SAJIHPTS Code Snippets

SAJIHPTS Team

October 20, 2016

Contents

1 Cal Geo

1.1 jisuan

```cpp
#include<iostream>
#include<cmath>
#include<algorithm>
#include<vector>
using namespace std;
#define EPS 1e-10
#define INF 1e10
#define PI 3.14159265358979323846
inline bool EQUAL(double t1, double t2){
  return t1 - t2 < EPS && t1 - t2 > -EPS;
}
inline bool LESS(double t1, double t2){
  return t1 <= t2 - EPS;
}
inline bool LESS_EQUAL(double t1, double t2){
  return t1 < t2 + EPS;
}
inline int SGN(double t){
  return LESS(t, 0) ? -1 : LESS(0, t) ? 1 : 0;
}
class Point
{
public:
  double x, y;
  Point(){}
  Point(double x, double y) :x(x), y(y){}

  bool operator == (const Point& p)const{
    return EQUAL(x, p.x) && EQUAL(y, p.y);
  }
  bool operator < (const Point& p)const{
    return LESS_EQUAL(x, p.x) && (LESS(x, p.x) || LESS(y, p.y));
  }
  Point operator + (const Point& p)const{
    return Point(x + p.x, y + p.y);
  }
  Point operator - (const Point& p)const{
    return Point(x - p.x, y - p.y);
  }
  double operator * (const Point& p)const{
    return x*p.y - y*p.x;
```

```
42    }
43    Point operator * (double value)const{
44      return Point(x*value, y*value);
45    }
46    Point operator / (double value)const{
47      return Point(x / value, y / value);
48    }
49    double dot(const Point& p)const{
50      return x*p.x + y*p.y;
51    }
52    double r2()const{ return x*x + y*y; }
53    double r()const{ return hypot(x, y); }
54    double dis2(const Point& p)const{
55      return (*this - p).r2();
56    }
57    double dis(const Point& p)const{
58      return (*this - p).r();
59    }
60
61    bool onLine(const Point& p1, const Point& p2)const{
62      return EQUAL((*this - p1)*(*this - p2), 0);
63    }
64    bool onLineSeg(const Point& p1, const Point& p2)const{
65      //include extream points
66      return onLine(p1, p2) && inRect(p1, p2);
67    }
68    double lineRelation(const Point& p1, const Point& p2)const{
69      Point t = p2 - p1;
70      return t.dot(*this - p1) / t.r2();
71      //ret 0, *this=p1; ret 1,*this=p2;
72      //ret (0,1), *this is interior to p1p2
73    }
74    Point footPoint(const Point& p1, const Point& p2)const{
75      double r = lineRelation(p1, p2);
76      return p1 + (p2 - p1)*r;
77    }
78    double lineDis(const Point& p1, const Point& p2)const{
79      return abs((p1 - *this)*(p2 - *this)) / p1.dis(p2);
80    }
81    double lineSegDis(const Point& p1, const Point& p2, Point&
      ↪  ret)const;
82    double lineSegArrayDis(const Point* p, int lineNum, Point&
      ↪  ret)const;
83    bool lineSegArrayDisCmp(const Point* p, int lineNum, double
      ↪  value)const;
84    Point mirror(Point& p1, Point& p2){
```

```
    Point foot = footPoint(p1, p2);
    return foot * 2 - *this;
  }

  Point rotate(double angle)const{
    Point f(sin(angle), cos(angle));
    return Point(*this * f, dot(f));
  }
  Point rotate90()const{
    return Point(-y, x);
  }
  double cosAngle(const Point& p1, const Point& p2)const{
    Point t1 = *this - p1, t2 = *this - p2;
    return t1.dot(t2) / sqrt(t1.r2()*t2.r2());
  }
  double tanAngle(const Point& o = Point(0, 0))const{
    if (EQUAL(x, o.x))return y - o.y >= 0 ? INF : -INF;
    return (y - o.y) / (x - o.x);
  }
  double angle(const Point& p1, const Point& p2)const{
    return acos(cosAngle(p1, p2));
  }
  double angle(const Point& o = Point(0, 0))const{
    return atan2(y - o.y, x - o.x);
  }
  //left return 1, right return -1, on line return 0.
  int direction(const Point& p1, const Point& p2)const{
    return SGN(x*(p1.y - p2.y) + p1.x*(p2.y - y) + p2.x*(y -
     ↪ p1.y));
  }

  bool inRect(const Point& p1, const Point& p2)const{
    return LESS_EQUAL((p1.x - x)*(p2.x - x), 0) &&
     ↪ LESS_EQUAL((p1.y - y)*(p2.y - y), 0);
  }
  int inPolygon(const Point* p, int n)const;
  int inConvex(const Point* p, int n)const;
  int inCircle(const Point& o, double r)const{
    double dist = dis2(o);
    return SGN(r*r - dist);
  }
  void pointcut(const Point& o, double r, Point& ret1, Point&
   ↪ ret2)const;
  Point nearnestPoint(const Point& o, double r)const;
};
```

```cpp
double Point::lineSegDis(const Point& p1, const Point& p2, Point& ret)const
{
  double r = lineRelation(p1, p2);
  if (LESS_EQUAL(r, 0))ret = p1;
  else if (LESS_EQUAL(1, r))ret = p2;
  else ret = footPoint(p1, p2);
  return dis(ret);
}
//input lineNum+1 points
double Point::lineSegArrayDis(const Point* p, int lineNum, Point& ret)const
{
  Point tp;
  double td, mind = INF;
  for (int i = 0; i < lineNum; i++){
    td = lineSegDis(p[i], p[i + 1], tp);
    if (LESS(td, mind)){
      mind = td; ret = tp;
    }
  }
  return mind;
}
//input lineNum+1 points
bool Point::lineSegArrayDisCmp(const Point* p, int lineNum, double value)const
{
  Point tp;
  double td;
  int flag = 1;
  for (int i = 0; i < lineNum; i++){
    td = lineSegDis(p[i], p[i + 1], tp);
    if (LESS_EQUAL(td, value))
      return true;
  }
  return false;
}

//donnot include extream points, and donnot include coincidence.
inline bool lineSegLineSegIntersect(const Point& p1, const Point& p2, const Point& q1, const Point& q2){
  Point pq1 = p1 - q1, p12 = p2 - p1, q12 = q2 - q1;
  return SGN(pq1*q12)*SGN((p2 - q1)*q12) < 0 &&
    SGN(pq1*p12)*SGN((p1 - q2)*p12) < 0;
}
//include extream points and coincidence.
```

```cpp
inline bool lineSegLineSegIntersect2(const Point& p1, const Point& p2, const Point& q1, const Point& q2){
  if (!(LESS_EQUAL(min(q1.x, q2.x), max(p1.x, p2.x)) && LESS_EQUAL(min(p1.x, p2.x), max(q1.x, q2.x))
    && LESS_EQUAL(min(q1.y, q2.y), max(p1.y, p2.y)) && LESS_EQUAL(min(p1.y, p2.y), max(q1.y, q2.y))))
    return false;
  Point pq1 = p1 - q1, p12 = p2 - p1, q12 = q2 - q1;
  return SGN(pq1*q12)*SGN((p2 - q1)*q12) <= 0 && SGN(pq1*p12)*SGN((p1 - q2)*p12) <= 0;
}
//donot include extream points, and donot include coincidence.
inline bool lineLineSegIntersect(const Point& l1, const Point& l2, const Point& p1, const Point& p2){
  Point line = l2 - l1;
  return SGN((p1 - l1)*line)*SGN((p2 - l1)*line) < 0;
}
//donnot include coincidence.
inline bool lineLineIntersect(const Point& p1, const Point& p2, const Point& q1, const Point& q2){
  return !EQUAL((p2 - p1)*(q2 - q1), 0);
}
inline Point lineLineIntersectPoint(const Point& p1, const Point& p2, const Point& q1, const Point& q2){
  Point q12 = q2 - q1;
  double k = (p2 - p1)*q12;
  if (EQUAL(k, 0))return Point(INF*INF, INF*INF);
  double r = ((q1 - p1)*q12) / k;
  return p1 + (p2 - p1) * r;
}

Point circumcenter(const Point& p1, const Point& p2, const Point& p3)
{
  Point t1 = (p1 + p2)*0.5, t2, t3 = (p2 + p3)*0.5, t4;
  t2 = t1 + (p1 - p2).rotate90();
  t4 = t3 + (p2 - p3).rotate90();
  return lineLineIntersectPoint(t1, t2, t3, t4);
}
Point incenter(const Point& p1, const Point& p2, const Point& p3)
{
  double r12 = p1.dis(p2), r23 = p2.dis(p3), r31 = p3.dis(p1);
  Point t1 = (p2*r31 + p3*r12) / (r12 + r31), t2 = (p1*r23 + p3*r12) / (r12 + r23);
  return lineLineIntersectPoint(p1, t1, p2, t2);
}
```

```
Point prepencenter(const Point& p1, const Point& p2, const Point&
 ↪  p3)
{
  Point t1 = p1 + (p2 - p3).rotate90();
  Point t2 = p2 + (p1 - p3).rotate90();
  return lineLineIntersectPoint(p1, t1, p2, t2);
}
inline Point barycenter(const Point& p1, const Point& p2, const
 ↪  Point& p3){
  return (p1 + p2 + p3) / 3;
}
inline double apothem(const Point& p1, const Point& p2, const
 ↪  Point& p3){
  Point p12 = p2 - p1, p13 = p3 - p1, p23 = p3 - p2;
  return abs(p12*p23) / (p12.r() + p13.r() + p23.r());
}
inline double circumradius(const Point& p1, const Point& p2, const
 ↪  Point& p3){
  Point p12 = p2 - p1, p13 = p3 - p1, p23 = p3 - p2;
  return sqrt(p12.r2()*p23.r2()*p13.r2()) / (2 * abs(p12*p23));
}

int getPolygonDirection(const Point* p, int n)
{
  int index = 0;
  for (int i = 1; i < n; i++){
    if (p[i] < p[index])index = i;
  }
  return p[index].direction(p[index + 1 < n ? index + 1 : 0],
   ↪  p[index - 1 >= 0 ? index - 1 : n - 1]);
}
bool checkConvex(const Point* p, int n)
{
  int direction = p[0].direction(p[n - 1], p[1]);
  if (direction == 0)return false;
  if (p[n - 1].direction(p[n - 2], p[0]) != direction)return
   ↪  false;
  for (int i = n - 2; i > 0; i--){
    if (p[i].direction(p[i - 1], p[i + 1]) != direction)
      return false;
  }
  return true;
}
bool checkConvex(const Point* p, int n, bool *ret)
{
  bool retValue = true;
```

```
245   int direction = getPolygonDirection(p, n);
246   if (!(ret[n - 1] = p[n - 1].direction(p[0], p[n - 2]) ==
      ↪  direction))
247     retValue = false;
248   if (!(ret[0] = p[0].direction(p[1], p[n - 1]) == direction))
249     retValue = false;
250   for (int i = n - 2; i > 0; i--){
251     if (!(ret[i] = p[i].direction(p[i + 1], p[i - 1]) ==
        ↪  direction))
252       retValue = false;
253   }
254   return retValue;
255 }
256 double polygonArea(const Point* p, int n)
257 {
258   double area = 0;
259   for (int i = n - 2; i > 0; i--)
260     area += p[i].y *(p[i - 1].x - p[i + 1].x);
261   area += p[0].y*(p[n - 1].x - p[1].x);
262   area += p[n - 1].y*(p[n - 2].x - p[0].x);
263   return area / 2;
264 }
265 int Point::inPolygon(const Point* p, int n)const
266 {
267   int i, j = n - 1, odd = -1;
268   for (i = 0; i < n; j = i++){
269     if (LESS(p[i].y, y) != LESS(p[j].y, y)){
270       double tx = (y - p[j].y) / (p[i].y - p[j].y)*(p[i].x -
          ↪  p[j].x) + p[j].x;
271       if (LESS_EQUAL(tx, x)){
272         if (LESS(tx, x))odd = -odd;
273         else return 0;
274       }
275     }
276     else if (onLineSeg(p[i], p[j]))return 0;
277   }
278   return odd;
279 }
280 int Point::inConvex(const Point* p, int n)const
281 {
282   int _direction = p[1].direction(p[2], p[0]);
283   if (direction(p[0], p[1]) != _direction){
284     if (onLineSeg(p[0], p[1]))return 0;
285     return -1;
286   }
287   if (direction(p[n - 1], p[0]) != _direction){
```

```
    if (onLineSeg(p[n - 1], p[0]))return 0;
    return -1;
  }
  int left = 2, right = n - 1;
  while (left < right){
    int mid = (left + right) >> 1;
    if (direction(p[0], p[mid]) == _direction)left = mid + 1;
    else right = mid;
  }
  int ret = direction(p[left-1],p[left]);
  return ret == _direction ? 1 : ret == 0 ? 0 : -1;
}
Point lineConvexIntersectPointInternal(const Point& p1, const
 ↪  Point& p2, const Point* p, int n, int start, int end)
{
  Point p12 = p2 - p1;
  if (end < start)end += n;
  double value = SGN((p[start] - p1)*p12);
  while (start + 1 < end){
    int mid = (start + end) / 2;
    Point cur = p[mid < n ? mid : mid - n];
    double t = (cur - p1)*p12*value;
    if (LESS(0, t))start = mid;
    else if (LESS(t, 0))end = mid;
    else return cur;
  }
  if (start >= n)start -= n;
  return lineLineIntersectPoint(p1, p2, p[start], p[start + 1]);
}
int lineConvexIntersectPoint(const Point& p1, const Point& p2,
 ↪  const Point* p, int n, Point& ret1, Point& ret2)
{
  Point p12 = p2 - p1;
  int pos = 0, step = n * 2 / 3;
  double d = (p[pos] - p1)*p12;
  int zero = -1, pos2 = -1;
  while (step > 1){
    step=(step + 1) / 2;
    int i = pos + step, k = pos - step;
    if (i >= n)i -= n;
    if (k < 0)k += n;
    double di = (p[i] - p1)*p12, dk = (p[k] - p1)*p12;
    if (SGN(di)*SGN(d) < 0){ pos2 = i; break; }
    if (SGN(dk)*SGN(d) < 0){ pos2 = k; break; }
    if (abs(di) < abs(d)){ d = di; pos = i; }
    if (abs(dk) < abs(d)){ d = dk; pos = k; }
```

```
332       if (EQUAL(d, 0)){ zero = pos; break; }
333     }
334     if (zero != -1){
335       ret1 = p[zero];
336       int left = zero - 1 >= 0 ? zero - 1 : n - 1;
337       int right = zero + 1 < n ? zero + 1 : 0;
338       double dl = (p[left] - p1)*p12, dr = (p[right] - p1)*p12;
339       if (EQUAL(dl, 0)){ ret2 = p[left]; return 3; }
340       else if (EQUAL(dr, 0)){ ret2 = p[right]; return 3; }
