++)
      if (i != root)
        if (in[i] == inf)   return -1;
    int tn = 0;
    memset(id,-1,sizeof(id));
    memset(visit,-1,sizeof(visit));
    in[root] = 0;
    for (int i = 0; i < n; i++) {
      res += in[i];
      v = i;
      while (visit[v] != i && id[v] == -1 && v != root) {
        visit[v] = i;
        v = pre[v];
      }
      if(v != root && id[v] == -1) {
        for(int u = pre[v] ; u != v ; u = pre[u])
          id[u] = tn;
        id[v] = tn++;
      }
    }
    if(tn == 0) break;
    for (int i = 0; i < n; i++)
      if (id[i] == -1)
        id[i] = tn++;
    for (int i = 0; i < m;) {
      int v = e[i].v;
      e[i].u = id[e[i].u];
      e[i].v = id[e[i].v];
      if (e[i].u != e[i].v)
        e[i++].cost -= in[v];
      else
        swap(e[i],e[--m]);
    }
    n = tn;
    root = id[root];
  }
  return res;
}
```

4.12 Kth shortest path

```
#include<cstdio>
#include<cstring>
#include<queue>
using namespace std;
int K;
class states {
public:
  int cost,id;
};
int dist[1000];
class cmp {
public:
  bool operator ()(const states &i,const states &j) {
    return i.cost>j.cost;
  }
};
class cmp2 {
public:
  bool operator ()(const states &i,const states &j) {
    return i.cost+dist[i.id]>j.cost+dist[j.id];
  }
};
struct edges {
  int to,next,cost;
} edger[100000],edge[100000];
int headr[1000],head[1000],Lr,L;
void dijkstra(int s) {
  states u;
  u.id=s;
  u.cost=0;
  dist[s]=0;
  priority_queue<states,vector<states>,cmp> q;
  q.push(u);
  while (!q.empty()) {
    u=q.top();
    q.pop();
    if (u.cost!=dist[u.id]) continue;
    for (int i=headr[u.id]; i!=-1; i=edger[i].next) {
      states v=u;
      v.id=edger[i].to;
      if (dist[v.id]>dist[u.id]+edger[i].cost) {
        v.cost=dist[v.id]=dist[u.id]+edger[i].cost;
        q.push(v);
      }
    }
  }
}
int num[1000];
void init(int n) {
  Lr=L=0;
  memset(head,-1,4*n);
  memset(headr,-1,4*n);
  memset(dist,63,4*n);
  memset(num,0,4*n);
}
void add_edge(int u,int v,int x) {
  edge[L].to=v;
  edge[L].cost=x;
  edge[L].next=head[u];
  head[u]=L++;
  edger[Lr].to=u;
  edger[Lr].cost=x;
  edger[Lr].next=headr[v];
  headr[v]=Lr++;
}
int a_star(int s,int t) {
  if (dist[s]==0x3f3f3f3f)
    return -1;
  priority_queue<states,vector<states>,cmp2> q;
  states tmp;
  tmp.id=s;
  tmp.cost=0;
  q.push(tmp);
  while (!q.empty()) {
    states u=q.top();
    q.pop();
    num[u.id]++;
    if (num[t]==K)
      return u.cost;
    for (int i=head[u.id]; i!=-1; i=edge[i].next) {
      int v=edge[i].to;
      tmp.id=v;
      tmp.cost=u.cost+edge[i].cost;
      q.push(tmp);
    }
  }
  return -1;
}
int main() {
  int n,m;
  scanf("%d%d",&n,&m);
  init(n);
  for (int i=0; i<m; i++) {
    int u,v,x;
    scanf("%d%d%d",&u,&v,&x);
    add_edge(u-1,v-1,x);
  }
  int s,t;
  scanf("%d%d%d",&s,&t,&K);
  if (s==t)
    K++;
  dijkstra(t-1);
  printf("%d\n",a_star(s-1,t-1));
}
```

4.13 Stable marriage problem

假定有 n 个男生和 个女生，理想的拍拖状态就是对于每对情侣 (a, b)，找不到另一对情侣 (c, d) 使得 c 更喜欢 b,b 也更喜欢 c，同理，对 a 来说也没有 (e, f) 使得 a 更喜欢 e 而 e 更喜欢 a, 当然最后会有一些人落单。这样子一个状态可以称为理想拍拖状态，它也有一个专业的名词叫稳定婚姻。求解这个问题可以用一个专有的算法，延迟认可算法，其核心就是让每个男生按自己喜欢的顺序逐个向女生表白，例如 leokan 向一个女生求爱，这个过程中，若这个女生没有男朋友，那么这个女生就暂时成为 leokan 的女朋友，或这个女生喜欢她现有男朋友的程度没有喜欢 leokan 高，这个女生也暂时成为 leokan 的女朋友，而她原有的男朋友则再将就找下一个次喜欢的女生来当女朋友。

```c
#include<string.h>
#include<stdio.h>
#define N 1050
int boy[N][N];
int girl[N][N];
int ans[N];
int cur[N];
int n;
void getMarry(int g) {
  for (int i=ans[g]+1; i<n; i++) {
    int b=girl[g][i]-1;
    if (cur[b]<0) {
      ans[g]=i;
      cur[b]=g;
      return;
    }
    int og=cur[b];
    if (boy[b][og] > boy[b][g]) {
      cur[b]=g;
      ans[g]=i;
      getMarry(og);
      return;
    }
  }
};
int main() {
  int t,a;
  scanf("%d",&t);
  while(t--) {
    memset(girl,0,sizeof(girl));
    memset(boy,0,sizeof(boy));
    scanf("%d",&n);
    for (int i=0; i<n; i++)
      for (int j=0; j<n; j++)
        scanf("%d",&girl[i][j]);
    for (int i=0; i<n; i++)
      for (int j=0; j<n; j++) {
        scanf("%d",&a);
        boy[i][a-1]=j;
      }
    memset(cur,0xff,sizeof(cur));
    memset(ans,0xff,sizeof(ans));
    for (int i=0; i<n; i++)
      getMarry(i);
    for (int i=0; i<n; i++)
      printf("%d\n",girl[i][ans[i]]);
  }
  return 0;
}
```

5 Math

5.1 FFT

5.1.1 Bit operation

$tf(X1, X2) = (tf(X1) - tf(X2), tf(X1) + tf(X2))$
异或:$tf(X1, X2) = (tf(X1) - tf(X2), tf(X1) + tf(X2))$
与:$tf(x1, x2) = (tf(x1) + tf(x2), tf(x1))$

```
// Transforms the interval [x, y) in a.
void transform(int x, int y) {
  if ( x == y - 1) {
    return;
  }
  int l2 = ( y - x ) / 2;
  int z = x + l2;
  transform(x, z);
  transform(z, y);
  for (int i=x; i<z; i++) {
    int x1 = a[i];
    int x2 = a[i+l2];
    a[i] = (x1 - x2 + MOD) % MOD;
    a[i+l2] = (x1 + x2) % MOD;
  }
}
// Reverses the transform in
// the interval [x, y) in a.
void untransform(int x, int y) {
  if ( x == y - 1) {
    return;
  }
  int l2 = ( y - x ) / 2;
  int z = x + l2;
  for (int i=x; i<z; i++) {
    long long y1 = a[i];
    long long y2 = a[i+l2];
    // x1 - x2 = y1
    // x1 + x2 = y2
    // 2 * x1  = y1 + y2
    // 2 * x2  = y2 - y1

    // In order to solve those equations, we need to divide by 2
    // But we are performing operations modulo 1000000007
    // that needs us to find the modular multiplicative inverse of 2.
    // That is saved in the INV2 variable.

    a[i] = (int)( ((y1 + y2)*INV2) % MOD );
    a[i+l2] = (int)( ((y2 - y1 + MOD)*INV2) % MOD );
  }
  untransform(x, z);
  untransform(z, y);
}
```

5.1.2 Standard

```
struct vir {
  long double re, im;
  vir(long double a = 0, long double b = 0) {
    re = a;
    im = b;
  }
  vir operator +(const vir& b) const {
    return vir(re + b.re, im + b.im);
  }
  vir operator -(const vir& b) const {
    return vir(re - b.re, im - b.im);
  }
  vir operator *(const vir& b) const {
    return vir(re * b.re - im * b.im, re * b.im + im * b.re);
  };
};
void change(vir *x, int len, int loglen) {
  int i, j, k, t;
  for (i = 0; i < len; i++) {
    t = i;
    for (j = k = 0; j < loglen; j++, t >>= 1)
      k = (k << 1) | (t & 1);
    if (k < i) {
      vir wt = x[k];
      x[k] = x[i];
      x[i] = wt;
    }
  }
}
void fft(vir *x, int len, int loglen) {
  int i, j, t, s, e;
  change(x, len, loglen);
  t = 1;
  for (i = 0; i < loglen; i++, t <<= 1) {
    s = 0;
    e = s + t;
    while (s < len) {
      vir a, b, wo(cos(PI / t), sin(PI / t)), wn(1, 0);
      for (j = s; j < s + t; j++) {
        a = x[j];
        b = x[j + t] * wn;
        x[j] = a + b;
        x[j + t] = a - b;
        wn = wn * wo;
      }
      s = e + t;
      e = s + t;
    }
  }
}
void dit_fft(vir *x, int len, int loglen) {
  int i, j, s, e, t = 1 << loglen;
  for (i = 0; i < loglen; i++) {
    t >>= 1;
    s = 0;
    e = s + t;
    while (s < len) {
      vir a, b, wn(1, 0), wo(cos(PI / t), -sin(PI / t));
      for (j = s; j < s + t; j++) {
        a = x[j] + x[j + t];
        b = (x[j] - x[j + t]) * wn;
        x[j] = a;
        x[j + t] = b;
        wn = wn * wo;
      }
      s = e + t;
      e = s + t;
    }
  }
  change(x, len, loglen);
  for (i = 0; i < len; i++)
```

```
    x[i].re /= len;
}
```

5.1.3 Usage

```
vir x1[MAXN], x2[MAXN];
void solve(long long *a, int lena, long long *b, int lenb, long long *ret, int& len) {
  int len1 = lena << 1;
  int len2 = lenb << 1;
  len = 1;
  int loglen = 0;
  while (len < len1 || len < len2) {
    len <<= 1;
    loglen++;
  }
  for (int i = 0; i < lena; i++)
    x1[i] = vir(a[i], 0);
  for (int i = lena; i < len; i++)
    x1[i] = vir(0, 0);
  for (int i = 0; i < lenb; i++)
    x2[i] = vir(b[i], 0);
  for (int i = lenb; i < len; i++)
    x2[i] = vir(0, 0);
  fft(x1, len, loglen);
  fft(x2, len, loglen);
  for (int i = 0; i < len; i++)
    x1[i] = x1[i] * x2[i];
  dit_fft(x1, len, loglen);
  for (int i = 0; i < len; i++)
    ret[i] = (long long)(x1[i].re + 0.5);
}
```

5.2 Euler function

```
int getEuler(int x) {
  getFactor(x);
  int ret=x;
  for (int i=0; i<N; i++)
    ret = ret/fac[i]*(fac[i]-1);
  return ret;
}
void getEuler2() {
  memset(euler,0,sizeof(euler));
  euler[1] = 1;
  for (int i = 2; i <= 3000000; i++) {
    if (!euler[i]) {
      for (int j = i; j <= 3000000; j += i) {
        if (!euler[j])
          euler[j] = j;
        euler[j] = euler[j]/i*(i-1);
      }
    }
  }
}
```

5.3 Ex-GCD

```
//Find one solution (x,y) of ax+by=gcd(a,b)
long long ex_gcd(long long a,long long b,long long &x,long long &y) {
  if (b) {
    long long ret = ex_gcd(b,a%b,x,y),tmp = x;
    x = y;
    y = tmp-(a/b)*y;
    return ret;
  } else {
    x = 1;
    y = 0;
    return a;
  }
}
```

5.4 Prime

5.4.1 Get primes

```
int N;
bool isPrime[10001];
int prime[10000];
void getPrime(int n) {
  memset(isPrime,1,++n);
  N=0;
  isPrime[0]=isPrime[1]=0;
  for (int i=2; i<n; i++) {
    if (isPrime[i])
      prime[N++]=i;
    for (int j=0; j<N && prime[j]*i<n; j++) {
      isPrime[i*prime[j]]=0;
      if (i%prime[j]==0)
        break;
    }
  }
}
```

5.4.2 Get factors

```
const int TIME = 8;
int factor[100],fac_top = -1;
//GCD of bint
bint gcd(bint small,bint big) {
  while(small) {
    swap(small,big);
    small%=big;
  }
  return abs(big);
}
//ret = (a*b)%n (n<2^62)
bint muti_mod(bint a,bint b,bint n) {
  bint exp = a%n, res = 0;
  while(b) {
    if(b&1) {
      res += exp;
      if(res>n) res -= n;
    }
    exp <<= 1;
    if (exp>n) exp -= n;
    b>>=1;
  }
  return res;
}
// ret = (a^b)%n
bint mod_exp(bint a,bint p,bint m) {
  bint exp=a%m, res=1;
  while(p>1) {
    if(p&1)
      res=muti_mod(res,exp,m);
    exp = muti_mod(exp,exp,m);
    p>>=1;
  }
  return muti_mod(res,exp,m);
}
//miller-rabin
bool miller_rabin(bint n, int times) {
  if(n==2)return 1;
  if(n<2||!(n&1))return 0;
  bint a, u=n-1, x, y;
  int t=0;
  while(u%2==0) {
    t++;
    u/=2;
  }
  srand(time(0));
  for(int i=0; i<times; i++) {
    a = rand() % (n-1) + 1;
    x = mod_exp(a, u, n);
    for(int j=0; j<t; j++) {
      y = muti_mod(x, x, n);
      if ( y == 1 && x != 1 && x != n-1 )
        return false; //must not
      x = y;
    }
    if( y!=1) return false;
  }
  return true;
}
bint pollard_rho(bint n,int c) {
  bint x,y,d,i = 1,k = 2;
  srand(time(0));
  x = rand()%(n-1)+1;
  y = x;
  while(true) {
    i++;
    x = (muti_mod(x,x,n) + c) % n;
    d = gcd(y-x, n);
    if (1 < d && d < n) return d;
    if( y == x) return n;
    if(i == k) {
      y = x;
      k <<= 1;
    }
  }
}
void findFactor(bint n,int k) {
  if(n==1)return;
  if(miller_rabin(n, TIME)) {
    factor[++fac_top] = n;
    return;
  }
  bint p = n;
  while(p >= n)
    p = pollard_rho(p,k--);
  findFactor(p,k);
  findFactor(n/p,k);
}
```

5.5 Simpson

```
double Simp(double l,double r) {
  double h = (r-l)/2.0;
  return h*(calc(l)+4*calc((l+r)/2.0)+calc(r))/3.0;
}
double rSimp(double l,double r) {
  double mid = (l+r)/2.0;
  if (abs((Simp(l,r)-Simp(l,mid)-Simp(mid,r)))/15 < eps)
    return Simp(l,r);
  else
    return rSimp(l,mid)+rSimp(mid,r);
}
```

5.6 Chinese remainder theorem

```
int m[10],a[10];//x mod m_i = a_i
bool solve(int &m0,int &a0,int m,int a) {
  int y,x;
  int g=ex_gcd(m0,m,x,y);
  if (abs(a-a0)%g) return 0;
  x*=(a-a0)/g;
  x%=m/g;
  a0=(x*m0+a0);
  m0*=m/g;
  a0%=m0;
  if (a0<0) a0+=m0;
  return 1;
}
int MLES() {
  bool flag=1;
  int m0=1,a0=0;
  for (int i=0; i<n; i++)
    if (!solve(m0,a0,m[i],a[i])) {
      flag=0;
      break;
    }
  if (flag)
    return a0;
  else
    return -1;
}
```

5.7 Lucas

```
//num[i] = i!
int comLucus(int n,int m,int p) {
  int ans=1;
  for (; n && m && ans; n/=p,m/=p) {
    if (n%p>=m%p)
      ans = ans*num[n%p]%p*getInv(num[m%p]%p)%p
            *getInv(num[n%p-m%p])%p;
    else
      ans=0;
  }
  return ans;
}
```

5.8 Primitive root

```
int getPriRoot(int p) {
  if (p==2) return 1;
  int phi = p - 1;
  getFactor(phi);
  for (int g = 2; g < p; ++g) {
    bool flag=1;
    for (int i = 0; flag && i < N; ++i)
      if (power(g, phi/fac[i], p) == 1)
        flag=0;
    if (flag)
      return g;
  }
}
```

5.9 Inverse element

```
void getInv2(int x) {
  inv[1]=1;
  for (int i=2; i<=x; i++)
    inv[i]=(mod-(mod/i)*inv[mod%i]%mod)%mod;
}
```

5.10 Calculator

注意灵活运用。
双目运算符在 calc() 中，左结合单目运算符在 P() 中，右结合单目运算符在 calc_exp 中。（但是还没遇到过。。）

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

char s[100000];
int n,cur;
const string OP = "+-*";

char next_char() {
  if (cur >= n) return EOF;
  return s[cur];
}

int get_priority(char ch) {
  if (ch == '*')  return 2;
  return 1;
}

int P();

int calc(int a,char op,int b) {
  if (op == '+')
    return a+b;
  if (op == '-')
    return a-b;
  if (op == '*')
    return a*b;
}

```

```
int calc_exp(int p) {
  int a = P();
  while ((OP.find(next_char()) != OP.npos) &&
         (get_priority(next_char()) >= p)) {
    char op = next_char();
    cur++;
    a = calc(a,op,calc_exp(get_priority(op)+1));
  }
  return a;
}

int totvar,m,var[26],varid[26];

int P() {
  if (next_char() == '-') {
    cur++;
    return -P();
  } else if (next_char() == '+') {
    cur++;
    return P();
  } else if (next_char() == '(') {
    cur++;
    int res = calc_exp(0);
    cur++;
    return res;
  } else {
    cur++;
    return var[varid[s[cur-1]-'a']];
  }
}

int id[26],minid;

int main() {
  while (true) {
    scanf("%d%d",&totvar,&var[0]);
    if (totvar == 0 && var[0] == 0)  break;
    for (int i = 1; i < totvar; i++)
      scanf("%d",&var[i]);
    scanf("%d",&m);
    scanf("%s",s);
    for (int i = 0; i < 26; i++)
      id[i] = -1;
    minid = 0;
    n = strlen(s);
    for (int i = 0; i < n; i++)
      if (s[i] >= 'a' && s[i] <= 'z') {
        if (id[s[i]-'a'] == -1) {
          id[s[i]-'a'] = minid;
          minid++;
        }
        s[i] = 'a'+id[s[i]-'a'];
      }
    for (int i = 0; i < totvar; i++)
      varid[i] = i;
    int res = 0;
    do {
      cur = 0;
      int tmp = calc_exp(0);
      if (tmp == m) {
        res++;
        break;
      }
    } while (next_permutation(varid,varid+totvar));
    //puts(s);
    if (res > 0)
      puts("YES");
    else
      puts("NO");
  }
  return 0;
}
```