341       else if (dl*dr < 0)return 1;
342       else{ pos = left; pos2 = right; }
343     }
344     if (pos2 == -1)return 0;
345     ret1 = lineConvexIntersectPointInternal(p1, p2, p, n, pos,
        ↪ pos2);
346     ret2 = lineConvexIntersectPointInternal(p1, p2, p, n, pos2,
        ↪ pos);
347     return 2;
348   }
349
350   bool lineSegInPolygon(const Point& p1, const Point& p2, const
      ↪ Point* p, int n)
351   {
352     bool flag = false;
353     Point minPoint;
354     switch (p1.inPolygon(p, n)){
355     case -1:return false;
356     case 0:flag = true;
357     }
358     switch (p2.inPolygon(p, n)){
359     case -1:return false;
360     case 1:flag = false;
361     }
362     if (flag)minPoint = max(p1, p2);
363     for (int i = 0, j = n - 1; i < n; j = i++){
364       if (p[i].onLineSeg(p1, p2) && !(p[i] == p1 || p[i] == p2)){
365         if (p[i > 0 ? i - 1 : n - 1].direction(p1, p2) * p[i + 1 < n
            ↪ ? i + 1 : 0].direction(p1, p2) < 0)
366           return false;
367         if (flag && p[i] < minPoint)minPoint = p[i];
368       }
369       else if (lineSegLineSegIntersect(p[i], p[j], p1, p2))
370         return false;
371     }
372     if (flag){
373       const Point& t = min(p1, p2);
```

```
    Point mid = (t + minPoint)*0.5;
    if (mid.inPolygon(p, n) == -1)return false;
  }
  return true;
}
Point gravityCenter(const Point* p, int n)
{
  if (n < 3){
    if (n == 1)return p[0];
    else return (p[0] + p[1])*0.5;
  }
  double area = 0;
  Point ret(0, 0);
  for (int i = 0, j = n - 1; i < n; j = i++){
    double t = p[i] * p[j];
    area += t;
    ret.x += (p[i].x + p[j].x)*t;
    ret.y += (p[i].y + p[j].y)*t;
  }
  return ret / (3 * area);
}
//ret[n] must be available to visit.
int convexHullSorted(const Point* p, int n, Point* ret)
{
  int j = 0;
  for (int i = 0; i < n; i++){
    while (j >= 2 && p[i].direction(ret[j - 2], ret[j - 1]) !=
    ↪  1)j--;
    ret[j++] = p[i];
  }
  int mid = j + 1;
  for (int i = n - 2; i >= 0; i--){
    while (j >= mid && p[i].direction(ret[j - 2], ret[j - 1]) !=
    ↪  1)j--;
    ret[j++] = p[i];
  }
  return j - 1;
}
void convexHullSorted(const Point* p, int n, Point* up, int&
↪  retUp, Point* down, int& retDown)
{
  retUp = retDown = 0;
  for (int i = 0; i < n; i++){
    while (retUp >= 2 && p[i].direction(up[retUp - 2], up[retUp -
    ↪  1]) != -1)retUp--;
```

```
415       while (retDown >= 2 && p[i].direction(down[retDown - 2],
          ↪  down[retDown - 1]) != 1)retDown--;
416       up[retUp++] = p[i];
417       down[retDown++] = p[i];
418     }
419   }
420   int halfPlainIntersectInternal(vector<pair<double, const Point*>>&
      ↪  v, int n, Point* ret)
421   {
422     for (int i = 0; i < n; i++)
423       v[i].first = v[i].second[1].angle(v[i].second[0]);
424     sort(v.begin(), v.end());
425     vector<const Point*> line(n);
426     vector<Point> point(n);
427     int first = 0, last = 0;
428     line[0] = v[0].second;
429     for (unsigned int i = 1; i < v.size(); i++){
430       while (first < last && point[last -
          ↪  1].direction(v[i].second[0], v[i].second[1]) == -1)
          ↪  last--;
431       while (first < last && point[first].direction(v[i].second[0],
          ↪  v[i].second[1]) == -1) first++;
432       line[++last] = v[i].second;
433       if (!lineLineIntersect(line[last - 1][0], line[last - 1][1],
          ↪  line[last][0], line[last][1])){
434         last--;
435         if (v[i].second[0].direction(line[last][0], line[last][1])
            ↪  == 1)line[last] = v[i].second;
436       }
437       if (first<last)
438         point[last - 1] = lineLineIntersectPoint(line[last - 1][0],
            ↪  line[last - 1][1], line[last][0], line[last][1]);
439     }
440     while (first < last && point[last - 1].direction(line[first][0],
      ↪  line[first][1]) == -1) last--;
441     if (last - first <= 1) return 0;
442     point[last] = lineLineIntersectPoint(line[first][0],
      ↪  line[first][1], line[last][0], line[last][1]);
443     int num = unique(&*point.begin() + first, &*point.begin() + last
      ↪  + 1) - &point[first];
444     while (num>1 && point[first] == point[first + num - 1])num--;
445     memcpy(ret, &point[first], sizeof(Point)*num);
446     return num;
447   }
448   int halfPlainIntersect(const Point(*p)[2], int n, Point* ret)
449   {
```

```
450   vector<pair<double, const Point*>> v(n + 4);
451   Point ext[4][2] = { { { -INF, -INF }, { INF, -INF } }, { { INF,
      ↪  -INF }, { INF, INF } },
452   { { INF, INF }, { -INF, INF } }, { { -INF, INF }, { -INF, -INF }
      ↪  } };
453   for (int i = 0; i < 4; i++)
454     v[i].second = ext[i];
455   for (int i = 0; i < n; i++)
456     v[i + 4].second = p[i];
457   return halfPlainIntersectInternal(v, n + 4, ret);
458 }
459 int polygonKernel(const Point* p, int n, Point* ret)
460 {
461   vector<pair<double, const Point*>> v;
462   Point ext[2] = { p[n - 1], p[0] };
463   v[0].second = ext;
464   for (int i = 1; i < n; i++)
465     v[i].second = &p[i - 1];
466   return halfPlainIntersectInternal(v, n, ret);
467 }
468
469 struct NearestPointsStruct{
470   Point p1, p2;
471   double d2;
472   vector<Point> v;
473 };
474 inline bool nearestPointsCmp(const Point& p1, const Point& p2){
475   return LESS_EQUAL(p1.y, p2.y) && (LESS(p1.y, p2.y) || LESS(p1.x,
      ↪  p2.x));
476 }
477 void nearestPointsInternal(const Point* p, int left, int right,
      ↪  NearestPointsStruct& s)
478 {
479   if (right - left < 8){
480     for (int i = left; i < right; i++){
481       for (int j = i + 1; i < right; j++){
482         double td2 = p[j].dis2(p[i]);
483         if (td2 < s.d2){
484           s.d2 = td2;
485           s.p1 = p[i]; s.p2 = p[j];
486         }
487       }
488     }
489     return;
490   }
491   int mid = (left + right) >> 1;
```

```
492   nearestPointsInternal(p, left, mid, s);
493   nearestPointsInternal(p, mid, right, s);
494   s.v.clear();
495   double l = (p[mid - 1].x + p[mid].x) / 2;
496   for (int i = mid - 1; i >= left && (p[i].x - l)*(p[i].x - l) <
      ↪  s.d2; i++)
497     s.v.push_back(p[i]);
498   for (int i = mid; i<right && (p[i].x - l)*(p[i].x - l) < s.d2;
      ↪  i++)
499     s.v.push_back(p[i]);
500   sort(s.v.begin(), s.v.end(), nearestPointsCmp);
501   for (unsigned int i = 0; i < s.v.size(); i++){
502     for (unsigned int j = i + 1; j < s.v.size() && (p[j].y -
        ↪  p[i].y)*(p[j].y - p[i].y) < s.d2; j++){
503       double td2 = p[j].dis2(p[i]);
504       if (td2 < s.d2){
505         s.d2 = td2;
506         s.p1 = p[i]; s.p2 = p[j];
507       }
508     }
509   }
510 }
511 double nearestPointsSorted(const Point* p, int n, Point& ret1,
    ↪  Point& ret2)
512 {
513   NearestPointsStruct s;
514   s.d2 = INF;
515   s.v.reserve(n);
516   nearestPointsInternal(p, 0, n, s);
517   ret1 = s.p1; ret2 = s.p2;
518   return sqrt(s.d2);
519 }
520 double farthestPointsConvex(const Point* p, int n, Point& ret1,
    ↪  Point& ret2)
521 {
522   double d2 = 0;
523   for (int i = n - 1, j = n - 2; i >0; i--){
524     while (1){
525       double td2 = p[i].dis2(p[j]);
526       if (td2 > d2){
527         d2 = td2;
528         ret1 = p[i]; ret2 = p[j];
529       }
530       if (!j)break;
531       j--;
532     }
```

```
533     }
534     return sqrt(d2);
535   }
536   double farthestPointsSorted(const Point* p, int n, Point& ret1,
      ↪  Point& ret2)
537   {
538     vector<Point> v;
539     v.reserve(n);
540     //convexHullSorted(p, n, &*v.begin());
541     return farthestPointsConvex(&*v.begin(), v.size(), ret1, ret2);
542   }
543
544   int circleLineRelation(const Point& o, double r, const Point& p1,
      ↪  const Point& p2)
545   {
546     double d = o.lineDis(p1, p2);
547     if (LESS(d, r))return 1;
548     if (LESS(r, d))return 3;
549     return 2;
550   }
551   int circleCircleRelation(const Point& o1, double r1, const Point&
      ↪  o2, double r2)
552   {
553     double r = o1.dis(o2);
554     if (LESS(r1 + r2, r))return 4;
555     if (!LESS_EQUAL(r1 + r2, r))return 3;
556     double sub = abs(r1 - r2);
557     if (LESS(sub, r))return 2;
558     if (!LESS_EQUAL(sub, r))return 1;
559     return 0;
560   }
561   bool circleLineSegIntersect(const Point& o, double r, const Point&
      ↪  p1, const Point& p2)
562   //include extream points.
563   {
564     int t1 = p1.inCircle(o, r), t2 = p2.inCircle(o, r);
565     if (t1 >= 0 || t2 >= 0)
566       return t1 != 1 || t2 != 1;
567     double t = o.lineRelation(p1, p2);
568     if (t >= 1 || t <= 0)return false;
569     Point foot = p1 + (p2 - p1)*r;
570     return foot.inCircle(o, r) >= 0;
571   }
572   void circleLineIntersect(const Point& o, double r, const Point&
      ↪  p1, const Point& p2, Point& ret1, Point& ret2)
573   {
```

```
574   Point foot = o.footPoint(p1, p2);
575   double t = sqrt((r*r - o.dis2(foot)) / p1.dis2(p2));;
576   ret1 = foot + (p2 - p1)*t;
577   ret2 = foot * 2 - ret1;
578 }
579 void circleCircleIntersect(const Point& o1, double r1, const
    ↪  Point& o2, double r2, Point& ret1, Point& ret2)
580 {
581   double d2 = o1.dis2(o2);
582   double t1 = (r1*r1 - r2*r2) / (2 * d2) + 0.5;
583   double t2 = sqrt(r1*r1 / d2 - t1*t1);
584   Point foot = o1 + (o2 - o1)*t1;
585   ret1 = foot + (o2 - o1).rotate90()*t2;
586   ret2 = foot * 2 - ret1;
587 }
588 void Point::pointcut(const Point& o, double r, Point& ret1, Point&
    ↪  ret2)const
589 {
590   double t1 = r*r / dis2(o);
591   Point foot = o + (o - *this)*t1;
592   double t2 = sqrt(t1 - t1*t1);
593   ret1 = foot + (*this - o).rotate90()*t2;
594   ret2 = foot * 2 - ret1;
595 }
596 Point Point::nearnestPoint(const Point& o, double r)const
597 {
598   Point p = *this - o;
599   double d = p.r();
600   if (EQUAL(d, 0))return o;
601   return o + p*(r / d);
602 }
603 //Upset the order before using this function.
604 double minCoveringCircle(const Point* p, int n, Point& ret)
605 {
606   if (n == 1){ ret = p[0]; return 0; }
607   double r2 = p[0].dis2(p[1]);
608   ret = (p[0] + p[1]) * 0.5;
609   for (int i = 2; i < n; i++){
610     if (LESS(r2, ret.dis2(p[i]))){
611       ret = (p[0] + p[i]) * 0.5;
612       r2 = p[0].dis2(p[i]);
613       for (int j = 1; j < i; j++){
614         if (LESS(r2, ret.dis2(p[j]))){
615           ret = (p[i] + p[j]) * 0.5;
616           r2 = p[i].dis2(p[j]);
617           for (int k = 0; k < j; k++){
```

```
618               if (LESS(r2, ret.dis2(p[k]))){
619                 ret = circumcenter(p[i], p[j], p[k]);
620                 r2 = ret.dis2(p[k]);
621               }
622             }
623           }
624         }
625       }
626     }
627     return sqrt(r2);
628   }
629   int unitCoveringCircle(const Point* p, int n, double r)
630   {
631     int ret = 0;
632     vector<pair<double, bool>> v;
633     v.reserve(2 * n);
634     double t = r*r * 4;
635     for (int i = 0; i < n; i++){
636       v.clear();
637       int value = 0;
638       for (int j = 0; j < n; j++){
639         if (LESS_EQUAL(p[i].dis2(p[j]), t) && i != j){
640           double a = p[j].angle(p[i]);
641           double b = acos(p[i].dis(p[j]) / r / 2);
642           double t1 = a - b, t2 = a + b;
643           if (t1 < -PI / 2){
644             if (t2 < -PI / 2){
645               a += 2 * PI;
646               b += 2 * PI;
647             }
648             else value++;
649           }
650           v.push_back(make_pair(t1, true));
651           v.push_back(make_pair(t2, false));
652         }
653       }
654       sort(v.begin(), v.end());
655       if (value > ret)ret = value;
656       for (unsigned int j = 0; j < v.size(); j++){
657         if (v[j].second){
658           value++;
659           if (value > ret)ret = value;
660         }
661         else value--;
662       }
663     }
```

```
664     return ret;
665   }
```