5.11 Linear programming

```
#define MAXM 20  //max num of basic varibles
#define INF 1E200

double A[MAXM+5][MAXN+MAXM+5];
double b[MAXM+5],c[MAXN+MAXM+5];
int N[MAXN+5],B[MAXM+5];
double X[MAXN+MAXM+5],V;
int n,m,R,C,nCnt,bCnt;
int v1[MAXN],v2[MAXN];

int fcmp(double a,double b) {
  if(fabs(a-b)<1E-7) return 0;
  if(a>b) return 1;
  return -1;
}

void Pivot(int l,int e) {
  double t=A[l][e],p=c[e];
  b[l]=b[l]/t;
  for(int i=1; i<=C; i++)
    A[l][i]/=t;
  V=V-c[e]*b[l];
  for(int i=1; i<=R; i++) {
    if(i==l||fcmp(A[i][e],0.0)==0)
      continue;
    t=A[i][e];
    b[i]=b[i]-t*b[l];
    for(int j=1; j<=C; j++)
      A[i][j]=A[i][j]-t*A[l][j];
  }
  for(int i=1; i<=C; i++)
    c[i]=c[i]-p*A[l][i];
  for(int i=1; i<=nCnt; i++) {
    if(N[i]==e) {
      N[i]=B[l];
      break;
    }
  }
  B[l]=e;
}

bool Process(double P[]) {
  while(true) {
    int e=-1;
    double mV=-INF;
    for(int i=1; i<=nCnt; i++)
      if(fcmp(P[N[i]],mV)==1)
        mV=P[N[i]],e=N[i];

    if(fcmp(mV,0.0)<=0) break;
    int l=-1;
    mV=INF;
    for(int i=1; i<=bCnt; i++) {
      if(fcmp(A[i][e],0.0)==1) {
        double t=b[i]/A[i][e];
        if(fcmp(mV,t)==1||(fcmp(mV,t)==0&&(l==-1||B[l]>B[i])))
          mV=t,l=i;
      }
    }
    if(l==-1) return false;
    Pivot(l,e);
  }
  return true;
}

bool initSimplex() {
  nCnt=bCnt=0;
  for(int i=1; i<=n; i++)
    N[++nCnt]=i;
  for(int i=1; i<=m; i++)
    B[++bCnt]=i+n,A[i][n+i]=1.0;
  R=bCnt,C=bCnt+nCnt;
  double minV=INF;
  int p=-1;
  for(int i=1; i<=m; i++)
    if(fcmp(minV,b[i])==1)
      minV=b[i],p=i;
  if(fcmp(minV,0.0)>=0)
    return true;
  N[++nCnt]=n+m+1;
  R++,C++;
  for(int i=0; i<=C; i++)
    A[R][i]=0.0;
  for(int i=1; i<=R; i++)
    A[i][n+m+1]=-1.0;
  Pivot(p,n+m+1);
  if(!Process(A[R])) return false;
  if(fcmp(b[R],0.0)!=0)
    return false;
  p=-1;
  for(int i=1; i<=bCnt&&p==-1; i++)
    if(B[i]==n+m+1) p=i;
  if(p!=-1) {
    for(int i=1; i<=nCnt; i++) {
      if(fcmp(A[p][N[i]],0.0)!=0) {
        Pivot(p,N[i]);
        break;
      }
    }
  }
  bool f=false;
  for(int i=1; i<=nCnt; i++) {
    if(N[i]==n+m+1) f=true;
    if(f&&i+1<=nCnt)
      N[i]=N[i+1];
  }
  nCnt--;
  R--,C--;
  return true;
}

//-1: no solution 1: no bound 0: has a solution -V
int Simplex() {
  if(!initSimplex())
    return -1;
  if(!Process(c))
```

```
    return 1;
  for(int i=1; i<=nCnt; i++)
    X[N[i]]=0.0;
  for(int i=1; i<=bCnt; i++)
    X[B[i]]=b[i];
  return 0;
}

int main() {
  //n = 1;m=1;
  //V= 0.0;
  //c[1] = 1.0;
  //A[1][1] = 1.0;
  //b[1] = 5.0;
  //Simplex();
  //printf("V = %.3f\n",V);

  while(scanf("%d",&v1[1]) == 1) {
    for(int i = 2; i<=6; i++)
      scanf("%d",&v1[i]);
    n = 4;
    m = 6;
    for(int i = 0 ; i<=m+1; i++)
      for(int j=0; j<=n+m+2; j++)
        A[i][j] = c[j] = 0;
    memset(b,0,sizeof(b));
    V = 0.0;
    /*
    n 为未知数个数
    m 为约束个数
    目标：siama(c[i]*xi)
    约束：sigma(A[i][j]*xj) <=b[i]; j = 1 ... n
    解存在X里面
    */
    b[1] = v1[1] ;
    A[1][1] = 1;
    A[1][4] = 1;
    b[2] = v1[2] ;
    A[2][1] = 1;
    A[2][3] = 1;
    b[3] = v1[3] ;
    A[3][3] = 1;
    A[3][4] = 1;
    b[4] = v1[4] ;
    A[4][2] = 1;
    A[4][3] = 1;
    b[5] = v1[5] ;
    A[5][2] = 1;
    A[5][4] = 1;
    b[6] = v1[6] ;
    A[6][1] = 1;
    A[6][2] = 1;
    c[1] = 1;
    c[2] = 1;
    c[3] = 1;
    c[4] = 1;
    Simplex();
    //printf("V = %.3f\n",V);
    printf("%.3f␣%.3f␣%.3f␣%.3f\n",X[1],X[2],X[3],X[4]);

  }
  return 0;
}
```

5.12 Factorization prime number p into $x^2 + y^2$

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
int p,expp,A,B,aa,ans,tt;
long long M;
long long exp(int a,int b,long long mod) {
  long long ans=1,num=a;
  while (b!=0) {
    if (b&1) {
      ans=((ans%mod)*(num%mod))%mod;
    }
    num=((num%mod)*(num%mod))%mod;
    b>>=1;
  }
  return ans;
}
int calcu(int p,int &x,int &y) {
  if (p%4!=1) return -1;
  else {
    expp=(p-1)/4;
    A,B;
    while (1) {
      aa=rand()%p;
      if (aa==0) continue;
      A=exp(aa,expp,p);
      ans=(((long long)A%p)*((long long)A%p))%p;
      if (ans==p-1) break;
    }
    B=1;
    M=((long long)A*(long long)A+(long long)B*(long long)B)/p;
    if (M!=1) B=p;
    while (M!=1) {
      if (B>A) {
        tt=A;
        A=B;
        B=tt;
      }
      tt=A;
      A=B;
      B=tt%B;
      M=((long long)A*(long long)A
         +(long long)B*(long long)B)/p;
    }
    if (B<=A) {
      x=B;
      y=A;
    } else {
      x=A;
      y=B;
    }
  }
}
int main() {
  while (scanf("%d",&p)!=EOF) {
    int x,y;
    if (calcu(p,x,y)!=-1)
    }
  return 0;
}
```

5.13 Partition ways of an integer

$O(n\sqrt{n})$

```
#include <cstdio>
#include <cmath>
#include <cstring>
#include <map>
#include <algorithm>
using namespace std;
bool check(int x) {
  for (int i=2; i*i<=x; i++)
    if (x%i==0)
      return 0;
  return 1;
}
int p[100000];
inline int calc(int x) {
  return x*(x*3-1)/2;
}
int main() {
  p[0]=1;
  for (int i=1; i<100000; i++) {
    for (int j=1,k=1; calc(j)<=i; j++,k*=-1) {
      p[i]+=k*p[i-calc(j)];
      if (p[i]<0)
        p[i]+=1000000;
      if (p[i]>=1000000)
        p[i]-=1000000;
      if (calc(-j)<=i)
        p[i]+=k*p[i-calc(-j)];
      if (p[i]<0)
        p[i]+=1000000;
      if (p[i]>=1000000)
        p[i]-=1000000;
    }
    if (!p[i])
      printf("%d\n",i);
  }
  return 0;
}
```

5.14 Pell's equation

```
import java.math.BigInteger;
import java.util.*;
public class Main {
  public static class Fraction {
    public BigInteger num,den;
    public Fraction() {
      num=BigInteger.ZERO;
      den=BigInteger.ONE;
    }
    public Fraction(int _num,int _den) {
      num=BigInteger.valueOf(_num);
      den=BigInteger.valueOf(_den);
    }
    public Fraction(BigInteger _num,BigInteger _den) {
      num=_num;
      den=_den;
    }
    public Fraction gen() {
      BigInteger g=num.gcd(den);
      return new Fraction(num.divide(g),den.divide(g));
    }
    public Fraction add(Fraction x) {
      return new Fraction(x.num.multiply(den).add(num.multiply(x.den)),x.den.multiply(den)).gen();
    }
    public Fraction reciprocal() {
      return new Fraction(den,num);
    }
    public void out() {
      System.out.println(num+"/"+den);
    }
  }
  public static BigInteger sqrt(BigInteger a) {
    BigInteger b=a;
    while (a.compareTo(b.multiply(b))<0)
      b=b.multiply(b).add(a).divide(b.multiply(BigInteger.valueOf(2)));
    return b;
  }
  public static boolean check(Fraction x,int n) {
    return x.num.multiply(x.num).add(x.den.multiply(x.den.multiply(BigInteger.valueOf(n))).negate()).compareTo(BigInteger.ONE)==0;
  }
  static int p[]=new int[1000];
  static int l;
  public static void main(String[] args) {
    BigInteger ans=BigInteger.ZERO;
    int idx=0;
    for (int n=2,r=2; n<=1000; n++) {
      if (n==r*r) {
        r++;
        continue;
      }
      int tmp=calc(n,0,1),a=tmp,b=n-tmp*tmp;
      p[0]=tmp;
      l=1;
      while (true) {
        tmp=calc(n,a,b);
        p[l++]=tmp;
        a=a-tmp*b;
        Fraction x=getFrac();
        if (check(x,n)) {
          if (ans.compareTo(x.num)<0) {
            ans=x.num;
            idx=n;
          }
          break;
        }
        a=-a;
        b=(n-a*a)/b;
      }
    }
    System.out.println(idx);
  }
  private static Fraction getFrac() {
    Fraction ret=new Fraction(p[l-1],1);
    for (int i=l-2; i>=0; i--)
      ret=new Fraction(p[i],1).add(ret.reciprocal());
    return ret;
  }
  private static int calc(int n, int a, int b) {
    for (long i=2;; i++)
      if ((i*b-a)*(i*b-a)>n)
        return (int)i-1;
  }
}
```

5.15 Polya

设 G 是 p 个对象的一个置换群，用 k 种颜色去染这 p 个对象，若一种染色方案在群 G 的作用下变为另一种方案，则这两个方案当作是同一种方案，这样的不同染色方案数为：
$L = \frac{1}{|G|} \times \Sigma(k^{C(f)}), f \in G$
$C(f)$ 为循环节，$|G|$ 表示群的置换方法数

对于有 n 个位置的手镯，有 n 种旋转置换和 n 种翻转置换

对于旋转置换：

$C(f_i) = gcd(n,i)$,i 表示一次转过 i 颗宝石，$i = 0$ 时 $c = n$；

对于翻转置换：

如果 n 为偶数：则有 $\frac{n}{2}$ 个置换 $C(f) = \frac{n}{2}$，有 $\frac{n}{2}$ 个置换 $C(f) = \frac{n}{2} + 1$

如果 n 为奇数：$C(f) = \frac{n}{2} + 1$

5.16 拉格朗日插值法

已知 $y = a_0 + a_1x + a_2x^2 + \cdots + a_{n-1}x^{n-1}$ 曲线上的 n 个点 $(x_1, y_1), (x_2, y_2), (x_3, y_3) \cdots (x_n, y_n)$ 用拉格朗日插值法可以不求系数可知任意 x 对应的 y 值。

$$\begin{aligned} y = {} & y_1 \frac{(x-x_2)(x-x_3)\cdots(x-x_n)}{(x_1-x_2)(x_1-x_3)\cdots(x_1-x_n)} \\ & + y_2 \frac{(x-x_1)(x-x_3)\cdots(x-x_n)}{(x_2-x_1)(x_2-x_3)\cdots(x_2-x_n)} \\ & + \cdots \\ & + y_n \frac{(x-x_1)(x-x_2)\cdots(x-x_{n-1})}{(x_n-x_1)(x_n-x_2)\cdots(x_n-x_{n-1})} \end{aligned}$$

特别的，如果 $x_1 \sim x_n$ 为连续自然数，那么对于下一个自然数对应的 y 值为：

$$y_{n+1} = (-1)^{n-1}C_n^0 y_1 + (-1)^{n-2}C_n^1 y_2 + \cdots + (-1)^0 C_n^{n-1} y_n$$

这个组合系数可以通过高斯消元暴出来，前提是要猜到它满足递推关系。

5.17 正多面体顶点着色

正四面体：$N = \frac{(n^4+11\times n^2)}{12}$
正六面体：$N = \frac{(n^8+17\times n^4+6\times n^2)}{24}$
正八面体：$N = \frac{(n^6+3\times n^4+12\times n^3+8\times n^2)}{24}$
正十二面体：$N = \frac{(n^{20}+15\times n^{10}+20\times n^8+24\times n^4)}{60}$
正二十面体：$N = \frac{(n^{12}+15\times n^6+44\times n^4)}{60}$

5.18 求和公式

$\sum k = \frac{n\times(n+1)}{2}$
$\sum 2k-1 = n^2$
$\sum k^2 = \frac{n\times(n+1)\times(2n+1)}{6}$
$\sum (2k-1)^2 = \frac{n\times(4n^2-1)}{3}$
$\sum k^3 = (\frac{n\times(n+1)}{2})^2$
$\sum (2k-1)^3 = n^2\times(2n^2-1)$
$\sum k^4 = \frac{n\times(n+1)\times(2n+1)\times(3n^2+3n-1)}{30}$
$\sum k^5 = \frac{n^2\times(n+1)^2\times(2n^2+2n-1)}{12}$
$\sum k\times(k+1) = \frac{n\times(n+1)\times(n+2)}{3}$
$\sum k\times(k+1)\times(k+2) = \frac{n\times(n+1)\times(n+2)\times(n+3)}{4}$
$\sum k\times(k+1)\times(k+2)\times(k+3) = \frac{n\times(n+1)\times(n+2)\times(n+3)\times(n+4)}{5}$

5.19 几何公式

球扇形：
全面积：$T=\pi r(2h+r_0)$，h 为球冠高，r_0 为球冠底面半径
体积：$V=\frac{2\pi r^2h}{3}$

5.20 小公式

Pick 公式：$A=E\times0.5+I-1$（A 是多边形面积，E 是边界上的整点，I 是多边形内部的整点）

海伦公式：$S=\sqrt{p(p-a)(p-b)(p-c)}$，其中 $p=\frac{(a+b+c)}{2}$，abc 为三角形的三条边长

求 $\binom{n}{k}$ 中素因子 P 的个数：

1. 把 n 转化为 P 进制，并记它每个位上的和为 $S1$
2. 把 $n-k$，k 做同样的处理，得到 $S2$，$S3$

则 $\binom{n}{k}$ 中素因子 P 的个数：$\frac{S2+S3-S1}{P-1}$

部分错排公式：
$n+m$ 个数中 m 个数必须错排求排列数

```
dp[i] = n*dp[i-1]+(i-1)*(dp[i-1]+dp[i-2]);
dp[0] = n!;
dp[1] = n*n!;
```

$dp[m]$ 为所求解

6 Search

6.1 Dancing links

```
struct DLX {
  int h,n,m,tot;
  int U[MaxN*MaxM],D[MaxN*MaxM],L[MaxN*MaxM],R[MaxN*MaxM],Row[MaxN*MaxM],Col[MaxN*MaxM];
  int S[MaxM],O[MaxN];
```

```
bool hasans;
void init() {
  h = 0;
  hasans = false;
  tot = m+n;
  for (int i = 0; i <= m; i++) {
    D[i] = U[i] = Col[i] = i;
    Row[i] = S[i] = 0;
    L[i] = (i+m)%(m+1);
    R[i] = (i+1)%(m+1);
  }
  for (int i = 1; i <= n; i++) {
    R[i+m] = L[i+m] = i+m;
    Row[i+m] = i;
    Col[i+m] = 0;
  }
}
void insert(int x,int y) {
  tot++;
  Row[tot] = x;
  Col[tot] = y;
  S[y]++;
  int colPos,rowPos;
  colPos = y;
  while (true) {
    colPos = D[colPos];
    if (colPos == y || Row[colPos] > x)    break;
  }
  colPos = U[colPos];
  if (Row[colPos] == x)   return;
  U[tot] = colPos;
  D[tot] = D[colPos];
  U[D[tot]] = D[U[tot]] = tot;
  rowPos = x+m;
  while (true) {
    rowPos = R[rowPos];
    if (rowPos == x+m || Col[rowPos] > y)    break;
  }
  rowPos = L[rowPos];
  if (Col[rowPos] == y)   return;
  L[tot] = rowPos;
  R[tot] = R[rowPos];
  L[R[tot]] = R[L[tot]] = tot;
}
void print(int deep) {
  for (int i = 0; i < deep; i++)
    printf("%d␣", O[i]);
  printf("\n");
}
void cover(int col) {
  L[R[col]] = L[col];
  R[L[col]] = R[col];
  for (int i = D[col]; i != col; i = D[i])
    for (int j = R[i]; j != i; j = R[j])
      if (Col[j] != col) {
        U[D[j]] = U[j];
        D[U[j]] = D[j];
        S[Col[j]]--;
      }
}
void resume(int col) {
  for (int i = U[col]; i != col; i = U[i])
    for (int j = L[i]; j != i; j = L[j])
      if (Col[j] != col) {
          S[Col[j]]++;
          U[D[j]] = j;
          D[U[j]] = j;
        }
    L[R[col]] = col;
    R[L[col]] = col;
  }
  void initDFS() {
    for (int i = 1; i <= n; i++) {
      L[R[i+m]] = L[i+m];
      R[L[i+m]] = R[i+m];
    }
  }
  void DFS(int deep) {
    if (hasans == true) return;
    if (R[0] == 0)  {
      hasans = true;
      print(deep);
      return;
    };
    int tc = R[0];
    for (int i = R[0]; i != 0; i = R[i])
      if (S[i] < S[tc])   tc = i;
    cover(tc);
    for (int i = D[tc]; i != tc; i = D[i]) {
      int temp = O[deep];
      O[deep] = Row[i];
      for (int j = R[i]; j != i; j = R[j])
        cover(Col[j]);
      DFS(deep+1);
      for (int j = L[i]; j != i; j = L[j])
        resume(Col[j]);
      O[deep] = temp;
    }
    resume(tc);
  }
}
```

6.1.1 Usage

```
DLX g;
g.n = ROW_SIZE;
g.m = COL_SIZE;
g.init();
g.insert(ROW, COL);
g.initDFS();
g.DFS(0);
```

6.2 Dancing links (A-star)

```
namespace DLX {
const int MAXN = 1000;
const int MAXM = 400;
const int INF = 0x3f3f3f3f;
int D[MAXN * MAXM], U[MAXN * MAXM], L[MAXN * MAXM], R[MAXN * MAXM], COL[MAXN * MAXM], ROW[MAXN * MAXM];
int CNT, BEG[MAXN * MAXM], END[MAXN * MAXM], ANS, USE[MAXM], _USE[MAXM];
int SUM[MAXM];
bool vis[MAXM];
void init(int n) {
  memset(BEG, 0xff, sizeof(BEG));
  for(int i = 1; i <= n; i++)
    SUM[L[i + 1] = R[i – 1] = D[i] = U[i] = i] = 0;
  L[L[1] = R[n] = 0] = n, CNT = n + 1;
  ANS = n + 1;
}
void link(int r, int c) {
  D[CNT] = D[c], U[CNT] = c, U[D[c]] = CNT, D[c] = CNT, COL[CNT] = c, ROW[CNT] = r, SUM[c]++;
  if (BEG[r] == –1) BEG[r] = END[r] = CNT;
  R[END[r]] = CNT, L[CNT] = END[r], R[CNT] = BEG[r], L[BEG[r]] = CNT, END[r] = CNT++;
}
void DLX_Remove_Repeat(int c) {
  for (int i = D[c]; i != c; i = D[i])
    L[R[i]] = L[i], R[L[i]] = R[i], SUM[COL[i]]––;
}
void DLX_Resume_Repeat(int c) {
  for (int i = U[c]; i != c; i = U[i])
    L[R[i]] = i, R[L[i]] = i, SUM[COL[i]]++;
}
int Heuristics() {
  memset(vis, true, sizeof(vis));
  int c, i, j, cnt=0;
  for(c=R[0]; c; c=R[c])
    if(vis[c])
      for(cnt++, vis[c] = false, i = D[c]; i != c; i = D[i])
        for(j = R[i]; j != i; j = R[j])
          vis[COL[j]] = false;
  return cnt;
}
void DLX_Dfs(int n) {
  if (Heuristics() + n >= ANS) return;
  if (R[0] == 0) {
    ANS = n;
    for (int i = 0; i < n; i++)
      USE[i] = _USE[i];
    return ;
  }
  int i,now = INF,c;
  for (i = R[0]; i; i = R[i])
    if (now > SUM[i])
      now = SUM[c = i];
  for(i = D[c]; i != c; i = D[i]) {
    DLX_Remove_Repeat(i);
    for(int j = R[i]; j != i; j = R[j])
      DLX_Remove_Repeat(j);
    _USE[n] = ROW[i];
    DLX_Dfs(n + 1);
    for(int j = L[i]; j != i; j = L[j])
      DLX_Resume_Repeat(j);
    DLX_Resume_Repeat(i);
  }
}
void solve() {
  //ANS = m
  DLX_Dfs(0);
}
};
```

7 String

7.1 Aho-Corasick automation

Don’t forget running **BUILD**!

7.1.1 Static memory version