2 Data Structures

2.1 Cartesian Tree

```
/*
@title: Cartesian Tree 笛卡尔树
@description:
    Cartesian Tree 笛卡尔树
    可以实现线性时间内建立具有 BST 性质的树
@structure:
    CartesianTreeNode:
        parent:   父指针
        l: 左孩子指针
        r: 右孩子指针
@arguments:
    BuildFromArray:
        value: 源数组
        N: 数组大小
        index: 源数组的逆映射数组
        tree: 目标建树数组内存首地址
        stack: 堆栈空间
@performance:
    BuildFromArray:
        Time: O(N)
        Space: O(N)
@dependence: null
@range:
    for i in [0, N)
    value[i] in [0, N)
    index[i] in [0, N)
    |value| = |index| = |tree| = |stack| = N
@note:
    value 与 index 互为逆映射故满足双射性质
        index[value[i]] == i
        value[index[i]] == i
    index 无须在函数外初始化，建树过程可以计算 index
    stack 无须在函数外初始化，但建树过程对 stack 有污染
    最后结束迭代的时候栈底一定为 value[0]
    笛卡尔树的树根一定为 value[0]
    因此笛卡尔树的 parent 不一定要保存，仅保存孩子指针也可以完成遍历
*/

struct CartesianTreeNode {
  int parent, left, right;
};
```

```
42
43  void BuildFromArray(int *value, int N, int *index,
    ↪  CartesianTreeNode *tree,
44                      int *stack) {
45    // 计算逆映射
46    for (int i = 0; i < N; i++) {
47      index[value[i]] = i;
48    }
49    // 初始化节点
50    for (int i = 0; i < N; i++) {
51      tree[i].parent = tree[i].left = tree[i].right = -1;
52    }
53    int size = 0; // 初始化清空栈
54    for (int i = 0; i < N; i++) {
55      int nextSize = size;
56      // 维护单调栈
57      while (nextSize > 0 && index[stack[nextSize - 1]] > index[i])
        ↪  {
58        nextSize--;
59      }
60      // 下面两个 if 语句块的顺序可变
61      if (nextSize > 0) { // 栈中有元素
62        // 当前元素为栈顶元素的右孩子
63        int top = stack[nextSize - 1];
64        tree[i].parent = top;
65        tree[top].right = i;
66      }
67      if (nextSize < size) { // 弹过栈
68        // 最后出栈的元素为当前元素的左孩子
69        int lastPop = stack[nextSize];
70        tree[lastPop].parent = i;
71        tree[i].left = lastPop;
72      }
73      stack[nextSize++] = i; // 入栈
74      size = nextSize;       // 更新栈大小
75    }
76  }
```

2.2 Heap

```cpp
#include<cstdio>
#include<cstring>
#include<algorithm>
using namespace std;
template<typename T>
struct Heap{
  T a[1000002];
  int n;
  Heap(){ clear(); }
  void down(int i){
    for (int j = i << 1; j <= n; i = j, j <<= 1){
      if (a[j + 1] < a[j])j++;
      if (a[j] < a[i])swap(a[i], a[j]);
      else break;
    }
  }
  void build(T src[], int n){
    memcpy(a + 1, src, n*sizeof(T));
    this->n = n;
    for (int i = n >> 1; i; i--){
      a[n + 1] = a[n];
      down(i);
    }
  }
  void push(T value){
    a[++n] = value;
    for (int j = n, i = j >> 1; i && a[j] < a[i]; j = i, i >>= 1)
      swap(a[i], a[j]);
  }
  void pop(){
    a[1] = a[n--];
    down(1);
  }
  T top()const{ return a[1]; }
  bool empty()const{ return n > 0; }
  void clear(){ n = 0; }
};
```

2.3 mergeable heap

```
int n;            //点数
struct node
{
  int v,dis;
  node *l,*r;
}mem[maxn],*head[maxn];
int cnt;
node* merge(node* a,node* b)
{
  if (a==mem) return b;
  if (b==mem) return a;
  if (a->v<b->v) swap(a,b);
  a->r=merge(a->r,b);
  if (a->r->dis>a->l->dis) swap(a->l,a->r);
  if (a->r==mem) a->dis=0;
  else a->dis=a->r->dis+1;
  return a;
}
void init()
{
  mem[0].dis=-1;
  mem[0].l=mem[0].r=mem;
  for (int i=1;i<=n;i++)
  {
    mem[i].l=mem[i].r=mem;
    head[i]=mem+i;
  }
}
//BZOJ 2809
int m;
queue<int> q;
int f[maxn],c[maxn],l[maxn],ind[maxn],ths[maxn];
long long cost[maxn],ans;
int main()
{
  freopen("in.txt","r",stdin);
  freopen("out.txt","w",stdout);
  scanf("%d%d",&n,&m);
  init();
  for (int i=1;i<=n;i++)
  {
    scanf("%d%d%d",&f[i],&c[i],&l[i]);
    ind[f[i]]++;
  }
```

```
45    for (int i=1;i<=n;i++)
46    {
47      if (ind[i]==0) q.push(i);
48      ths[i]=1;cost[i]=c[i];
49      mem[i].v=c[i];
50    }
51    while(!q.empty())
52    {
53      int now=q.front();q.pop();
54      while(cost[now]>m)
55      {
56        cost[now]-=head[now]->v;
57        head[now]=merge(head[now]->l,head[now]->r);
58        ths[now]--;
59      }
60      ans=max(ans,1ll*l[now]*ths[now]);
61      if (f[now]!=0)
62      {
63        head[f[now]]=merge(head[f[now]],head[now]);
64        ths[f[now]]+=ths[now];
65        cost[f[now]]+=cost[now];
66        if ((--ind[f[now]])==0) q.push(f[now]);
67      }
68    }
69    printf("%lld",ans);
70    return 0;
71  }
```

2.4 Segment Tree

```
struct tree
{
  int mi,ls,rs,ll,rr,add;
}a[3*maxn];

int c[maxn];
int i,j,k,l,m,n,t,T;

void update(int x)
{
  if(a[x].ls+a[x].rs==0)return;
  a[x].mi=min(a[a[x].ls].mi,a[a[x].rs].mi);
}

void downdate(int x)
{
  if(a[x].ls+a[x].rs==0)return;
  if(a[x].add==0)return;
  a[a[x].ls].add+=a[x].add;
  a[a[x].ls].mi+=a[x].add;
  a[a[x].rs].add+=a[x].add;
  a[a[x].rs].mi+=a[x].add;
  a[x].add=0;
}

void mt(int l,int r)
{
  if(l==r)
  {
    a[k].ll=a[k].rr=l;
    a[k].mi=c[l];
    a[k].ls=a[k].rs=0;
    return;
  }
  int t=k;
  int mid=(l+r)>>1;
  a[t].ll=l;a[t].rr=r;
  k++;a[t].ls=k;mt(l,mid);
  k++;a[t].rs=k;mt(mid+1,r);
  update(t);
}

void add(int l,int r,int nu,int x)
{
```

```
45    if(a[x].ll==l && a[x].rr==r)
46    {
47      a[x].add+=nu;
48      a[x].mi+=nu;
49      return;
50    }
51    downdate(x);
52    int mid=(a[x].ll+a[x].rr)>>1;
53    if(mid<l)add(l,r,nu,a[x].rs);
54    else if(mid>=r)add(l,r,nu,a[x].ls);
55    else
56    {
57      add(l,mid,nu,a[x].ls);
58      add(mid+1,r,nu,a[x].rs);
59    }
60    update(x);
61  }
62
63  int find(int l,int r,int x)
64  {
65    if(a[x].ll==l && a[x].rr==r)return a[x].mi;
66    downdate(x);
67    update(x);
68    int mid=(a[x].ll+a[x].rr)>>1;
69    if(mid<l)return find(l,r,a[x].rs);
70    if(mid>=r)return find(l,r,a[x].ls);
71    return min(find(l,mid,a[x].ls),find(mid+1,r,a[x].rs));
72  }
73
74  inline void read(int &x) {
75      char ch=getchar();
76      while(ch<'0'||ch>'9') ch=getchar();
77      x=0;
78      while(ch<='9'&&ch>='0'){
79          x=x*10+ch-'0';
80          ch=getchar();
81      }
82  }
83
84  int main()//read 不能读负数!!!!!!!
85  {
86    read(n);read(m);
87    for(i=1;i<=n;i++)read(c[i]);
88    k=1;mt(1,n);
89    for(i=1;i<=m;i++)
```

```
90      {
91        read(l);read(j);read(k);
92        if(find(j,k,1)<l)
93        {
94          printf("-1\n%d",i);
95          break;
96        }
97        add(j,k,-l,1);
98      }
99      if(i>m)printf("0");
100     return 0;
101   }
```

2.5 Splay

```
struct tree
{
  int key,size,le,ri,add,rev,min,pre;
}a[maxn];
int n,T,node;
int s[maxn];

void pushdown(int cur)
{
  int ls=a[cur].le,rs=a[cur].ri;
  if(a[cur].add>0)
  {
    a[ls].add+=a[cur].add;
    a[rs].add+=a[cur].add;
    a[ls].key+=a[cur].add;
    a[rs].key+=a[cur].add;
    a[ls].min+=a[cur].add;
    a[rs].min+=a[cur].add;
    a[cur].add=0;
  }
  if(a[cur].rev>0)
  {
    a[ls].rev^=1;
    a[rs].rev^=1;
    a[cur].le=rs;
    a[cur].ri=ls;
    a[cur].rev=0;
  }
}

void update(int cur)
{
  int ls=a[cur].le,rs=a[cur].ri;
  a[cur].size=a[ls].size+a[rs].size+1;
  a[cur].min=a[cur].key;
  if(ls&&a[ls].min<a[cur].min)a[cur].min=a[ls].min;
  if(rs&&a[rs].min<a[cur].min)a[cur].min=a[rs].min;
}

void leftrotate(int x)
{
  int y=a[x].ri,p=a[x].pre;
  a[x].ri=a[y].le;
  if(a[x].ri)a[a[x].ri].pre=x;
```

```
45    a[y].le=x;
46    a[x].pre=y;
47    a[y].pre=p;
48    if(!p)T=y;
49    else
50      a[p].ri==x?a[p].ri=y:a[p].le=y;
51    update(x);
52  }
53
54  void rightrotate(int x)
55  {
56    int y=a[x].le,p=a[x].pre;
57    a[x].le=a[y].ri;
58    if(a[x].le)a[a[x].le].pre=x;
59    a[y].ri=x;
60    a[x].pre=y;
61    a[y].pre=p;
62    if(!p)T=y;
63    else
64      a[p].ri==x?a[p].ri=y:a[p].le=y;
65    update(x);
66  }
67
68  void splay(int x,int goal)
69  {
70    int y,z;
71    while(1)
72    {
73      if((y=a[x].pre)==goal)break;
74      if((z=a[y].pre)==goal)
75        a[y].ri==x?leftrotate(y):rightrotate(y);
76      else
77      {
78        if(a[z].ri==y)
79        {
80          if(a[y].ri==x)
81            leftrotate(z),leftrotate(y);
82          else
83            rightrotate(y),leftrotate(z);
84        }
85        else
86        {
87          if(a[y].le==x)
88            rightrotate(z),rightrotate(y);
89          else
90            leftrotate(y),rightrotate(z);
```

```
      }
    }
  }
  update(x);
}

void rotateto(int k,int goal)
{
  int i=T;
  while(1)
  {
    pushdown(i);
    if(a[a[i].le].size+1==k)break;
    if(k<=a[a[i].le].size)i=a[i].le;
    else k-=a[a[i].le].size+1,i=a[i].ri;
  }
  splay(i,goal);
}

void newnode(int &cur,int v)
{
  cur=++node;
  a[cur].min=a[cur].key=v;
  a[cur].size=1;
  a[cur].le=a[cur].ri=a[cur].rev=a[cur].add=0;
}

void build(int &cur,int x,int y,int p)
{
  int mid=(x+y)>>1;
  newnode(cur,s[mid]);
  a[cur].pre=p;
  if(x==y)return;
  if(x<mid)build(a[cur].le,x,mid-1,cur);
  if(y>mid)build(a[cur].ri,mid+1,y,cur);
  update(cur);
}

void init(int n)
{
  int i;
  memset(s,0,sizeof(s));
  memset(a,0,sizeof(a));
  for(i=1;i<=n;i++)scanf("%d",&s[i]);
  T=node=0;
  build(T,0,n+1,0);
```

```
}

void Add(int x,int y,int z)
{
  int k;
  rotateto(x,0);rotateto(y+2,T);
  k=a[a[T].ri].le;
  a[k].add+=z;a[k].key+=z;a[k].min+=z;
}

void Reverse(int x,int y)
{
  int k;
  rotateto(x,0);rotateto(y+2,T);
  k=a[a[T].ri].le;
  a[k].rev^=1;
}

void Revolve(int x,int y,int z)
{
  int k=z%(y-x+1),t;
  if(k)
  {
    rotateto(x,0);rotateto(y-k+2,T);
    t=a[a[T].ri].le;
    a[a[T].ri].le=0;
    update(a[T].ri);update(T);
    rotateto(x+k,0);rotateto(x+k+1,T);
    a[a[T].ri].le=t;a[t].pre=a[T].ri;
    update(a[T].ri);update(T);
  }
}

void Insert(int x,int y)
{
  rotateto(x+1,0);rotateto(x+2,T);
  newnode(a[a[T].ri].le,y);
  a[a[a[T].ri].le].pre=a[T].ri;
  update(a[T].ri);update(T);
}

void Delete(int x)
{
  rotateto(x,0);rotateto(x+2,T);
  a[a[T].ri].le=0;
  update(a[T].ri);update(T);
```

```
183 }
184
185 void Min(int x,int y)
186 {
187   rotateto(x,0);rotateto(y+2,T);
188   printf("%d\n",a[a[a[T].ri].le].min);
189 }
```