```
struct Trie {
  int next[50][10],fail[50];
  bool end[50];
  int L,root;
  int newNode() {
    for (int i = 0; i < 10; i++)
      next[L][i] = -1;
    end[L] = false;
    return L++;
  }
  void Init() {
    L = 0;
    root = newNode();
  }
  void Insert(char s[]) {
    int now = root;
    for (int i = 0; s[i] != 0; i++) {
      if (next[now][s[i]-'0'] == -1)
        next[now][s[i]-'0'] = newNode();
      now = next[now][s[i]-'0'];
    }
    end[now] = true;
  }
  void Build() {
    queue<int> Q;
    for (int i = 0; i < 10; i++)
      if (next[root][i] == -1)
        next[root][i] = root;
      else {
        fail[next[root][i]] = root;
        Q.push(next[root][i]);
      }
    while (!Q.empty()) {
      int now = Q.front();
      Q.pop();
      end[now] |= end[fail[now]];
      for (int i = 0; i < 10; i++)
        if (next[now][i] == -1)
          next[now][i] = next[fail[now]][i];
        else {
          fail[next[now][i]] = next[fail[now]][i];
          Q.push(next[now][i]);
        }
    }
  }
};
```

7.1.2 Pointer version

```
const int CHAR=26;
const int TOTLEN=500000;
const int MAXLEN=1000000;
struct Vertex {
  Vertex *fail,*next[CHAR];
  Vertex() {}
  Vertex(bool flag) { //为什么要这样写？
    fail=0;
    memset(next,0,sizeof(next));
  }
};
int size;
Vertex vertex[TOTLEN+1];
void init() {
  vertex[0]=Vertex(0);
  size=1;
}
void add(Vertex *pos,int cha) {
  vertex[size]=Vertex(0);
  pos->next[cha]=&vertex[size++];
}
void add(vector<int> s) {
  int l=s.size();
  Vertex *pos=&vertex[0];
  for (int i=0; i<l; i++) {
    if (pos->next[s[i]]==NULL)
      add(pos,s[i]);
    pos=pos->next[s[i]];
  }
}
void bfs() {
  queue<Vertex *> que;
  Vertex *u=&vertex[0];
  for (int i=0; i<CHAR; i++)
    if (u->next[i]!=NULL) {
      que.push(u->next[i]);
      u->next[i]->fail=u;
    } else
      u->next[i]=u;
  u->fail=NULL;
  while (!que.empty()) {
    u=que.front();
    que.pop();
    for (int i=0; i<CHAR; i++)
      if (u->next[i]!=NULL) {
        que.push(u->next[i]);
        u->next[i]->fail=u->fail->next[i];
      } else
        u->next[i]=u->fail->next[i];
  }
}
```

7.2 KMP

Match the suffix of $A[\cdots i]$ and the prefix of B

```
//Self match
int j;
p [0] = j = -1;
for ( int i = 1; i < lb; i++) {
  while (j >= 0 && b[j + 1] != b[i]) j = p[j];
  if (b[j + 1] == b[i]) j ++;
  p[i] = j;
}
//Match
j = -1;
for ( int i = 0; i < la; i++) {
  while (j >= 0 && b[j + 1] != a[i]) j = p[j];
  if (b[j + 1] == a[i]) j ++;
  KMP[i] = j + 1;
}
```

7.3 E-KMP

Common prefix of $A[i\cdots]$ and B

```
//Self match
int j = 0;
while (j < lb && b[j] == b[j + 1])
  j++;
p[0] = lb, p[1] = j;
int k = 1;
for (int i = 2; i < lb; i++) {
  int Len = k + p[k] - 1, L = p[i - k];
  if (L < Len - i + 1)
    p[i] = L;
  else {
    j = max(0, Len - i + 1);
    while (i + j < lb && b[i + j] == b[j])
      j++;
    p[i] = j, k = i;
  }
}
//Match
j = 0;
while (j < la && j < lb && a[j] == b[j])
  j++;
eKMP[0] = j;
k = 0;
for (int i = 1; i < la; i++) {
  int Len = k + eKMP[k] - 1, L = p[i - k];
  if (L < Len - i + 1)
    eKMP[i] = L;
  else {
    j = max(0, Len - i + 1);
    while (i + j < la && j < lb && a[i + j] == b[j])
      j++;
    eKMP[i] = j, k = i;
  }
}
```

7.4 Manacher

```
const int maxn = 110000;

char Ma[maxn*2];
int Mp[maxn*2];
void Manacher(char s[],int len) {
  int l = 0;
  Ma[l++] = '.';
  Ma[l++] = ',';
  for (int i = 0; i < len; i++) {
    Ma[l++] = s[i];
    Ma[l++] = ',';
  }
  Ma[l] = 0;
  int pnow = 0,pid = 0;
  for (int i = 1; i < l; i++) {
    if (pnow > i)
      Mp[i] = min(Mp[2*pid-i],pnow-i);
    else
      Mp[i] = 1;
    for (; Ma[i-Mp[i]] == Ma[i+Mp[i]]; Mp[i]++);
    if (i+Mp[i] > pnow) {
      pnow = i+Mp[i];
      pid = i;
    }
  }
}
/*
abaaba
 .  ,  a  ,  b  ,  a  ,  a  ,  b  ,  a  ,
 0  1  2  1  4  1  2  7  2  1  4  1  2  1
*/
```

7.5 Suffix array

```
const int maxn = 200010;
int wx[maxn],wy[maxn],*x,*y,wss[maxn],wv[maxn];

bool cmp(int *r,int n,int a,int b,int l) {
  return a+l<n && b+l<n && r[a]==r[b]&&r[a+l]==r[b+l];
}
void da(int str[],int sa[],int rank[],int height[],int n,int m) {
  int *s = str;
  int *x=wx,*y=wy,*t,p;
  int i,j;
  for(i=0; i<m; i++)wss[i]=0;
  for(i=0; i<n; i++)wss[x[i]=s[i]]++;
  for(i=1; i<m; i++)wss[i]+=wss[i-1];
  for(i=n-1; i>=0; i--)sa[--wss[x[i]]]=i;
  for(j=1,p=1; p<n && j<n; j*=2,m=p) {
    for(i=n-j,p=0; i<n; i++)y[p++]=i;
    for(i=0; i<n; i++)if(sa[i]-j>=0)y[p++]=sa[i]-j;
    for(i=0; i<n; i++)wv[i]=x[y[i]];
    for(i=0; i<m; i++)wss[i]=0;
    for(i=0; i<n; i++)wss[wv[i]]++;
    for(i=1; i<m; i++)wss[i]+=wss[i-1];
    for(i=n-1; i>=0; i--)sa[--wss[wv[i]]]=y[i];
    for(t=x,x=y,y=t,p=1,i=1,x[sa[0]]=0; i<n; i++)
      x[sa[i]]=cmp(y,n,sa[i-1],sa[i],j)?p-1:p++;
  }
  for(int i=0; i<n; i++) rank[sa[i]]=i;
  for(int i=0,j=0,k=0; i<n; height[rank[i++]]=k)
    if(rank[i]>0)
      for(k?k--:0,j=sa[rank[i]-1];
          i+k < n && j+k < n && str[i+k]==str[j+k];
          k++);
}
```

7.5.1 Longest common prefix

```
int lcp(int x,int y) {
  if (x > y)  swap(x,y);
  if (x == y)
    return len-sa[x];//NOTICE!
  x++;
  int k = lent[y-x+1];
  return min(f[x][k],f[y-(1<<k)+1][k]);
}
//Interval
```

```
void getinterval(int pos,int comlen,int& pl,int& pr) {
  int l,r,mid,cp;
  l = 0;
  r = pos;
  while (l < r) {
    mid = l+r>>1;
    cp = lcp(mid,pos);
    if (cp < comlen)
      l = mid+1;
    else
      r = mid;
  }
  pl = l;
  l = pos;
  r = len-1;
  while (l < r) {
    mid = l+r+1>>1;
    cp = lcp(pos,mid);
    if (cp < comlen)
      r = mid-1;
    else
      l = mid;
  }
  pr = l;
}
```

7.6 Smallest represention

```
int Gao(char a[],int len) {
  int i = 0,j = 1,k = 0;
  while (i < len && j < len && k < len) {
    int cmp = a[(j+k)%len]-a[(i+k)%len];
    if (cmp == 0)
      k++;
    else {
      if (cmp > 0)
        j += k+1;
      else
        i += k+1;
      if (i == j) j++;
      k = 0;
    }
  }
  return min(i,j);
}
```

8 Tool

8.1 Bit compression

```
int bit[5];
inline int getbit26(int sta, int pos) {
  return sta / bit[pos] % bit[1];
}
inline int setbit26(int sta, int pos, int val) {
  return sta / bit[pos + 1] * bit[pos + 1] + val * bit[pos] + sta % bit[pos];
}
//bin
inline int getbit(int sta, int pos) {
  return (sta >> pos) & 1;
}
```

```
inline int setbit(int sta, int pos, int val) {
  return ((sta >> (pos + 1)) << (pos + 1)) | (val << pos) | (sta & ((1 << pos) - 1));
}
```

8.2 Hash map

```
struct hash_map {
  int head[MOD];
  struct hash_tables {
    int key1, key2;
    long long val;
    int next;
  } ele[ELE];
  int N;
  int getHash(int key1, int key2) {
    return (key1 * 1000000 + key2) % MOD;
  }
  void init() {
    memset(head, -1, sizeof(head));
    N = 0;
  }
  void clear() {
    for (int i = 0; i < N; i++)
      head[getHash(ele[i].key1, ele[i].key2)] = -1;
    N = 0;
  }
  int fint(int key1, int key2) {
    for (int i = head[getHash(key1, key2)]; i != -1; i = ele[i].next) {
      if (ele[i].key1 == key1 && ele[i].key2 == key2)
        return i;
    }
    return -1;
  }
  void insert(int key1, int key2) {
    int tmp = getHash(key1, key2);
    ele[N].key1 = key1;
    ele[N].key2 = key2;
    ele[N].val = 0;
    ele[N].next = head[tmp];
    head[tmp] = N++;
  }
  long long get(int key1, int key2) {
    int tmp = fint(key1, key2);
    if (tmp == -1) {
      insert(key1, key2);
      return ele[N - 1].val;
    } else
      return ele[tmp].val;
  }
  void set(int key1, int key2, long long val) {
    int tmp = fint(key1, key2);
    if (tmp == -1) {
      insert(key1, key2);
      ele[N - 1].val = val;
    } else
      ele[tmp].val = val;
  }
  void add(int key1, int key2, long long val) {
    int tmp = fint(key1, key2);
    if (tmp == -1) {
      insert(key1, key2);
      ele[N - 1].val += val;
    } else
```

```
      ele[tmp].val += val;
  }
};
```

8.3 120 bit integer

```
struct integer {
  long long pa, pb;
  integer() {}
  integer(long long _pa, long long _pb) {
    pa = _pa;
    pb = _pb;
  }
  integer negate() {
    if (pa == 0 && pb == 0)
      return integer(pa, pb);
    else if (pa == 0)
      return integer(pa, -pb);
    else
      return integer(-pa, pb);
  }
  integer operator +(const integer& b) const {
    integer ret = integer(pa + b.pa, pb + b.pb);
    if (ret.pb >= MOD) {
      ret.pa += 1;
      ret.pb -= MOD;
    }
    return ret;
  }
  bool operator <(const integer& b) const {
    if (pa == b.pa)
      return pb < b.pb;
    return pa < b.pa;
  }
};
```

8.4 Bash script

```
while true; do
  ./gen > input
  ./sol < input > output.sol
  ./bf < input > output.bf

  diff output.sol output.bf
  if [ $? -ne 0 ] ; then break; fi
done
```

8.5 Codeblocks settings

```
gnome-terminal -t $TITLE -x
```

8.6 Bit operation

8.6.1 基本操作

注意括号

功能	示例	位运算
去掉最后一位	$(101101 \to 10110)$	x shr 1
在最后加一个 0	$(101101 \to 1011010)$	x shl 1
在最后加一个 1	$(101101 \to 1011011)$	x shl 1+1
把最后一位变成 1	$(101100 \to 101101)$	x or 1
把最后一位变成 0	$(101101 \to 101100)$	x or 1-1
最后一位取反	$(101101 \to 101100)$	x xor 1
把右数第 k 位变成 1	$(101001 \to 101101, k = 3)$	x or (1 shl (k-1))
把右数第 k 位变成 0	$(101101 \to 101001, k = 3)$	x and not (1 shl (k-1))
右数第 k 位取反	$(101001 \to 101101, k = 3)$	x xor (1 shl (k-1))
取末三位	$(1101101 \to 101)$	x and 7
取末 k 位	$(1101101 \to 1101, k = 5)$	x and (1 shl k-1)
取右数第 k 位	$(1101101 \to 1, k = 4)$	x shr (k-1) and 1
把末 k 位变成 1	$(101001 \to 101111, k = 4)$	x or (1 shl k-1)
末 k 位取反	$(101001 \to 100110, k = 4)$	x xor (1 shl k-1)
把右边连续的 1 变成 0	$(100101111 \to 100100000)$	x and (x+1)
把右起第一个 0 变成 1	$(100101111 \to 100111111)$	x or (x+1)
把右边连续的 0 变成 1	$(11011000 \to 11011111)$	x or (x-1)
取右边连续的 1	$(100101111 \to 1111)$	(x xor (x+1)) shr 1
去掉右起第一个 1 的左边	$(100101000 \to 1000)$	x and (x xor (x-1))

8.6.2 枚举长为 n 含 k 个 1 的 01 串

```c
int n = 5,k = 3;
for (int s = (1 << k)-1,u = 1 << n; s < u;) {
  for (int i = 0;i < n;i++)
    printf("%d",(((s>>(n-1-i))&1) == 1));
  printf("\n");

  int b = s & -s;
  s = (s+b)|(((s^(s+b))>>2)/b);
}
```

8.7 vimrc

```vim
syntax on

set backspace=start,indent,eol
set showmode
set showcmd
set hlsearch
set nowrap
set smarttab
set autoindent
set tabstop=4
set softtabstop=4
set shiftwidth=4
set number
filetype indent on

set makeprg=g++\ '%:p'\ -o\ '%:p.mzry'\ -Wall\ -g
function! Gao()
  exec "silent␣w"
  exec "silent␣!rm␣-f␣'%:p.mzry1992'"
  exec "silent␣make"
  exec "cw"
endfunction
```

```
function! Run()
  call Gao()
  let execFile = expand("%:p").".mzry"
  if filereadable(execFile)
    exec "silent␣!gnome-terminal␣-t␣'%:p.mzry'␣--working-directory='%:p:h'␣-x␣/usr/bin/cb_console_runner␣'%:p.mzry'"
  endif
endfunction

colorscheme slate
set gfn=Monospace\ 14

map <C-F9> :call Gao()<Enter>
imap <C-F9> <Esc>:call Gao()<Enter>
map <F9> :call Run()<Enter>
imap <F9> <Esc>:call Run()<Enter>

map <C-c> :s!^!//<Enter>:noh<Enter>
imap <C-c> <Esc>:s!^!//<Enter>:noh<Enter>
map <C-x> :s!//!<Enter>:noh<Enter>
imap <C-x> <Esc>:s!//!<Enter>:noh<Enter>
```

9 Appendix

9.1 Template by elfness

9.1.1 AC machine

```
#include<cstdio>
#include<cstring>
#include<cstdlib>
#include<cmath>
#include<algorithm>
#include<iostream>
using namespace std;
typedef long long LL;
struct tree {
  tree *ne[26],*fail;
  int ct;
} tr[500100],VD,*root,*Q[500100];
int tn;
void init() {
  tr[tn=0]=VD;
  root=tr+(tn++);
}
char s[1000100];
void build() {
  tree *p=root;
  for(int i=0; s[i]; i++) {
    if(p->ne[s[i]-'a']==NULL) {
      tr[tn]=VD;
      p->ne[s[i]-'a']=tr+(tn++);
    }
    p=p->ne[s[i]-'a'];
  }
  p->ct++;
}
void pre() {
  int i,top,tail;
  tree *p,*q;
  top=0;
  tail=0;
  for(i=0; i<26; i++)
    if(root->ne[i]!=NULL) {
      Q[++tail]=root->ne[i];
      root->ne[i]->fail=root;
    } else root->ne[i]=root;
  while(top<tail) {
    p=Q[++top];
    for(i=0; i<26; i++)
      if(p->ne[i]!=NULL) {
        q=p->ne[i];
        Q[++tail]=q;
        q->fail=p->fail->ne[i];
        if(q->fail==NULL)q->fail=root;
      } else p->ne[i]=p->fail->ne[i];
  }
}
int doit() {
  int ret=0;
  tree *p=root,*q;
  for(int i=0; s[i]; i++) {
    p=p->ne[s[i]-'a'];
    q=p;
    while(root!=q&&q->ct!=-1) {
      ret+=q->ct;
      q->ct=-1;
      q=q->fail;
    }
  }
  return ret;
}
int i,n,_;
int main() {
  for(i=0; i<26; i++)VD.ne[i]=NULL;
  VD.ct=0;
  scanf("%d",&_);
  while(_--) {
    scanf("%d",&n);
    init();
    for(i=0; i<n; i++) {
      scanf("%s",s);
      build();
    }
    pre();
    scanf("%s",s);
    printf("%d\n",doit());
  }
}
```

9.1.2 E-KMP

```
#include<cstdio>
#include<cstring>
#include<cstdlib>
#include<cmath>
#include<algorithm>
#include<iostream>
using namespace std;
typedef long long LL;
void e_kmp(char *s,char *t,int *has,int *e_has) {
  int sp,p,mx,tn;
  for(sp=p=mx=0; s[p]>0; p++) {
    if(mx==p||p+e_has[p-sp]>=mx ) {
      for(tn=mx-p; s[mx]==t[tn]; tn++)mx++;
      has[sp=p]=mx-p;
      if(mx==p)sp=++mx;
    } else has[p]=e_has[p-sp];
  }
}
const int V=1001000;
char t[V],s[V];
int e_has[V],has[V],tn;
int main() {
  scanf("%s%s",s,t);
  tn=strlen(t);
  t[tn]=-1;
  e_has[0] = tn;
  e_kmp(t+1,t,e_has+1,e_has);
  e_kmp(s,t,has,e_has);
}
```

9.1.3 KM (list)

```
#include<cstdio>
#include<cstring>
#include<cstdlib>
#include<cmath>
#include<algorithm>
using namespace std;
const int V=1200;
const int En=21000;
const int oo=1000000000;
struct Edge {
  int num,ne,w;
} e[En];
int p[V],K;
void add(int x,int y,int z) {
  e[K].num=y;
  e[K].w=z;
  e[K].ne=p[x];
  p[x]=K++;
}
bool sx[V],sy[V];
int lx[V],ly[V],mat[V];
bool path(int u) {
  sx[u]=true;
  for(int i=p[u]; i!=-1; i=e[i].ne) {
    int v=e[i].num;
    if(!sy[v]&&lx[u]+ly[v]==e[i].w) {
      sy[v]=true;
      if(mat[v]==-1||path(mat[v])) {
        mat[v]=u;
        return true;
      }
    }
  }
  return false;
}
int N;
int KM() {
  int i,j;
  for(i=0; i<N; i++) {
    lx[i]=-oo;
    for(j=p[i]; j!=-1; j=e[j].ne)
      lx[i]=max(lx[i],e[j].w);
  }
  for(i=0; i<N; i++)ly[i]=0,mat[i]=-1;
  for(int u=0; u<N; u++)
    while(1) {
      for(i=0; i<N; i++)sx[i]=0,sy[i]=0;
      if(path(u))break;
      int dx=oo;
      for(i=0; i<N; i++)if(sx[i])
          for(j=p[i]; j!=-1; j=e[j].ne)
            if(!sy[e[j].num])
              dx=min(dx,lx[i]+ly[e[j].num]-e[j].w);
      if(dx==oo)return -1;
      for(i=0; i<N; i++)if(sx[i])lx[i]-=dx;
      for(i=0; i<N; i++)if(sy[i])ly[i]+=dx;
    }
  int ret=0;
  for(i=0; i<N; i++)ret+=lx[i]+ly[i];
  return -ret;
}
int _,ca,n,m,i,x,y,z,te;
int main() {
  scanf("%d",&_);
  ca=0;
  while(_--) {
    ca++;
    scanf("%d%d",&n,&m);
    N=n;
    for(i=0; i<n; i++)p[i]=-1;
    K=0;
    for(i=0; i<m; i++) {
      scanf("%d%d%d",&x,&y,&z);
      x--;
      y--;
      add(x,y,-z);
      add(y,x,-z);
    }
    te=KM();
    printf("Case␣%d:␣",ca);
    if(te==-1)puts("NO");
    else printf("%d\n",te);
  }
}
```

9.1.4 Nearest point pair

```
/*
 * nearestPointPair.cpp
 *
 *  Created on: 2011-10-10
 *      Author: Fish
 */