2.6 Treap

```
struct data{
    int l,r,v,size,rnd,w;
}tr[100005];
int n,size,root,ans;

void update(int k)//更新结点信息
{
    tr[k].size=tr[tr[k].l].size+tr[tr[k].r].size+tr[k].w;
}

void rturn(int &k)
{
    int t=tr[k].l;tr[k].l=tr[t].r;tr[t].r=k;
    tr[t].size=tr[k].size;update(k);k=t;
}

void lturn(int &k)
{
    int t=tr[k].r;tr[k].r=tr[t].l;tr[t].l=k;
    tr[t].size=tr[k].size;update(k);k=t;
}

void insert(int &k,int x)
{
    if(k==0)
    {
        size++;k=size;
        tr[k].size=tr[k].w=1;tr[k].v=x;tr[k].rnd=rand();
        return;
    }
    tr[k].size++;
    if(tr[k].v==x)tr[k].w++;
    else if(x>tr[k].v)
    {
        insert(tr[k].r,x);
        if(tr[tr[k].r].rnd<tr[k].rnd)lturn(k);
    }
    else
    {
        insert(tr[k].l,x);
        if(tr[tr[k].l].rnd<tr[k].rnd)rturn(k);
    }
}

```

```
45  void del(int &k,int x)
46  {
47      if(k==0)return;
48      if(tr[k].v==x)
49      {
50          if(tr[k].w>1)
51          {
52              tr[k].w--;tr[k].size--;return;
53          }
54          if(tr[k].l*tr[k].r==0)k=tr[k].l+tr[k].r;
55          else if(tr[tr[k].l].rnd<tr[tr[k].r].rnd)
56              rturn(k),del(k,x);
57          else lturn(k),del(k,x);
58      }
59      else if(x>tr[k].v)
60          tr[k].size--,del(tr[k].r,x);
61      else tr[k].size--,del(tr[k].l,x);
62  }
63
64  int query_rank(int k,int x)
65  {
66      if(k==0)return 0;
67      if(tr[k].v==x)return tr[tr[k].l].size+1;
68      else if(x>tr[k].v)
69          return tr[tr[k].l].size+tr[k].w+query_rank(tr[k].r,x);
70      else return query_rank(tr[k].l,x);
71  }
72
73  int query_num(int k,int x)
74  {
75      if(k==0)return 0;
76      if(x<=tr[tr[k].l].size)
77          return query_num(tr[k].l,x);
78      else if(x>tr[tr[k].l].size+tr[k].w)
79          return query_num(tr[k].r,x-tr[tr[k].l].size-tr[k].w);
80      else return tr[k].v;
81  }
82
83  void query_pro(int k,int x)
84  {
85      if(k==0)return;
86      if(tr[k].v<x)
87      {
88          ans=k;query_pro(tr[k].r,x);
89      }
90      else query_pro(tr[k].l,x);
```

```
}

void query_sub(int k,int x)
{
    if(k==0)return;
    if(tr[k].v>x)
    {
        ans=k;query_sub(tr[k].l,x);
    }
    else query_sub(tr[k].r,x);
}

int main()
{
    scanf("%d",&n);
    int opt,x;
    for(int i=1;i<=n;i++)
    {
        scanf("%d%d",&opt,&x);
        switch(opt)
        {
        case 1:insert(root,x);break;
        case 2:del(root,x);break;
        case 3:printf("%d\n",query_rank(root,x));break;
        case 4:printf("%d\n",query_num(root,x));break;
        case
        ↪  5:ans=0;query_pro(root,x);printf("%d\n",tr[ans].v);break;
        case
        ↪  6:ans=0;query_sub(root,x);printf("%d\n",tr[ans].v);break;
        }
    }
    return 0;
}
```

2.7 Tree Cut(edge)

```cpp
#include<cstdio>
#include<vector>
using namespace std;
vector<pair<int, int>> v[200001];//边及该边的编号
int w[200001];//边权
int n, cnt;
int father[200001], depth[200001], top[200001], id[200001];
int f[200001];//边在树状数组（线段树）中的位置
int tmp[200001];
int dfs1(int i, int fa)
{
  father[i] = fa;
  depth[i] = depth[fa] + 1;
  tmp[i] = -1;
  int ret = 0, maxSize = 0;
  for (unsigned int j = 0; j < v[i].size(); j++){
    int t = v[i][j].first;
    if (t == fa)continue;
    int size = dfs1(t, i);
    ret += size;
    if (size > maxSize){
      maxSize = size;
      tmp[i] = j;
    }
  }
  return ret + 1;
}
void dfs2(int i, int tp, int index)
{
  top[i] = tp;
  id[i] = cnt;
  f[index] = cnt++;
  if (tmp[i] != -1)
    dfs2(v[i][tmp[i]].first, tp, v[i][tmp[i]].second);
  for (unsigned int j = 0; j < v[i].size(); j++){
    int t = v[i][j].first;
    if (t != father[i] && j != tmp[i])
      dfs2(t, t, v[i][j].second);
  }
}
int queryTree(int s, int t)
{
  int ret = 0;
  int top1 = top[s], top2 = top[t];
```

```
  while (top1 != top2){
    if (depth[top1] < depth[top2]){
      ret += sum(id[t]) - sum(id[top2] - 1);
      t = father[top2]; top2 = top[t];
    }
    else{
      ret += sum(id[s]) - sum(id[top1] - 1);
      s = father[top1]; top1 = top[s];
    }
  }
  if (s != t){
    if (depth[s] > depth[t])swap(s, t);
    ret += sum(id[t]) - sum(id[s]);
  }
  return ret;
}
void init()
{
  cnt = 0;
  dfs1(1, 1);
  dfs2(1, 1, 0);
  for (int i = 1; i < n; i++)
    tree[f[i]] = w[i];
  build();
}
int main()
{
  int q, cur;
  scanf("%d%d%d", &n, &q, &cur);
  for (int i = 1; i < n; i++){
    int s, t, value;
    scanf("%d%d%d", &s, &t, &value);
    v[s].push_back(make_pair(t, i));
    v[t].push_back(make_pair(s, i));
    w[i] = value;
  }
  init();
  for (int i = 0; i < q; i++){
    int t, u, value;
    scanf("%d%d", &t, &u);
    if (t == 0){
      printf("%d\n", queryTree(cur, u));
      cur = u;
    }
    else{
      scanf("%d", &value);
```

```
91          add(f[u], value - w[u]);
92          w[u] = value;
93        }
94      }
95      return 0;
96    }
```

2.8 Tree Cut(vertex)

```cpp
#include<cstdio>
#include<cstring>
#include<vector>
using namespace std;
vector<int> v[100001];
int n, cnt, color;
int father[100001], depth[100001], top[100001], id[100001],
    son[100001];
struct Tree{
  int maxValue, maxId, delta;
  bool set;
}tree[1 << 18];
int treeLen;
int dfs1(int i, int fa)
{
  father[i] = fa;
  depth[i] = depth[fa] + 1;
  son[i] = 0;
  int ret = 0, maxSize = 0;
  for (unsigned int j = 0; j < v[i].size(); j++){
    int t = v[i][j];
    if (t == fa)continue;
    int size = dfs1(t, i);
    ret += size;
    if (size > maxSize){
      maxSize = size;
      son[i] = t;
    }
  }
  return ret + 1;
}
void dfs2(int i, int tp)
{
  top[i] = tp;
  id[i] = cnt++;
  if (son[i])dfs2(son[i], tp);
  for (unsigned int j = 0; j < v[i].size(); j++){
    int t = v[i][j];
    if (t != father[i] && t != son[i])dfs2(t, t);
  }
}
void init()
{
  cnt = 0; depth[1] = 0;
```

```
  dfs1(1, 1);
  dfs2(1, 1);
  for (treeLen = 1; treeLen < n; treeLen *= 2);
  memset(tree, 0, sizeof(Tree) * 2 * treeLen);
}
void pushDown(int i)
{
  if (tree[i].set){
    tree[2 * i].set = tree[2 * i + 1].set = true;
    tree[2 * i].delta = tree[2 * i + 1].delta = tree[i].delta;
    tree[i].delta = 0; tree[i].set = false;
  }
  else if (tree[i].delta){
    tree[2 * i].delta += tree[i].delta;
    tree[2 * i + 1].delta += tree[i].delta;
    tree[i].delta = 0;
  }
}
int queryL, queryR;
void addInternal(int i, int l, int len)
{
  if (queryL <= l && queryR >= l + len){
    tree[i].delta++;
    return;
  }
  len >>= 1; pushDown(i);
  int mid = l + len;
  if (mid > queryL)addInternal(2 * i, l, len);
  if (mid < queryR)addInternal(2 * i + 1, mid, len);
}
inline void addValue(int l, int r){
  queryL = l; queryR = r;
  addInternal(1, 0, treeLen);
}
int addTree(int s, int t)
{
  int ret = 0;
  int top1 = top[s], top2 = top[t];
  while (top1 != top2){
    if (depth[top1] < depth[top2]){
      addValue(id[top2], id[t] + 1);
      t = father[top2]; top2 = top[t];
    }
    else{
      addValue(id[top1], id[s] + 1);
      s = father[top1]; top1 = top[s];
```

```
    }
  }
  if (depth[s] > depth[t])swap(s, t);
  addValue(id[s], id[t] + 1);
  return ret;
}
void process(int i)
{
  if (tree[i].set){
    if (tree[i].delta > tree[i].maxValue){
      tree[i].maxValue = tree[i].delta;
      tree[i].maxId = color;
    }
    return;
  }
  pushDown(i);
  process(2 * i);
  process(2 * i + 1);
}
inline void pushDownColor(int i, int j){
  if (tree[j].maxValue < tree[i].maxValue
    || (tree[j].maxValue == tree[i].maxValue && tree[i].maxId <
    ↪  tree[j].maxId)){
    tree[j].maxValue = tree[i].maxValue;
    tree[j].maxId = tree[i].maxId;
  }
}
void getAns(int i)
{
  if (i < treeLen){
    pushDownColor(i, 2 * i);
    pushDownColor(i, 2 * i + 1);
    getAns(2 * i);
    getAns(2 * i + 1);
  }
}
vector<pair<int, int>> z[100001];
int main()
{
  int m;
  while (scanf("%d%d", &n, &m) == 2 && n){
    for (int i = 1; i <= n; i++)
      v[i].clear();
    for (int i = 1; i < n; i++){
      int s, t;
      scanf("%d%d", &s, &t);
```

```
135         v[s].push_back(t);
136         v[t].push_back(s);
137       }
138       for (int i = 0; i < m; i++){
139         int s, t, w;
140         scanf("%d%d%d", &s, &t, &w);
141         z[w].push_back(make_pair(s, t));
142       }
143       init();
144       for (color = 1; color <= 100000; color++){
145         tree[1].set = true; tree[1].delta = 0;
146         for (unsigned int j = 0; j < z[color].size(); j++)
147           addTree(z[color][j].first, z[color][j].second);
148         process(1);
149         z[color].clear();
150       }
151       getAns(1);
152       for (int i = 1; i <= n; i++)
153         printf("%d\n", tree[treeLen + id[i]].maxId);
154     }
155     return 0;
156 }
```

3 Graph

3.1 (Euler)Fluery

```
//起点很重要
//stack 栈深度会超过点数，最大为边数
const int maxn=10005;
int stac[maxn],sta;
struct edge
{
  int p,q;
  edge(int p=0,int q=0):p(p),q(q){}
}edg[2*maxn];
bool used[2*maxn];
vector<int> g[maxn];
int du[maxn];
int i,j,k,l,m,n;

int other(int num,int x)
{
  return x==edg[num].p?edg[num].q:edg[num].p;
}

void dfs(int x)
{
  stac[++sta]=x;
  for(unsigned int i=0;i<g[x].size();i++)
  {
    if(used[g[x][i]])continue;
    used[g[x][i]]=true;
    dfs(other(g[x][i],x));
    break;
  }
}

void Fleury(int x)
{
  sta=1;stac[sta]=x;
  while(sta>=1)
  {
    x=stac[sta];
    bool f=false;
    for(unsigned int i=0;i<g[x].size();i++)
    {
      if(!used[g[x][i]]){f=true;break;}
```

```
42      }
43      if(!f)printf("%d ",stac[sta--]);
44      else
45      {
46        sta--;
47        dfs(stac[sta+1]);
48      }
49    }
50  }
51
52  int main()
53  {
54    //未判断图是否连通，为严谨可加入 bfs 检查
55    scan2(n,m);
56    memset(edg,0,sizeof(edg));
57    for(i=1;i<=n;i++)g[i].clear();
58    memset(used,0,sizeof(used));
59    memset(du,0,sizeof(du));
60    for(i=1;i<=m;i++)
61    {
62      scan2(j,k);
63      edg[i]=edge(j,k);
64      du[j]++;du[k]++;
65      g[j].push_back(i);
66      g[k].push_back(i);
67    }
68    int tot=0,st=0;
69    for(i=1;i<=n;i++)
70    {
71      if(du[i]==0){tot=3;break;}
72      if(du[i]%2==1){tot++;st=i;}
73    }
74    if(st==0)st=1;
75    if(tot>2)printf("HeHeDa!");else Fleury(st);
76      return 0;
77  }
```

3.2 0-1 分数规划

```
const int maxn=1005;
double a[maxn],b[maxn];
struct hei
{
  double num;
  int pos;
  hei(double num=0,int pos=0):num(num),pos(pos){}
  bool operator < (struct hei p)const
  {return num>p.num;}
}d[maxn];
double p,q,ans,l;
int i,n,m;

int main()
{
  while(scan2(n,m)==2 && n+m>0)
  {
    m=n-m;
    for(i=1;i<=n;i++)scanf("%lf",&a[i]);
    for(i=1;i<=n;i++)scanf("%lf",&b[i]);
    l=0;
    while(true)
    {
      ans=l;
      for(i=1;i<=n;i++)d[i]=hei(a[i]-ans*b[i],i);
      sort(d+1,d+n+1);
      double fz,fm;
      fz=fm=0.0;
      for(i=1;i<=m;i++)
      {
        fz+=a[d[i].pos];
        fm+=b[d[i].pos];
      }
      l=fz/fm;
      if(fabs(ans-l)<eps)break;
    }
    printf("%.0f\n",100.0*ans);
  }
    return 0;
}
```