#include <cstdio>
#include <cstring>
#include <cstdlib>
#include <cmath>
#include <algorithm>

using namespace std;

const int MaxN = 120000;
const int Log = 20;

struct Point {
  double x, y;
  Point() {
  }
  Point(double x, double y) :
    x(x), y(y) {
  }
  Point operator-(const Point& p) const {
    return Point(x - p.x, y - p.y);
  }
  double norm() const {
    return hypot(x, y);
  }
  void init() {
    scanf("%lf%lf", &x, &y);
  }
} p[MaxN];
int x[MaxN], y[Log][MaxN], tmp[MaxN], n;
bool vst[MaxN];

bool comp_x(const int& i, const int& j) {
  return p[i].x < p[j].x;
}

bool comp_y(const int& i, const int& j) {
  return p[i].y < p[j].y;
}

double dfs(int k, int l, int r) {
  double ret = 1e100;
  if (r - l <= 2) {
    for (int i = l; i < r; i++)
      for (int j = i + 1; j <= r; j++)
        ret = min(ret, (p[x[i]] - p[x[j]]).norm());
    return ret;
  }

  int mid = (l + r) >> 1;
  int lp = l, rp = mid + 1;
  for (int i = l; i <= r; i++)
    vst[x[i]] = i <= mid;
  for (int i = l; i <= r; i++)
    if (vst[y[k][i]])
      y[k + 1][lp++] = y[k][i];
    else
      y[k + 1][rp++] = y[k][i];
  double lhs = dfs(k + 1, l, mid);
  double rhs = dfs(k + 1, mid + 1, r);
  double mx = (p[x[mid + 1]].x + p[x[mid]].x) / 2.0;
  ret = min(lhs, rhs);

  lp = 0;
  for (int i = l; i <= r; i++)
    if (fabs(mx - p[y[k][i]].x) < ret)
      tmp[lp++] = y[k][i];

  for (int i = 0; i < lp; i++)
    for (int j = 1; j < 8 && i + j < lp && (p[tmp[i + j]].y - p[tmp[i]].y) < ret; j++)
      ret = min(ret, (p[tmp[i]] - p[tmp[i + j]]).norm());

  return ret;
}

int main() {
#ifdef __FISH__
```

```
  freopen("data.in", "r", stdin);
  freopen("nlogn.out", "w", stdout);
#endif
  while (scanf("%d", &n) == 1 && n) {
    for (int i = 0; i < n; i++) {
      p[i].init();
      x[i] = y[0][i] = i;
    }
    sort(x, x + n, comp_x);
    sort(y[0], y[0] + n, comp_y);
    printf("%.2f\n", dfs(0, 0, n - 1) / 2.0);
    // printf("%.6f\n", dfs(0, 0, n - 1));
  }

  return 0;
}
```

9.1.5 SA

```
#include<cstdio>
#include<cstring>
#include<cstdlib>
#include<cmath>
#include<algorithm>
#include<iostream>
#include<vector>
#include<string>
using namespace std;
typedef long long LL;
const int N=100100;
char s[N];  /// 长度+1，对于非字符串，加一个小于最小值的元素，
int sa[N];  /// 倍增算法，结果 下标 1—n，第 i 大的是 sa[i]
int rk[N];  /// 第 i 位开始的后缀，的排名为 rk[i]
int wa[N],wb[N],wv[N],rmq[20][N];
int sn,to[N];
bool cmp(int *y,int a,int b,int L) {
  return y[a]==y[b]&&y[a+L]==y[b+L];
}
void da(char *s,int *sa,int len,int dn) {
  int i,j,p;
  int *x,*y,*t;
  x=wa;
  y=wb;
  for(i=0; i<dn; i++)rk[i]= 0;
  for(i=0; i<len; i++)rk[x[i]=s[i]]++;
  for(i=0; i<dn; i++)rk[i+1]+=rk[i];
  for(i=len-1; i>=0; i--)sa[--rk[x[i]]]=i;
  for(j=1,p=1; p<len; j*=2,dn=p) {
    for(p=0; p<j; p++)y[p]=len-j+p;
    for(i=0; i<len; i++)if(sa[i]>=j)y[p++]=sa[i]-j;
    for(i=0; i<len; i++)wv[i]=x[y[i]];
    for(i=0; i<dn; i++)rk[i]=0;
    for(i=0; i<len; i++)rk[wv[i]]++;
    for(i=0; i<dn; i++)rk[i+1]+=rk[i];
    for(i=len-1; i>=0; i--)sa[--rk[wv[i]]]=y[i];
    swap(x,y);
    x[sa[0]]=0;
    for(p=i=1; i<len; i++) {
      p+=!cmp(y,sa[i],sa[i-1],j);
      x[sa[i]]=p-1;
    }
  }
}
```

```
void find_height(char *s,int *sa,int len) {
  int *h=rmq[0];
  int i,j,k=0;
  for(i=1; i<=len; i++)
    rk[sa[i]] = i;
  for(i=0; i<len; i++) {
    if(k>0)k--;
    j=sa[rk[i]-1];
    while(s[i+k]==s[j+k])k++;
    h[rk[i]]=k;
  }
}
void RMQ(int n) {
  int i,j;
  int rn=(int)floor(log(n*2.0)/log(2.0));
  for(i=1; i<rn; i++)
    for(j=0; j<n+2-(1<<(i-1)); j++)
      rmq[i][j]=min(rmq[i-1][j],rmq[i-1][j+(1<<(i-1))]);
}
int askRMQ(int a,int b) { /// [a,b]闭区间
  int rq=to[b-a];
  return min(rmq[rq][a],rmq[rq][b+1-(1<<rq)]);
}
void PT(char *s,int *sa) {
  int i,sn;
  sn=strlen(s);
  for(i=0; i<sn; i++)
    puts(s+sa[i+1]);
  puts("");
  for(i=0; i<sn; i++)
    printf("rank␣%d␣=␣%d\n",i,rk[i]);
}
int lcp(int a,int b,int len) {
  if(a==b)
    return len-a;
  a=rk[a];
  b=rk[b];
  if(a>b)swap(a,b);
  return askRMQ(a+1,b);
}
void pre_log() {
  int i;
  to[0]=to[1]=0;
  for(i=1; i*2<N; i++)
    to[i*2]=to[i*2+1]=to[i]+1;
}
int main() {
  int T,_=0;
  pre_log();
  while(~scanf("%s",s)) {
    sn=strlen(s);
    da(s,sa,sn+1,128);
    find_height(s,sa,sn);
    RMQ(sn);
    PT(s,sa);
    scanf("%d",&T);
    while(T--) {
      int a,b;
      scanf("%d%d",&a,&b);
      a--,b--;/// 求原串的 a b 开始的后缀的公共前缀
      printf("lcp␣=␣%d\n",lcp(a,b,sn));
    }
  }
  return 0;
}
```

9.1.6 SAP

```
#include<cstdio>
#include<cstring>
#include<cstdlib>
#include<cmath>
#include<algorithm>
using namespace std;
const int V=220;
const int En=200000;
const int oo=0x3f3f3f3f;
struct Edge {
  int num,ne,c;
} e[En];
int d[V],p[V],pre[V],low[V];
int gap[V],cur[V];
int N,K,st,ed;
void add(int x,int y,int c) {
  e[K].num=y;
  e[K].c=c;
  e[K].ne=p[x];
  p[x]=K++;
  e[K].num=x;
  e[K].c=0;
  e[K].ne=p[y];
  p[y]=K++;
}
int sap() {
  int ret=0;
  bool fail;
  for(int i=0; i<=N; i++) {
    low[i]=gap[i]=d[i]=0;
    cur[i]=p[i];
  }
  low[st]=oo;
  gap[0]=N;
  int u=st;
  while(d[st]<N) {
    fail=true;
    for(int i=cur[u]; i!=-1; i=e[i].ne) {
      int v=e[i].num;
      cur[u]=i;
      if(e[i].c&&d[u]==d[v]+1) {
        pre[v]=i;
        low[v]=min(low[u],e[i].c);
        u=v;
        if(u==ed) {
          do {
            e[pre[u]].c-=low[ed];
            e[pre[u]^1].c+=low[ed];
            u=e[pre[u]^1].num;
          } while(u!=st);
          ret+=low[ed];
        }
        fail=false;
        break;
      }
    }
    if(fail) {
      gap[d[u]]--;
      if(!gap[d[u]])return ret;
      d[u]=N;
      for(int i=p[u]; i!=-1; i=e[i].ne)
        if(e[i].c)d[u]=min(d[u],d[e[i].num]+1);
      gap[d[u]]++;
      cur[u]=p[u];
      if(u!=st)u=e[pre[u]^1].num;
    }
  }
  return ret;
}
```

9.1.7 一般图最大匹配

```
#include <stdio.h>
#include <string.h>
#include <algorithm>
#include <vector>
#define maxn 300
#define maxm 90010

using namespace std;

int match[maxn];            //标记是否匹配
int st[maxn],aim[maxm],nxt[maxm],ln;  //边表
int q[maxn];              //bfs队列
int level[maxn];            //离根深度的奇偶性
vector<int> ar[maxn];         //存每个点到根的路径
vector<int> a;              //找到的一条增广路
int n;
void init() {
  for(int i=0; i<n; i++)st[i]=-1;
  ln=0;
}
void in_edge(int x,int y) {
  aim[ln]=y;
  nxt[ln]=st[x];
  st[x]=ln++;
}
int lca(int p,int q) {          //求p和q的最近公共祖先
  int ret=0;
  while (ret<ar[p].size() && ret<ar[q].size() && ar[p][ret]==ar[q][ret]) ret++;
  return ret-1;
}
int FindAlterRoad(int sp) {
  int qn=1;
  memset(level,-1,sizeof(level));
  level[q[0]=sp]=1;
  ar[sp].clear();
  ar[sp].push_back(sp);
  for (int p=0; p<qn; p++) {
    int x=q[p];
    for (int i=st[x]; i!=-1; i=nxt[i]) {
      int u=aim[i];
      if (match[u]==u) continue;
      if (level[u]==-1) {     //u是未访问的点
        if (match[u]==-1) {   //u是未匹配的，找到增广路
          a=ar[x];
          a.push_back(u);
          return 1;
        } else {          //u是已匹配的点
          int v=match[u];
          if (level[v]!=-1) continue;
          ar[v]=ar[x];
          ar[v].push_back(u);
          ar[v].push_back(v);
          level[u]=0;
          level[v]=1;
          q[qn++]=v;
        }
      } else if (level[u]==1) {     //u和x同为偶点.形成花
        int root=lca(u,x);
        vector<int> tmp=ar[x];
        for (int i=ar[u].size()-1; i>root; i--) {
          int y=ar[u][i];
          tmp.push_back(y);
          if (level[y]==0) {
            level[y]=1;
            ar[y]=tmp;
            level[y]=1;
            q[qn++]=y;
          }
        }
        tmp=ar[u];
        for (int i=ar[x].size()-1; i>root; i--) {
          int y=ar[x][i];
          tmp.push_back(y);
          if (level[y]==0) {
            level[y]=1;
            ar[y]=tmp;
            level[y]=1;
            q[qn++]=y;
          }
        }
      }
    }
  }
  return 0;
}
int MaximumMatch() {
  int ret=0;             //最大匹配数
  memset(match,-1,sizeof(match));
  for (int i=0; i<n; i++)
    if (match[i]==-1)
      if (FindAlterRoad(i)) {
        for (int i=0; i<a.size(); i+=2) {
          int u=a[i],v=a[i+1];
          match[u]=v;
          match[v]=u;
        }
        ret++;
      } else match[i]=i;
  return ret;
}
```

9.1.8 上下界最大流