3.3 Biggest Tuan

```cpp
//最大独立集即补图的最大团
#include<cstdio>
#include<cstring>
#define N 1010
bool flag[N], a[N][N];
int ans, cnt[N], group[N], n, vis[N];
// 最大团： V 中取 K 个顶点，两点间相互连接
// 最大独立集： V 中取 K 个顶点，两点间不连接
// 最大团数量 = 补图中最大独立集数

bool dfs( int u, int pos ){
    int i, j;
    for( i = u+1; i <= n; i++){
        if( cnt[i]+pos <= ans ) return 0;
        if( a[u][i] ){
             // 与目前团中元素比较，取 Non-N(i)
            for( j = 0; j < pos; j++ ) if( !a[i][ vis[j] ] )
            ↪  break;
            if( j == pos ){     // 若为空，则皆与 i 相邻，则此时将
            ↪  i 加入到 最大团中
                vis[pos] = i;
                if( dfs( i, pos+1 ) ) return 1;
            }
        }
    }
    if( pos > ans ){
            for( i = 0; i < pos; i++ )
                group[i] = vis[i]; // 最大团 元素
            ans = pos;
            return 1;
    }
    return 0;
}
void maxclique()
{
    ans=-1;
    for(int i=n;i>0;i--)
    {
        vis[0]=i;
        dfs(i,1);
        cnt[i]=ans;
    }
}
```

3.4 dijkstra(nlogn)

```cpp
using namespace std;
#define pii pair<int, int>
priority_queue<pii, vector<pii>, greater<pii> > heap;
struct edge
{
  int v, l, next;
}e[13007];
int n, m, S, T;
int tot=2, dist[2502], head[2502];
bool visited[2502];
void addedge(int x,int y,int z)
{
  e[tot].v=y, e[tot].l=z, e[tot].next=head[x], head[x]=tot++;
  e[tot].v=x, e[tot].l=z, e[tot].next=head[y], head[y]=tot++;
}
void dij(int x)
{
  if (x == T) return ;
  visited[x] = true;
  for (int p = head[x]; p; p = e[p].next)
    if (!visited[e[p].v] && dist[e[p].v] > (dist[x] + e[p].l))
      dist[e[p].v] = dist[x] + e[p].l,
      ↪ heap.push(make_pair(dist[e[p].v], e[p].v));
  while (!heap.empty() && visited[heap.top().second])
    heap.pop();
  dij(heap.top().second);
}
int main()
{
  scanf("%d%d%d%d",&n,&m,&S,&T);
  int x, y, z;
  for (int i=1;i<=m;i++)
    scanf("%d%d%d",&x,&y,&z), addedge(x, y, z);
  for (int i=1;i<=n;i++)
    dist[i]=0x7fffffff;
  dist[S]=0;
  dij(S);
  printf("%d\n",dist[T]);
  return 0;
}
```

3.5 Hamilton

```
/*
【题目来源】
http://poj.org/problem?id=2438
【题目分析】
有敌对关系的小朋友，不能坐在一起。最后围成一个圈，吃饭。。。
将小朋友看成点，有敌对关系的看成没有边，最后构成一个回路。
哈密顿回路。

【小小总结】
哈密顿回路
充分条件：
无向连通图中任意 2 点度数之和大于等于顶点数，则必定存在哈密顿回路。

思路分析：
1. 任意找两个相邻的节点 S 和 T，在它们基础上扩展出一条尽量长的没有重复节点的路径。
也就是说，如果 S 与节点 v 相邻，而且 v 不在路径 S->T 上，则可以把该路径变成 v->S->T，然后 v 成为新的 S。
从 S 和 T 分别向两头扩展，直到无法扩为止，即所有与 S 或 T 相邻的节点都在路径 S->T 上。
2. 若 S 与 T 相邻，则路径 S->T 形成了一个回路。
3. 若 S 与 T 不相邻，可以构造出一个回路。设路径 S->T 上有 k+2 个节点，依次为 S、v1、v2.....vk 和 T。
可以证明 v1 到 vk 中必定存在 vi，满足 vi 与 T 相邻，且 vi+1 与 S 相邻。（其实 vi,vi+1 与 s t 同时相邻）（怎么证明就不赘述了，反正刷题肯定不会叫你证）
找到了满足条件的节点 vi 以后，就可以把原路径变成 S->vi+1->T->vi->S，即形成了一个回路。（自己画图就知道了）
4. 现在我们有了一个没有重复节点的回路。如果它的长度为 N，则哈密顿回路就找到了。
如果回路的长度小于 N，由于整个图是连通的，所以在该回路上，一定存在一点与回路以外的点相邻。
那么从该点处把回路断开，就变回了一条路径。再按照步骤 1 的方法尽量扩展路径，则一定有新的节点被加进来。（画图就知道了）
接着回到步骤 2。

伪代码：
思路清楚后主要是理解好伪代码，伪代码一懂代码就写出来了。关于下面步骤中为什么要倒置，自己画画图就清楚了。
s 为哈密顿回路起点，t 为当前哈密顿回路的终点，ans[] 就是哈密顿回路啦，默认不包含 0 顶点
1. 初始化，令 s=1,t 为任意与 s 相邻的点。
2. 若 ans[] 中的元素个数小于 n，则从 t 开始扩展，若可扩展，则把新点 v 加入 ans[]，并令 t=v，继续扩展到无法扩展。
```

```
3. 将 ans[] 倒置，s,t 互换，从 t（原来的 s）开始扩展，若可扩展，则把新点 v 加入 ans[]，并令 t=v，继续扩展到无法扩展。
4. 此时 s,t 两头都无法扩展了，若 s,t 相连，则继续步骤 5。若 st 不相连，则遍历 ans[]，必定会有 2 点，ans[i] 与 t 相连，ans[i+1] 与 s 相连，
将 ans[i+1] 到 t 倒置，t=ans[i+1](未倒置前的)
5.st 相连，此时为一个环。若 ans[] 个数等于 n，算法结束，ans[] 为哈密顿回路，如需要再添加一个起点。
若 ans[] 个数小于 n，遍历 ans[]，寻找 ans[i]，使得 ans[i] 与 ans[] 外一点 j 相连，倒置 ans[] 中 s 到 ans[i-1] 部分，令 s = ans[i-1]，
再倒置 ans[] 中 ans[i] 到 t 的部分，j 加入 ans[],t = j. 继续步骤 2

下面去掉 main 函数，就是求解哈密顿回路的模版了。
*/
#include <iostream>
#include <cstring>
#include <algorithm>
using namespace std;

#define Max 500

int map[Max][Max];
int ans[Max];
bool vis[Max];

//ans 数组的 index
int index;
int n, m;
int s, t;

void init()
{
    for (int i = 0; i < Max; ++i)
        for (int j = 0; j < Max; ++j)
            if (i == j)
                map[i][j] = 0;
            else
                map[i][j] = 1;

    memset(ans, 0, sizeof(ans));
    memset(vis, 0 , sizeof(vis));
    index = 0;
}

```

```
71  void reverse(int a, int b)
72  {
73      while (a < b)
74      {
75          swap(ans[a], ans[b]);
76          a++;
77          b--;
78      }
79  }
80
81  void expand()
82  {
83      while (true)
84      {
85          int i;
86          for (i = 1; i <= n; ++i)
87          {
88              if (!vis[i] && map[i][t])//未被访问且与 t 相连
89              {
90                  ans[index++] = i;
91                  vis[i] = true;
92                  t = i;
93                  break;
94              }
95          }
96          if (i > n) break;//无法扩展
97      }
98  }
99
100 void Hamilton()
101 {
102     //初始化 s = 1
103     s = 1;
104
105     //取任意连接 s 的点
106     for (int i = 1; i <= n; ++i)
107     {
108         if (map[i][s])
109         {
110             t = i;
111             break;
112         }
113     }
114     vis[s] = true;
115     vis[t] = true;
```

```
116         ans[index++] = s;
117         ans[index++] = t;
118 
119         while (true)
120         {
121             //从 t 向外扩展
122             expand();
123 
124             //t 扩展完毕，倒置 ans 并交换 s,t
125             reverse(0, index-1);
126 
127             swap(s, t);
128 
129             //从另一头，t(原来的 s) 继续扩展
130             expand();
131 
132             //若 s,t 不相连，处理成相连
133             if (!map[s][t])
134             {
135                 //在 ans[1] 到 ans[index-2] 中寻找两个相邻的且与 st 同
                    ↪ 时相连的点（必存在） 因为涉及 i+1 所以 i < index-2
136                 for (int i = 1; i < index-2; ++i)
137                 {
138                     if (map[ans[i+1]][s] && map[ans[i]][t])
139                     {
140                         reverse(i+1, index-1);//倒置 ans[i+1] 到
                            ↪ ans[index-1]
141                         t = ans[index-1];//更新 t
142                         break;
143                     }
144                 }
145             }
146 
147             //若 ans 元素有 n 个，说明算法完成
148             if (index == n) return;
149 
150             //若 ans 元素不满 n 个，ans[] 中寻找与未被遍历过的点相连的
                ↪ 点，但这一点必定不是 s,t. 因为 s,t 已经遍历到无法遍历才
                ↪ 能走到这一步
151             for (int j = 1; j <= n; ++j)
152             {
153                 if (!vis[j])
154                 {
155                     int i;
156                     for (i = 1; i < index-1; ++i)//排除 st
```

```
157                 {
158                     if (map[ans[i]][j])
159                     {
160                         s = ans[i-1];
161                         t = j;
162                         reverse(0, i-1);
163                         reverse(i,index-1);
164                         ans[index++] = j;
165                         vis[j] = true;
166                         break;
167                     }
168                 }
169                 if (map[ans[i]][j])break;//记得有 2 个循环，要
                    ↪ break 两次
170             }
171         }
172         //继续返回，从 t 扩展。。
173     }
174 }
175
176 int main()
177 {
178     while (cin >> n >> m, n||m)
179     {
180         n *= 2;
181         init();
182         int temp1, temp2;
183         for (int i = 0; i < m; ++i)
184         {
185             cin >> temp1 >> temp2;
186             map[temp1][temp2] = 0;
187             map[temp2][temp1] = 0;
188         }
189         Hamilton();
190         cout << ans[0];
191         for (int i = 1; i < index; ++i)
192             cout << ' ' << ans[i];
193         cout << endl;
194     }
195 }
```

3.6 K-shortest path

```
int n,m,s,t,k,dis[MAXN];
struct node
{
        int v,c;
        node(int v,int c):v(v),c(c){}
        inline bool operator<(const node &b) const//用于优先队列先出的条件
        {
                return c+dis[v]>b.c+dis[b.v];
        }
};
vector<node> map1[MAXN];//用于 dijkstra 算法
vector<node> map2[MAXN];//用于 A_star 算法
void dijkstra()
{
        int i,find[MAXN],v;
        for(i=1;i<=n;i++)dis[i]=INF;
        memset(find,0,sizeof(find));
        priority_queue<node> heap;
        dis[t]=0;
        heap.push(node(t,0));
        while(!heap.empty())
        {
                v=heap.top().v;
                heap.pop();
                if(find[v])continue;
                find[v]=1;
                for(i=0;i<map1[v].size();i++)
                        if(!find[map1[v][i].v] && dis[v]+map1[v][i].c<dis[map1[v][i].v])
                        {
                                dis[map1[v][i].v]=dis[v]+map1[v][i].c;
                                heap.push(node(map1[v][i].v,dis[map1[v][i].v]));
                        }
        }
}
int A_star()
{
        int i,cnt[MAXN],v,g;
        if(dis[s]==INF)return -1;
        priority_queue<node> heap;
        memset(cnt,0,sizeof(cnt));
```

```
41          heap.push(node(s,0));//0 是 g(x)
42          while(!heap.empty())
43          {
44                  v=heap.top().v;
45                  g=heap.top().c;
46                  heap.pop();
47                  cnt[v]++;
48                  if(cnt[t]==k)return g;
49                  if(cnt[v]>k)continue;
50                  for(i=0;i<map2[v].size();i++)
51
                            ↪ heap.push(node(map2[v][i].v,g+map2[v][i].c));
52          }
53          return -1;
54  }
55  int main()
56  {
57          int i,u,v,c;
58          cin>>n>>m;
59          for(i=0;i<m;i++)
60          {
61                  cin>>u>>v>>c;
62                  map2[u].push_back(node(v,c));
63                  map1[v].push_back(node(u,c));//反向储存求各节点
                    ↪ 到目标节点的最短距离
64          }
65          cin>>s>>t>>k;
66          if(s==t)k++;
67          dijkstra();
68          int ans=A_star();
69          cout<<ans<<endl;
70          return 0;
71  }
```

3.7 Stable Wedding

```cpp
//对男性（na）最优
const int maxn=2005;

int na[maxn][maxn],nv[maxn][maxn];
queue<int> q;
int i,j,k,m,n,T;

int main()
{
    scanf("%d",&T);
    while(T--)
    {
        scanf("%d",&n);
        memset(na,0,sizeof(na));
        memset(nv,0,sizeof(nv));
        for(i=1;i<=n;i++)
            for(j=1;j<=n;j++)scanf("%d",&na[i][j]);
        for(i=1;i<=n;i++)
            for(j=1;j<=n;j++)
            {
                scanf("%d",&m);
                nv[i][m]=j;
            }
        while(!q.empty())q.pop();
        for(i=1;i<=n;i++)q.push(i);
        while(!q.empty())
        {
            m=q.front();
            q.pop();
            k=na[m][++na[m][0]];
            if(nv[k][0]==0)
            {
                nv[k][0]=m;
                continue;
            }else
            {
                j=nv[k][0];
                if(nv[k][m]<nv[k][j])
                {
                    q.push(j);
                    nv[k][0]=m;
                    continue;
                }else q.push(m);
            }
```

```
45          }
46          for(i=1;i<=n;i++)na[nv[i][0]][0]=i;
47          for(i=1;i<=n;i++)printf("%d\n",na[i][0]);
48          if(T>0)printf("\n");
49      }
50      return 0;
51  }
```

4 Bipartite Graph Match

4.1 Bipartite Graph Match–BFS

```
struct Edge
{
    int from;
    int to;
    int weight;

    Edge(int f, int t, int w):from(f), to(t), weight(w) {}
};

vector<int> G[__maxNodes]; /* G[i] 存储顶点 i 出发的边的编号 */
vector<Edge> edges;
typedef vector<int>::iterator iterator_t;
int num_nodes;
int num_left;
int num_right;
int num_edges;

queue<int> Q;
int prev[__maxNodes];
int Hungarian()
{
    int ans = 0;
    memset(matching, -1, sizeof(matching));
    memset(check, -1, sizeof(check));
    for (int i=0; i<num_left; ++i) {
        if (matching[i] == -1) {
            while (!Q.empty()) Q.pop();
            Q.push(i);
            prev[i] = -1; // 设 i 为路径起点
            bool flag = false; // 尚未找到增广路
            while (!Q.empty() && !flag) {
                int u = Q.front();
                for (iterator_t ix = G[u].begin(); ix !=
                ↪ G[u].end() && !flag; ++ix) {
                    int v = edges[*ix].to;
                    if (check[v] != i) {
                        check[v] = i;
                        Q.push(matching[v]);
                        if (matching[v] >= 0) { // 此点为匹配点
                            prev[matching[v]] = u;
                        } else { // 找到未匹配点，交替路变为增广路
```

```
                                flag = true;
                                int d=u, e=v;
                                while (d != -1) {
                                    int t = matching[d];
                                    matching[d] = e;
                                    matching[e] = d;
                                    d = prev[d];
                                    e = t;
                                }
                            }
                        }
                    }
                    Q.pop();
                }
                if (matching[i] != -1) ++ans;
            }
        }
        return ans;
}

int main()
{

    return 0;
}
```

4.2 Bipartite Graph Match–DFS

```cpp
#include<bits/stdc++.h>

using namespace std;

vector<int> g[1005];
int vis[1005];
int vv[1005],res[1005];
int i,j,k,l,m,n,T;

void dfs(int x)//染色区别二分图
{
  for(unsigned int i=0;i<g[x].size();i++)
  {
    int v=g[x][i];
    if(!vis[v])
    {
      vis[v]=-vis[x];dfs(v);
      continue;
    }else if(vis[v]==vis[x])return;
  }
}

bool dfss(int x)
{
  for(unsigned int i=0;i<g[x].size();i++)
  {
    int v=g[x][i];
    if(vv[v])continue;
    vv[v]=1;//如果当前结点已被搜索过（剪枝）
    if(res[v]==0||dfss(res[v]))//寻找增广路
    {
      res[v]=x;//res 表示与该点匹配的点编号
      return true;
    }
  }
  return false;
}

int main()//本程序默认二分图，非二分图会出错
{
  memset(vis,0,sizeof(vis));
  scanf("%d%d",&n,&m);
  for(i=1;i<=n;i++)g[i].clear();
  for(i=1;i<=m;i++)
```

```
45      {
46        scanf("%d%d",&j,&k);
47        g[j].push_back(k);
48        g[k].push_back(j);
49      }
50      for(i=1;i<=n;i++)
51        if(!vis[i]){vis[i]=1;dfs(i);}//先进行黑白染色，区分开二分图
52      memset(res,0,sizeof(res));k=0;
53      for(i=1;i<=n;i++)//进行增广
54        if(vis[i]==1)
55        {
56          memset(vv,0,sizeof(vv));
57          if(dfss(i))k++;
58        }
59      printf("%d",k);//最大匹配数
60      return 0;
61  }
```

5 Flow

5.1 dinic

```cpp
#include<iostream>
#include<cstdio>
#include<queue>
#define maxn 1005          //点数
#define maxm 80005          //边数
#define INF 0x3f3f3f3f
#define rever(x) (mem+((x-mem)^1))
using namespace std;
struct edge
{
  int s,t,v,c;
  edge* next;
}mem[maxm],*head[maxn],*prev[maxn];
queue<int> q;
int cnt=-1,n;
int dis[maxn];
int S,T;
void add_edge(int s,int t,int v,int c)
{
  mem[++cnt].s=s;mem[cnt].t=t;mem[cnt].v=v;mem[cnt].c=c;mem[cnt].next=head[s];head[s]=mem+cnt
  mem[++cnt].s=t;mem[cnt].t=s;mem[cnt].v=0;mem[cnt].c=-
   ↪  c;mem[cnt].next=head[t];head[t]=mem+cnt;
}
bool bfs()
{
  for (int i=0;i<=n;i++) dis[i]=INF;
  q.push(S);dis[S]=0;
  while(!q.empty())
  {
    for (edge *it=head[q.front()];it;it=it->next)
    if (it->v&&dis[q.front()]+it->c<dis[it->t])
    {
      dis[it->t]=dis[q.front()]+it->c;
      prev[it->t]=it;
      q.push(it->t);
    }
    q.pop();
  }
  return (dis[T]!=INF);
}
int cost=0;
```

```
int dinic()
{
  int flow=0;
  while(bfs())
  {
    int augflow=INF,tmpcost=0;
    for (edge* it=prev[T];it;it=prev[it->s])
    {
      augflow=min(augflow,it->v);
      tmpcost+=it->c;
    }
    for (edge* it=prev[T];it;it=prev[it->s])
    {
      it->v-=augflow;
      rever(it)->v+=augflow;
    }
    flow+=augflow;cost+=augflow*tmpcost;
  }
  return flow;
}
int N,M,A,B,C;
int main()
{
  scanf("%d%d",&N,&M);
  S=0;T=N+1;n=T;
  add_edge(S,1,2,0);add_edge(N,T,2,0);
  for (int i=1;i<=M;i++)
  {
    scanf("%d%d%d",&A,&B,&C);
    add_edge(A,B,1,C);
    add_edge(B,A,1,C);
  }
  dinic();
  printf("%d\n",cost);
  return 0;
}
```

5.2 isap

```cpp
#include<iostream>
#include<cstdio>
#include<queue>
#include<algorithm>
#include<cstring>
#include<cmath>
using namespace std;
#define maxn 205            //最大点数
#define maxm 205            //最大边数
#define rever(x) (mem+((x-mem)^1))
struct edge
{
  int s,t,v;
  edge* next;
}mem[maxm*2],*head[maxn];
int cnt=-1;
void add_edge(int s,int t,int v)
{
  mem[++cnt].s=s;mem[cnt].t=t;mem[cnt].v=v;mem[cnt].next=head[s];head[s]=mem+cnt;
  mem[++cnt].s=t;mem[cnt].t=s;mem[cnt].v=0;mem[cnt].next=head[t];head[t]=mem+cnt;
}
int n,m;

int S,T;
int numbs[maxn];
int d[maxn];
edge* cur[maxn],*revpath[maxn];

void bfs()
{
  queue<int> q;
  while(!q.empty()) q.pop();
  for (int i=1;i<=n;i++) d[i]=maxn-1;        //由初始下标决定 01
  d[T]=0;q.push(T);
  while(!q.empty())
  {
    int u=q.front();
    q.pop();
    for (edge* it=head[u];it;it=it->next)
    {
      edge *now=rever(it);
      if (now->v==0||d[now->s]<n) continue;
      d[now->s]=d[u]+1;
      q.push(now->s);
```

```
45      }
46    }
47    memset(numbs,0,sizeof(numbs));
48    for (int i=1;i<=n;i++) numbs[d[i]]++;       //由初始下标决定
      ↪ 01
49  }
50
51  int isap()
52  {
53    int flow=0;
54    for (int i=1;i<=n;i++) cur[i]=head[i];       //由初始下标决定 01
55    int u=S;
56    while(d[S]<n)
57    {
58      if (u==T)
59      {
60        int augflow=2147483647;
61        for (int i=S;i!=T;i=cur[i]->t)
62          augflow=min(augflow,cur[i]->v);
63        for (int i=S;i!=T;i=cur[i]->t)
64        {
65          cur[i]->v-=augflow;
66          rever(cur[i])->v+=augflow;
67        }
68        flow+=augflow;u=S;
69      }
70      edge *e;
71      for (e=cur[u];e;e=e->next)
72        if (e->v&&d[u]==(d[e->t]+1)) break;
73      if (e)
74      {
75        cur[u]=e;
76        revpath[e->t]=rever(e);
77        u=e->t;
78      }
79      else
80      {
81        numbs[d[u]]--;
82        if (numbs[d[u]]==0) break;
83        cur[u]=head[u];
84        int mindist=n;
85        for (edge* it=head[u];it;it=it->next)
86          if (it->v) mindist=min(mindist,d[it->t]);
87        d[u]=mindist+1;
88        numbs[d[u]]++;
89        if (u!=S) u=revpath[u]->t;
```

```
90        }
91      }
92      return flow;
93    }
94    int main()
95    {
96      while(~scanf("%d%d",&m,&n))
97      {
98        cnt=-1;
99        memset(head,0,sizeof(head));
100       for (int i=1;i<=m;i++)
101       {
102         int s,e,c;
103         scanf("%d%d%d",&s,&e,&c);
104         add_edge(s,e,c);
105       }
106       S=1;T=n;n=n;
107       bfs();
108       printf("%d\n",isap());
109     }
110     return 0;
111   }
```

6 LCA

6.1 Double

```cpp
#include<bits/stdc++.h>

using namespace std;

struct tree
{
  vector<int> son;
  int fat,dep;
}a[100005];

map<string,int> my;
map<int,string> ym;

int i,j,k,l,m,n,T,o,md,sta;
queue<int> q;
bool vis[100005];
int ans[100005];
int lca[100005][20];

void bfs(int x)
{
  while(!q.empty())q.pop();
  q.push(x);
  while(!q.empty())
  {
    int t=q.front();q.pop();
    a[t].dep=a[a[t].fat].dep+1;
    if(a[t].dep>md)md=a[t].dep;
    for(unsigned int i=0;i<a[t].son.size();i++)
      q.push(a[t].son[i]);
  }
}

int find(int x,int y)
{
  if(a[x].dep<a[y].dep)
  {int temp=x;x=y;y=temp;}
  while(a[x].dep>a[y].dep)
  {
    int j=0;
    while(a[lca[x][j]].dep>a[y].dep)j++;
```

```
42      if(a[lca[x][j]].dep==a[y].dep){x=lca[x][j];break;}
43      x=lca[x][--j];
44    }
45    if(x==y)return y;
46    while(x!=y)
47    {
48      j=0;
49      while(lca[x][j]!=lca[y][j])j++;
50      if(j==0)break;j--;
51      x=lca[x][j];y=lca[y][j];
52    }
53    return a[x].fat;
54  }
55
56  int main()
57  {
58    memset(a,sizeof(a),0);
59    memset(vis,0,sizeof(vis));
60    cin>>n;
61    for(i=1;i<=100005;i++)a[i].son.clear();
62    my.clear();ym.clear();l=0;
63    for(i=1;i<=n;i++)
64    {
65      string s1,s2;
66      cin>>s1>>s2;
67      j=cl(s1);k=cl(s2);
68      a[k].fat=j;
69      a[j].son.push_back(k);
70      vis[k]=true;
71    }
72    for(i=1;i<=l;i++)if(!vis[i]){sta=i;break;}
73    memset(vis,0,sizeof(vis));
74    md=a[0].dep=a[0].fat=a[sta].fat=0;bfs(sta);
75    for(i=1;i<=l;i++)lca[i][0]=a[i].fat;
76    for(j=1;(1<<j)<=md;j++)
77      for(i=1;i<=l;i++)
78        lca[i][j]=lca[lca[i][j-1]][j-1];
79    T=o=0;
80    cin>>m;
81    for(i=1;i<=m;i++)
82    {
83      string s1,s2;
84      cin>>s1>>s2;
85      j=cl(s1);k=cl(s2);
86      cout<<ym[find(j,k)]<<endl;
87    }
```

```
88      return 0;
89  }
```

6.2 ST

```cpp
#include<bits/stdc++.h>

using namespace std;

struct tree
{
  vector<int> son;
  int fat,dep;
}a[100005];

int i,j,k,l,m,n,T,o,md,sta;
bool vis[100005];
int ans[100005];
int st[100005][20];
int se[2100000],no[100005],fir[1000005];

void dfs(int x)
{
  no[++md]=x;
  int t=md;
  se[++T]=t;
  fir[x]=T;
  for(unsigned int i=0;i<a[x].son.size();i++)
  {
    dfs(a[x].son[i]);
    se[++T]=t;
  }
}

void getST()
{
  for(int i=1;i<=T;i++)st[i][0]=se[i];
  for(int j=1;(1<<j)<=T;j++)
    for(int i=1;i<=T-(1<<j)+1;i++)
      st[i][j]=min(st[i][j-1],st[i+(1<<(j-1))][j-1]);
}

int find(int x,int y)
{
  if(x>y){int temp=x;x=y;y=temp;}
  int j=0;
  while((1<<j)<=(y-x+1))j++;
  j--;
  return min(st[x][j],st[y-(1<<j)+1][j]);
```

```
}

int main()
{
  memset(a,sizeof(a),0);
  memset(vis,0,sizeof(vis));
  cin>>n;md=T=0;
  for(i=1;i<=100005;i++)a[i].son.clear();
  my.clear();ym.clear();l=0;
  for(i=1;i<=n;i++)
  {
    string s1,s2;
    cin>>s1>>s2;
    j=cl(s1);k=cl(s2);
    a[k].fat=j;
    a[j].son.push_back(k);
    vis[k]=true;
  }
  for(i=1;i<=l;i++)if(!vis[i]){sta=i;break;}
  memset(st,0,sizeof(st));
  dfs(sta);cin>>m;
  getST();
  for(i=1;i<=m;i++)
  {
    string s1,s2;
    cin>>s1>>s2;
    j=cl(s1);k=cl(s2);
    cout<<ym[no[find(fir[j],fir[k])]]<<endl;
  }
  return 0;
}
```

6.3 Tarjan

```cpp
#include<bits/stdc++.h>

using namespace std;

struct edge
{
  int to,dist;
};

vector<edge> g[40005];
vector<edge> q[40005];

int i,j,k,l,m,n,T,o;
bool vis[40005];
int dis[40005];
int ans[205];
int bcj[40005];

int find(int x)
{
  if(bcj[x]==x)return x;
  else return bcj[x]=find(bcj[x]);
}

void dfs(int x)
{
  vis[x]=true;
  for(unsigned int i=0;i<q[x].size();i++)
  {
    edge r=q[x][i];
    if(vis[r.to])
    {
      int zx=find(r.to);
      ans[r.dist]=dis[x]+dis[r.to]-2*dis[zx];
    }
  }
  for(unsigned int i=0;i<g[x].size();i++)
  {
    edge v=g[x][i];
    if(!vis[v.to])
    {
      dis[v.to]=dis[x]+v.dist;
      dfs(v.to);
      bcj[v.to]=x;
```

```
    }
  }
}

int main()
{
  scanf("%d",&T);
  while(T--)
  {
    memset(ans,0,sizeof(ans));
    memset(vis,0,sizeof(vis));
    memset(dis,0,sizeof(dis));

    scanf("%d%d",&n,&m);
    for(i=1;i<=n;i++)q[i].clear();
    for(i=1;i<=n;i++)bcj[i]=i;
    for(i=1;i<=n;i++)g[i].clear();

    for(i=1;i<=n-1;i++)
    {
      scanf("%d%d%d",&j,&k,&l);
      edge r;
      r.to=k;r.dist=l;
      g[j].push_back(r);
      r.to=j;r.dist=l;
      g[k].push_back(r);
    }

    for(i=1;i<=m;i++)
    {
      scanf("%d%d",&j,&k);
      edge r;
      r.to=k;r.dist=i;
      q[j].push_back(r);
      r.to=j;r.dist=i;
      q[k].push_back(r);
    }
    memset(vis,0,sizeof(vis));
    dfs(1);
    for(i=1;i<=m;i++)printf("%d\n",ans[i]);
  }
  return 0;
}
```