```
/*
Author: elfness@UESTC
*/
#include<cstdio>
#include<cstring>
#include<cstdlib>
#include<cmath>
#include<algorithm>
#include<iostream>
#include<vector>
```

```
#include<string>
using namespace std;
typedef long long LL;
const int V=1500;
const int En=900000;
const int inf=0x3f3f3f3f;
struct Edge {
  int num,ne;
  int c;
} e[En];
int p[V],K;
void add(int x,int y,int c) {
  e[K].num=y;
  e[K].c=c;
  e[K].ne=p[x];
  p[x]=K++;
  e[K].num=x;
  e[K].c=0;
  e[K].ne=p[y];
  p[y]=K++;
}
int d[V],pre[V],pree[V],gap[V],cur[V];
int N,st,ed;
int low[V];
int sap() {
  int ret=0;
  bool fail;
  for(int i=0; i<=N; i++) {
    d[i]=0;
    gap[i]=0;
    cur[i]=p[i];
    low[i]=0;
  }
  low[st]=inf;
  gap[0]=N;
  int u=st;
  while(d[st]<N) {
    fail=true;
    for(int i=cur[u]; i!=-1; i=e[i].ne) {
      int v=e[i].num;
      cur[u]=i;
      if(e[i].c&&d[u]==d[v]+1) {
        pre[v]=u;
        pree[v]=i;
        low[v]=min(low[u],e[i].c);
        u=v;
        if(u==ed) {
          do {
            e[pree[u]].c-=low[ed];
            e[pree[u]^1].c+=low[ed];
            u=pre[u];
          } while(u!=st);
          ret+=low[ed];
        }
        fail=false;
        break;
      }
    }
    if(fail) {
      gap[d[u]]--;
      if(!gap[d[u]])return ret;
      d[u]=N;
      for(int i=p[u]; i!=-1; i=e[i].ne)
        if(e[i].c)d[u]=min(d[u],d[e[i].num]+1);
      gap[d[u]]++;
      cur[u]=p[u];
      if(u!=st)u=pre[u];
    }
  }
  return ret;
}
int n,m,s,t;
struct Elf {
  int u,v,lo,up;
} b[12000];
int lb[12000];
int doit() {
  int i;
  N=n+2;
  st=n;
  ed=n+1;
  for(i=0; i<N; i++)p[i]=-1;
  K=0;
  for(i=0; i<n; i++)lb[i]=0;
  for(i=0; i<m; i++) {
    lb[b[i].u]-=b[i].lo;
    lb[b[i].v]+=b[i].lo;
    add(b[i].u,b[i].v,b[i].up-b[i].lo);
  }
  for(i=0; i<n; i++) {
    if(lb[i]>0)add(st,i,lb[i]);
    else add(i,ed,-lb[i]);
  }
  add(t,s,inf);
  int te=sap();
  for(i=p[st]; i!=-1; i=e[i].ne)
    if(e[i].c!=0)return -1;
  st=s;
  ed=t;
  te=sap();
  return te;
}
```

9.1.9 上下界最小流

```
#include<cstdio>
#include<cstdlib>
#include<cstring>
#include<cmath>
#include<algorithm>
using namespace std;
const int V=600;
const int En=50000;
const int oo=0x3f3f3f3f;
struct Edge {
  int num,ne,c;
} e[En];
int p[V],K;
void add(int x,int y,int c) {
  e[K].num=y;
  e[K].c=c;
  e[K].ne=p[x];
  p[x]=K++;
  e[K].num=x;
  e[K].c=0;
  e[K].ne=p[y];
  p[y]=K++;
```

```
}
int d[V],cur[V],low[V],pre[V],gap[V],pree[V];
int st,ed,N;
int sap() {
  int ret=0;
  bool fail;
  memset(gap,0,sizeof(gap));
  memset(low,0,sizeof(low));
  memset(d,0,sizeof(d));
  for(int i=0; i<N; i++)cur[i]=p[i];
  gap[0]=N;
  low[st]=oo;
  int u=st;
  while(d[st]<N) {
    fail=true;
    for(int i=cur[u]; i!=-1; i=e[i].ne) {
      int v=e[i].num;
      cur[u]=i;
      if(e[i].c&&d[u]==d[v]+1) {
        pre[v]=u;
        pree[v]=i;
        low[v]=min(low[u],e[i].c);
        u=v;
        if(u==ed) {
          do {
            e[pree[u]].c-=low[ed];
            e[pree[u]^1].c+=low[ed];
            u=pre[u];
          } while(u!=st);
          ret+=low[ed];
        }
        fail=false;
        break;
      }
    }
    if(fail) {
      gap[d[u]]--;
      if(!gap[d[u]])return ret;
      d[u]=N;
      for(int i=p[u]; i!=-1; i=e[i].ne)
        if(e[i].c)d[u]=min(d[u],d[e[i].num]+1);
      gap[d[u]]++;
      cur[u]=p[u];
      if(u!=st)u=pre[u];
    }
  }
  return ret;
}
struct ELF {
  int u,v,lo;
} b[En];
int n,m,lb[V],ts,tt;
void solve() {
  N=n+4;
  ts=0;
  tt=n+1;
  st=n+2;
  ed=n+3;
  memset(lb,0,sizeof(lb));
  int i,u,v;
  for(i=0; i<N; i++)p[i]=-1;
  K=0;
  for(i=0; i<m; i++) {
    u=b[i].u;
    v=b[i].v;
    lb[v]+=b[i].lo;
    lb[u]-=b[i].lo;
    add(u,v,oo-b[i].lo);
  }
  for(i=1; i<=n; i++) {
    add(ts,i,oo);
    add(i,tt,oo);
  }
  for(i=0; i<n+2; i++) {
    if(lb[i]>0)add(st,i,lb[i]);
    else add(i,ed,-lb[i]);
  }
  int ans=sap();
  add(tt,ts,oo);
  printf("%d\n",sap());
}
int _,ca,i;
int main() {
  scanf("%d",&_);
  ca=0;
  while(_--) {
    ca++;
    scanf("%d%d",&n,&m);
    for(i=0; i<m; i++) {
      scanf("%d%d%d",&b[i].u,&b[i].v,&b[i].lo);
    }
    printf("Case␣#%d:␣",ca);
    solve();
  }
}
```

9.1.10 全局最小割

```
using namespace std;
#define inf 100000000
bool visit[502],com[502];
int map[502][502],W[502],s,t;
int maxadj(int N,int V) {
  int CUT;
  memset(visit,0,sizeof(visit));
  memset(W,0,sizeof(W));
  for(int i=0; i<N; i++) {
    int Num=0,Max=-inf;
    for(int j=0; j<V; j++)
      if(!com[j]&&!visit[j]&&W[j]>Max) {
        Max=W[j];
        Num=j;
      }
    visit[Num]=true;
    s=t;
    t=Num;
    CUT=W[t];
    for(int j=0; j<V; j++)
      if(!com[j]&&!visit[j])W[j]+=map[Num][j];
  }
  return CUT;
}
int stoer(int V) {
  int Mincut=inf;
  int N=V;
  memset(com,0,sizeof(com));
  for(int i=0; i<V-1; i++) {
    int Cut;
    s=0,t=0;
    Cut=maxadj(N,V);
    N--;
    if(Cut<Mincut)Mincut=Cut;
    com[t]=true;
    for(int j=0; j<V; j++)
      if(!com[j]) {
        map[j][s]+=map[j][t];
        map[s][j]+=map[t][j];
      }
  }
  return Mincut;
}
```

9.1.11 最小树型图

```
#include<cstdio>
#include<cstring>
#include<cstdlib>
#include<cmath>
#include<algorithm>
using namespace std;
const int V=1200;
const int En=2100000;
struct Elf {
  int u,v,len;
} b[En];
const int oo=1000000000;
int ret;
int N,M,Root;//点数，边数，根，默认从0开始
int id[V],pre[V],cnt,vis[V];
int in[V];
bool TreeMST() {
  ret=0;
  int i,u,v;
  while(1) {
    for(i=0; i<N; i++)
      in[i]=oo;
    memset(pre,-1,sizeof(pre));
    for(i=0; i<M; i++) {
      u=b[i].u;
      v=b[i].v;
      if(b[i].len<in[v]&&u!=v) {
        pre[v]=u;
        in[v]=b[i].len;
      }
    }
    for(i=0; i<N; i++) {
      if(i==Root)continue;
      if(pre[i]==-1)return false;
    }
    in[Root]=0;
    cnt=0;
    memset(id,-1,sizeof(id));
    memset(vis,-1,sizeof(vis));
    for(i=0; i<N; i++) {
      ret+=in[i];
      v=i;
      while(vis[v]!=i&&id[v]==-1&&v!=Root) {
        vis[v]=i;
        v=pre[v];
      }
      if(v!=Root&&id[v]==-1) {
        for(u=pre[v]; u!=v; u=pre[u])
          id[u]=cnt;
        id[v]=cnt++;
      }
    }
    if(cnt==0)return true;
    for(i=0; i<N; i++)
      if(id[i]==-1)id[i]=cnt++;
    for(i=0; i<M; i++) {
      v=b[i].v;
      b[i].u=id[b[i].u];
      b[i].v=id[b[i].v];
      if(b[i].u!=b[i].v)
        b[i].len-=in[v];
    }
    N=cnt;
    Root=id[Root];
  }
  return true;
}
```