7 MST

7.1 Directed-MST

```
#define type int//type 可选择 int 或者 double

const type inf=2147483640;
const int maxn=1005;

int pre[maxn],id[maxn],vis[maxn];
type in[maxn];

struct edge
{
  int from,to;
  type cost;
  edge(int from=0,int to=0,type
   ↪  cost=0):from(from),to(to),cost(cost){}
}edg[10005];

type ZLEdmonds(int n,int m,int root)//自环在输入建图时直接忽略，如
 ↪  需加入，可另存
{
  type tot=0.0;
  //判断是否有树
  while(true)
  {
    for(int i=1;i<=n;i++)in[i]=inf;
    for(int i=1;i<=m;i++)
    {
      int u=edg[i].from;
      int v=edg[i].to;
      if(edg[i].cost<in[v] && u!=v){pre[v]=u;in[v]=edg[i].cost;}
    }
    for(int i=1;i<=n;i++)if(i!=root && in[i]==inf)return -1;
    //找环
    int cnt=1;
    memset(id,0,sizeof(id));
    memset(vis,0,sizeof(vis));
    in[root]=0;
    for(int i=1;i<=n;i++)//标记每个环
    {
      tot+=in[i];
      int v=i;
      while(vis[v]!=i && id[v]==0 && v!=root)
```

```
40          {vis[v]=i;v=pre[v];}
41          if(v!=root && id[v]==0)//缩点
42          {
43            for(int u=pre[v];u!=v;u=pre[u])id[u]=cnt;
44            id[v]=cnt++;
45          }
46        }
47        if(cnt==1)break;
48        for(int i=1;i<=n;i++)if(id[i]==0)id[i]=cnt++;
49        //建立新图
50        for(int i=1;i<=m;i++)
51        {
52          int u=edg[i].from;
53          int v=edg[i].to;
54          edg[i].from=id[u];
55          edg[i].to=id[v];
56          if(id[u]!=id[v])edg[i].cost-=in[v];
57        }
58        n=cnt-1;
59        root=id[root];
60      }
61      return tot;
62  }
63
64  int main()
65  {
66
67      return 0;
68  }
```

7.2 K-Degree MST

```
/************************************************
算法引入：
最小 k 度限制生成树，就是指有特殊的某一点的度不能超过 k 时的最小生成树；
如果 T 是 G 的一个生成树且 dT(v0)=k，则称 T 为 G 的 k 度限制生成树；
G 中权值和最小的 k 度限制生成树称为 G 的最小 k 度生成树；

算法思想：
设特殊的那点为 v0，先把 v0 删除，求出剩下连通图的所有最小生成树；
假如有 m 棵最小生成树，那么这些生成树必定要跟 v0 点相连；
也就是说这棵生成树的 v0 点至少是 m 度的；
若 m>k，条件不成立，无法找到最小 k 度限制生成树；
若 m<=k，则枚举 m 到 k 的所有最小生成树，即一步步将 v0 点的度加 1，直到 v0 点的度为 k 为止；
则 v0 点度从 m 到 k 的 (k-m+1) 棵最小生成树中最小的那棵即为答案；

算法步骤：
(1) 先求出最小 m 度限制生成树：
原图中去掉和 V0 相连的所有边（可以先存两个图，建议一个邻接矩阵，一个邻接表，用方便枚举边的邻接表来构造新图）；
得到 m 个连通分量，则这 m 个连通分量必须通过 v0 来连接；
则在图 G 的所有生成树中 dT(v0)>=m；
则当 k<m 时，问题无解；
对每个连通分量求一次最小生成树；
对于每个连通分量 V'，用一条与 V0 直接连接的最小的边把它与 V0 点连接起来，使其整体成为一个生成树；
就得到了一个 m 度限制生成树，即为最小 m 度限制生成树；

(2) 由最小 m 度限制生成树得到最小 m+1 度限制生成树；
连接和 V0 相邻的点 v，则可以知道一定会有一个环出现（因为原来是一个生成树）；
只要找到这个环上的最大权边（不能与 v0 点直接相连）并删除，就可以得到一个 m+1 度限制生成树；
枚举所有和 V0 相邻点 v，找到替换后，增加权值最小的一次替换（如果找不到这样的边，就说明已经求出）；
就可以求得 m+1 度限制生成树；
如果每添加一条边，都需要对环上的边一一枚举，时间复杂度将比较高；
用动态规划解决；
设 dp(v) 为路径 v0—v 上与 v0 无关联且权值最大的边；
定义 father(v) 为 v 的父结点，由此可以得到状态转移方程：
dp(v)=max(dp(father(v)),ω(father(v),v));
边界条件为 dp[v0]=-∞（因为每次寻找的是最大边，所以-∞ 不会被考虑),dp[v']=-∞|(v0,v')∉E(T);

```

```
(3) 当 dT(v0)=k 时停止 (即当 V0 的度为 k 的时候停止), 但不一定 k 的时候最优;

算法实现：
并查集 +kruskal;
首先，每个连通分量的的最小生成树可以直接用一个循环，循环着 Kruskal 求出;
这里利用了联通分量间的独立性，对每个连通分量分别求最小生成树，和放在一起求，毫不影响;
而且 kruskral 算法保证了各连通分量边的有序性;
找最小边的时候，可以用动态规划，也可以这么做：
先走一个循环，但我们需要逆过来加边，将与 v0 关联的所有边从小到达排序;
然后将各连通分量连接起来，利用并查集可以保证每个连通分量只有一条边与 v0 相连;
由于边已经从小到达排序，故与每个连通分量相连的边就是每个连通分量与 v0 相连中的最小边;
然后求 m+1 度的最小生成树时，可以直接用 DFS，最小生成树要一直求到 k 度，然后从中找出一个最优值;

算法测试：
PKU1639(Picnic Planning);

题目大意:
给出 m 条边，每条边有两个端点和一个权值;
求这个图在满足以下条件的情况下的最小生成树;
在所有点中，有一个特殊点 Park，它在求得的最小生成树中的度必须小于等于某个值;
**************************************************/
#include<iostream>
#include<string>
#include<cstdio>
#include<map>
#include<cstring>
#include<algorithm>
using namespace std;

const int INF=99999999;
const int N=100;

int n,m;//n 为边的数量,m 表示限度值
int cnt;//计算出来的结点数
int set[N];
bool flag[N][N];
int G[N][N];
```

```cpp
int ans;

map<string,int> Map;

struct node
{
    int x,y,v;
} a[N*N];

struct edge
{
    int x,y,v;
} dp[N];

int get_num(string s)//返回每个人对应结点
{
    if(Map.find(s)==Map.end())//没有搜索到该键值
    {
        Map[s]=++cnt;//对应建图
    }
    // cout<<"  Map["<<s<<"]=="<<Map[s]<<endl;
    return Map[s];
}

bool cmp(node a,node b)
{
    return a.v<b.v;
}

int find_set(int x)
{
    if(x!=set[x])
        set[x]=find_set(set[x]);
    return set[x];
}

inline void union_set(int x,int y)
{
    set[y]=x;
}

void kruskal()//求 m 个连通分量的最小生成树
{
    for(int i=1; i<=n; i++)
    {
```

```
119         if(a[i].x==1||a[i].y==1)
120             continue;
121         int x=find_set(a[i].x);
122         int y=find_set(a[i].y);
123         if(x==y)
124             continue;
125         flag[a[i].x][a[i].y]=flag[a[i].y][a[i].x]=true;
126         set[y]=x;
127         ans+=a[i].v;
128     }
129 }
130
131 void dfs(int x,int fa)
132 {
133     for(int i=2; i<=cnt; i++)
134         if(i!=fa&&flag[x][i])
135         {
136             if(dp[i].v==-1)
137             {
138                 if(dp[x].v>G[x][i])//dp(v)=max(dp(father(v)),w(father(v),v));
139                 {
140                     dp[i]=dp[x];
141                 }
142                 else
143                 {
144                     dp[i].v=G[x][i];
145                     dp[i].x=x;
146                     dp[i].y=i;
147                 }
148             }
149             dfs(i,x);
150         }
151 }
152
153 void init()
154 {
155     ans=0;
156     cnt=1;
157     Map["Park"]=1;
158     memset(flag,0,sizeof(flag));
159     memset(G,-1,sizeof(G));
160     scanf("%d",&n);
161     for(int i=1; i<N; i++)//并查集初始化
162         set[i]=i;
```

```
163     string s;
164     for(int i=1; i<=n; i++)
165     {
166         cin>>s;
167         a[i].x=get_num(s);
168         cin>>s;
169         a[i].y=get_num(s);
170         cin>>a[i].v;
171         if(G[a[i].x][a[i].y]==-1)
172             G[a[i].x][a[i].y]=G[a[i].y][a[i].x]=a[i].v;
173         else//有重边
174 
                ↪  G[a[i].x][a[i].y]=G[a[i].y][a[i].x]=min(G[a[i].y][a[i].x],a[i].v);
175     }
176     scanf("%d",&m);//m 表示限度值
177 }
178 
179 void solve()
180 {
181     int tmp[N],Min[N];
182     for(int i=1; i<=cnt; i++)
183         Min[i]=INF;
184     sort(a+1,a+1+n,cmp);
185     kruskal();
186     for(int i=2; i<=cnt; i++)
187     {
188         if(G[1][i]!=-1)
189         {
190             int t=find_set(i);
191             if(Min[t]>G[1][i])//求每个连通分量中和顶点 1 连接的最小
                ↪  权边
192             {
193                 tmp[t]=i;
194                 Min[t]=G[1][i];
195             }
196         }
197     }
198 
199     int t=0;//t 表示最小限度
200     for(int i=1; i<=cnt; i++)
201         if(Min[i]!=INF)
202         {
203             t++;
204             flag[1][tmp[i]]=flag[tmp[i]][1]=true;
205             ans+=G[1][tmp[i]];
```

```
206             }
207
208         for(int i=t+1; i<=m; i++)//枚举 t 到 m 的所有最小生成树，即一步步将 v1 点的度加 1，直到 v1 点的度为 m 为止;
209         {
210             memset(dp,-1,sizeof(dp));//dp[v] 为路径 v0—v 上与 v0 无关联且权值最大的边;
211             dp[1].v=-INF;
212             for(int j=2; j<=cnt; j++)
213                 if(flag[1][j])
214                     dp[j].v=-INF;
215             dfs(1,-1);
216             int tmp,Min=INF;
217             for(int j=2; j<=cnt; j++)
218                 if(G[1][j]!=-1)
219                 {
220                     if(Min>G[1][j]-dp[j].v)
221                     {
222                         Min=G[1][j]-dp[j].v;
223                         tmp=j;
224                     }
225                 }
226             if(Min>=0)//找不到这样的边，就说明已经求出
227                 break;
228             flag[1][tmp]=flag[tmp][1]=true;
229             int x=dp[tmp].x;
230             int y=dp[tmp].y;
231             flag[x][y]=false;
232             flag[y][x]=false;
233             ans+=Min;
234         }
235
236         printf("Total miles driven: %d\n",ans);
237     }
238
239     int main()
240     {
241         fre-
            ↪ open("C:\\Users\\Administrator\\Desktop\\kd.txt","r",stdin);
242         init();
243         solve();
244         return 0;
245     }
```

7.3 Second-MST

```
int a[maxn];
int cost[maxn][maxn];
int lowcost[maxn],fat[maxn],maxd[maxn][maxn];
bool vis[maxn];
int i,j,k,l,m,n,T,u,v,ans,mini;

int main()
{
  scan(T);
  while(T--)
  {
    memset(lowcost,inf,sizeof(lowcost));
    memset(a,0,sizeof(a));
    memset(fat,0,sizeof(fat));
    memset(maxd,0,sizeof(maxd));
    memset(cost,inf,sizeof(cost));
    memset(vis,0,sizeof(vis));
    scan2(n,m);ans=0;
    for(i=1;i<=m;i++)
    {
      scan3(j,k,l);
      cost[j][k]=cost[k][j]=l;
    }
    vis[1]=true;a[k=1]=1;
    for(i=2;i<=n;i++){maxd[i][1]=maxd[1][i]=lowcost[i]=cost[1][i];fat[i]=1;}
    for(i=1;i<=n;i++)maxd[i][i]=cost[i][i]=0;
    for(u=1,i=1;i<=n-1;i++)
    {
      mini=inf,v=-1;
      for(j=1;j<=n;j++)
        if(!vis[j] && lowcost[j]<mini)
        {mini=lowcost[j];v=j;}
      vis[v]=true;
      ans+=mini;
      for(j=1;j<=k;j++)
        maxd[a[j]][v]=maxd[v][a[j]]=max(mini,maxd[fat[v]][a[j]]);
      a[++k]=v;
      for(j=1;j<=n;j++)
        if(!vis[j] && cost[v][j]<lowcost[j])
        {lowcost[j]=cost[v][j];fat[j]=v;}
    }
    mini=inf;
    for(i=1;i<=n-1;i++)
      for(j=i+1;j<=n;j++)
```

```
45          {
46            if(fat[i]==j || fat[j]==i || cost[i][j]==inf)continue;
47            mini=min(mini,cost[i][j]-maxd[i][j]);
48          }
49        if(mini==0)printf("Not Unique!\n");else printf("%d\n",ans);
50      }
51        return 0;
52  }
```

8 Tarjan

8.1 bridge cut-vertex

```
int dfs(int u,int fat)
{
  int lowu,lowv;
  lowu=pre[u]=++dfs_clock;
  int child=0;
  for(unsigned int i=0;i<g[u].size();i++)
  {
    int v=g[u][i];
    if(!pre[v])
    {
      child++;
      lowv=dfs(v,u);
      lowu=min(lowu,lowv);
      if(lowv>pre[u])p.push(edge(min(u,v),max(u,v)));
      if(lowv>=pre[u])iscut[u]=true;
    }else if(v!=fat) lowu=min(lowu,pre[v]);
  }
  if(fat==-1 && child<=1)iscut[u]=false;
  return lowu;
}

void tarjan(int n)
{
  dfs_clock=0;
  memset(pre,0,sizeof(pre));
  memset(iscut,0,sizeof(iscut));
  for(int i=1;i<=n;i++)if(!pre[i])dfs(i,-1);
}
```

8.2 edge-double connected

```
1 //DFS 遍历不走桥即可
```

8.3 SCC

```
//有向图

int pre[maxn],low[maxn],a[maxn],sccno[maxn],tot[maxn];
int edge[100005][2];
int dfs_clock,scc_cnt,maxx;
vector<int> g[maxn];
stack<int> s;

int n,m,p0,p1,i,j,k,l;

void dfs(int u)
{
  pre[u]=low[u]=++dfs_clock;
  s.push(u);
  for(unsigned int i=0;i<g[u].size();i++)
  {
    int v=g[u][i];
    if(!pre[v])
    {
      dfs(v);
      low[u]=min(low[u],low[v]);
    }else if(!sccno[v])low[u]=min(low[u],pre[v]);
  }
  if(low[u]==pre[u])
  {
    scc_cnt++;
    for(;;)
    {
      int x=s.top();s.pop();
      sccno[x]=scc_cnt;
      if(x==u)break;
    }
  }
}

void find(int n)
{
  dfs_clock=scc_cnt=0;
  memset(sccno,0,sizeof(sccno));
  memset(pre,0,sizeof(pre));
  memset(low,0,sizeof(low));
  while(!s.empty())s.pop();
  for(i=1;i<=n;i++)if(!pre[i])dfs(i);
}
```

8.4 vertex-double connected

```
struct edge
{
  int p,q;
  edge(int p=0,int q=0):p(p),q(q){}
}edg[maxm];

vector<int> g[maxn];
int bcc[maxm],pre[maxn];
int p[maxm],s[maxm];
int dfs_clock,bcc_cnt;

int dfs(int u,int fa)
{
  int lowu=pre[u]=++dfs_clock;
  for(unsigned int i=0;i<g[u].size();i++)
  {
    int side=g[u][i];
    int v=other(side,u);
    if(!pre[v])
    {
      s[++s[0]]=side;
      int lowv=dfs(v,u);
      lowu=min(lowu,lowv);
      if(lowv>=pre[u])
      {
        bcc_cnt++;
        for(;;)
        {
          int x=s[s[0]];s[0]--;
          bcc[x]=bcc_cnt;
          p[bcc_cnt]=min(p[bcc_cnt],x);
          if(x==side)break;
        }
      }
    }else if(pre[v]<pre[u] && v!=fa)
    {
      s[++s[0]]=side;lowu=min(lowu,pre[v]);
    }
  }
i  return lowu;
}

void tarjan(int n)
{
```

```
45    dfs_clock=bcc_cnt=0;
46    memset(pre,0,sizeof(pre));
47    memset(bcc,0,sizeof(bcc));
48    memset(p,0x3f3f3f3f,sizeof(p));
49    s[0]=0;
50    for(int i=1;i<=n;i++)if(!pre[i])dfs(i,-1);
51  }
```

9 Math

9.1 FFT

```cpp
#include<bits/stdc++.h>

using namespace std;

const double eps=1e-10;
const double pi=3.14159265358979323846264338327950;
const double eln=2.71828182845904523536028747135266;

const int maxn=105000;

complex<double> epsilon[maxn];
complex<double> arti_epsilon[maxn];
complex<double> a[maxn],b[maxn],c[maxn],temp[maxn];

int n1,n2,m;

void init_epsilon(int n)
{
  for(int i=0;i!=n;i++)
  {
    epsilon[i]=complex<double>(cos(2.0*pi*i/n),sin(2.0*pi*i/n));
    arti_epsilon[i]=conj(epsilon[i]);
  }
}

int calc(int t)
{
  int j=0;
  while((1<<j)<=t)j++;
  return 1<<j;
}


void DFT(int n,complex<double>*  buffer,int offset,int
↪  step,complex<double>* epsilon)
{
  if(n==1)return;
  int m=n>>1;
  DFT(m,buffer,offset,step<<1,epsilon);
  DFT(m,buffer,offset+step,step<<1,epsilon);
  for(int k=0;k!=m;k++)
```

```
41    {
42      int pos=2*step*k;
43      temp[k]=buffer[pos+offset]+epsilon[k*step]*buffer[pos+offset+step];
44      temp[k+m]=buffer[pos+offset]-
        ↪  epsilon[k*step]*buffer[pos+offset+step];
45    }
46    for(int i=0;i!=n;i++)buffer[i*step+offset]=temp[i];
47  }
48
49  //IDFT 将 DFT 的 epsilon 改为 arti_epsilon
50
51  void FFT(int m,complex<double>* a,complex<double>*
    ↪  b,complex<double>* c)
52  {
53    init_epsilon(m);
54    DFT(m,a,0,1,epsilon);
55    DFT(m,b,0,1,epsilon);
56    for(int i=0;i<=m;i++)c[i]=a[i]*b[i];
57    IDFT(m,c,0,1,epsilon);
58    double mm=m;
59    for(int i=0;i<=m;i++)c[i]/=mm;
60  }
61
62  int init()//n1,n2 表示多项式次数
63  {
64    double x,y;
65    scanf("%d%d",&n1,&n2);
66    memset(a,0,sizeof(a));
67    memset(b,0,sizeof(b));
68    for(int i=0;i<=n1;i++)
69    {
70      scanf("%lf %lf",&x,&y);
71      a[i].real(x);
72      a[i].imag(y);
73    }
74    for(int i=0;i<=n2;i++)
75    {
76      scanf("%lf %lf",&x,&y);
77      b[i].real(x);
78      b[i].imag(y);
79    }
80    m=calc(n1+n2);
81    return m;
82  }
83
84  void print()
```

```
85  {
86    for(int i=0;i<m;i++)printf("%lf %lf\n",real(c[i]),imag(c[i]));
87  }
88
89  int main()
90  {
91    m=init();
92    FFT(m,a,b,c);
93    print();
94  }
```

9.2 Integer High Accuracy

```cpp
#include<cstdio>
#include<vector>
#include<cstring>
#include<algorithm>
using namespace std;
typedef unsigned int UInt;
typedef unsigned long long ULL;
class Number :public vector<UInt>
{
  bool flag;
  Number(UInt value){ flag = 0; if (value)push_back(value); }
public:
  int cmp(const Number& num)const;
  void add(const Number& num);
  void sub(const Number& num);
  Number mul(const Number& num)const;
  Number div(const Number& num, Number& mod)const;
  Number divInt(UInt num, UInt& mod)const;
  void shr(UInt num);
  void shl(UInt num);
  void shr31(UInt num);
  void shl31(UInt num);
  void and(const Number& num);
  void or(const Number& num);
  Number not(UInt k)const;
  void xor(const Number& num);
  static ULL link(ULL x, UInt y){ return (x << 31) | y; }
public:
  Number(){ flag = 0; }
  Number(int value);
  Number(long long value);
  Number(const char *s);
  Number(const char *s, UInt k);
  void convert10(char *s);
  void convert2k(char*s, UInt k);
  bool operator < (const Number& num)const;
  bool operator <= (const Number& num)const;
  bool operator == (const Number& num)const;
  Number operator + (const Number& num)const;
  Number operator - (const Number& num)const;
  Number operator  * (const Number& num)const;
  Number operator  / (const Number& num)const;
  Number operator % (const Number& num)const;
  Number operator >> (UInt num)const;
```

```
45    Number operator << (UInt num)const;
46    Number operator & (const Number& num)const;
47    Number operator | (const Number& num)const;
48    Number operator ^ (const Number& num)const;
49  };
50  Number::Number(const char *s)
51  {
52    if (s[0] == '-'){ flag = 1; s++; }
53    else flag = 0;
54    vector<char> str(strlen(s));
55    for (int i = str.size() - 1; i >= 0; i--)
56      str[i] = s[str.size() - i - 1] - '0';
57    while (str.size()){
58      ULL sum = 0;
59      for (int i = str.size() - 1; i >= 0; i--){
60        sum = sum * 10 + str[i];
61        str[i] = (char)(sum >> 31);
62        sum &= ~(1 << 31);
63      }
64      push_back((UInt)sum);
65      while (str.size() && !str.back())str.pop_back();
66    }
67    if (!back())pop_back();
68  }
69  Number::Number(const char *s, UInt k)
70  {
71    if (s[0] == '-'){ flag = 1; s++; }
72    else flag = 0;
73    UInt cnt = 0;
74    ULL value = 0;
75    for (int i = strlen(s) - 1; i >= 0; i--){
76      value |= (ULL)(s[i] <= '9' ? s[i] - '0' : s[i] - 'A' + 10) <<
       ↪  cnt;
77      cnt += k;
78      if (cnt >= 31){
79        push_back((UInt)value & ~(1 << 31));
80        value >>= 31; cnt -= 31;
81      }
82    }
83    if (value)push_back((UInt)value);
84  }
85  Number::Number(int value)
86  {
87    if (value)push_back(value > 0 ? value : -value);
88    flag = value < 0;
89  }
```

```
Number::Number(long long value)
{
  flag = value < 0;
  if (flag)value = -value;
  while (value){
    push_back(value & ~(1 << 31));
    value >>= 31;
  }
}
void Number::convert10(char *s)
{
  if (flag)*s++ = '-';
  vector<UInt> copy = *this;
  UInt len = 0;
  while (copy.size()){
    ULL sum = 0;
    for (int i = copy.size() - 1; i >= 0; i--){
      sum = link(sum, copy[i]);
      copy[i] = (UInt)(sum / 10);
      sum %= 10;
    }
    s[len++] = (char)sum + '0';
    if (!copy.back())copy.pop_back();
  }
  if (len == 0)s[len++] = '0';
  reverse(s, s + len);
  s[len] = 0;
}
void Number::convert2k(char*s, UInt k)
{
  const char *table = "0123456789ABCDEF";
  UInt len = 0, cnt = 0, bound = (1 << k) - 1;
  ULL value = 0;
  if (flag)*s++ = '-';
  if (empty())s[len++] = '0';
  else{
    for (UInt i = 0;; cnt -= k){
      if (cnt < k){
        if (i == size())break;
        value |= (ULL)(*this)[i++] << cnt;
        cnt += 31;
      }
      s[len++] = table[value & bound];
      value >>= k;
    }
    s[len++] = table[value];
```

```
136       while (s[len - 1] == '0')len--;
137       reverse(s, s + len);
138     }
139     s[len] = 0;
140   }
141   int Number::cmp(const Number& num)const
142   {
143     if (size() != num.size())
144       return size() < num.size() ? -1 : 1;
145     for (int i = size() - 1; i >= 0; i--){
146       if ((*this)[i] != num[i])
147         return (*this)[i] < num[i] ? -1 : 1;
148     }
149     return 0;
150   }
151   bool Number::operator == (const Number& num)const{
152     return flag == num.flag && !cmp(num);
153   }
154   bool Number::operator < (const Number& num)const{
155     if (flag != num.flag)return flag;
156     return flag ? cmp(num) > 0 : cmp(num) < 0;
157   }
158   bool Number::operator <= (const Number& num)const{
159     if (flag != num.flag)return flag;
160     return flag ? cmp(num) >= 0 : cmp(num) <= 0;
161   }
162   //为提高效率确保 *this 位数 >=num 位数
163   void Number::add(const Number& num)
164   {
165     UInt f = 0, i = 0;
166     if (size() < num.size())resize(num.size());
167     for (; i < num.size(); i++){
168       (*this)[i] += num[i] + f;
169       f = (*this)[i] >> 31;
170       if (f)(*this)[i] ^= 1 << 31;
171     }
172     push_back(0);
173     for (; f; i++){
174       f = ++(*this)[i] >> 31;
175       if (f)(*this)[i] ^= 1 << 31;
176     }
177     if (!back())pop_back();
178   }
179   //确保 *this>=num
180   void Number::sub(const Number& num)
```

```
{
  UInt f = 0, i = 0;
  for (; i < num.size(); i++){
    (*this)[i] -= num[i] + f;
    f = (*this)[i] >> 31;
    if (f)(*this)[i] ^= 1 << 31;
  }
  for (; f; i++){
    f = --(*this)[i] >> 31;
    if (f)(*this)[i] ^= 1 << 31;
  }
  while (size() && !back())pop_back();
}
Number Number::operator + (const Number& num)const
{
  Number ret;
  if (flag == num.flag){
    if (size() < num.size()){ ret = num; ret.add(*this); }
    else{ ret = *this; ret.add(num); }
    ret.flag = flag;
  }
  else{
    int t = cmp(num);
    if (t < 0){
      ret = num; ret.sub(*this);
      ret.flag = num.flag;
    }
    else if (t > 0){
      ret = *this; ret.sub(num);
      ret.flag = flag;
    }
  }
  return ret;
}
Number Number::operator - (const Number& num)const
{
  Number ret;
  if (flag != num.flag){
    if (size() < num.size()){ ret = num; ret.add(*this); }
    else{ ret = *this; ret.add(num); }
    ret.flag = flag;
  }
  else{
    int t = cmp(num);
    if (t < 0){
      ret = num; ret.sub(*this);
```

```
227         ret.flag = !flag;
228       }
229       else if (t > 0){
230         ret = *this; ret.sub(num);
231         ret.flag = flag;
232       }
233     }
234     return ret;
235   }
236   //为提高效率确保 *this 位数 >=num 位数
237   Number Number::mul(const Number& num)const
238   {
239     if (num.empty() || empty())return 0;
240     Number ret;
241     ret.resize(size() + num.size(), 0);
242     for (int i = num.size() - 1; i >= 0; i--){
243       ULL sum = 0;
244       for (UInt j = 0; j < size(); j++){
245         sum += (ULL)num[i] * (*this)[j];
246         ret[i + j] += sum & ~(1 << 31);
247         sum >>= 31;
248         if (ret[i + j] & (1 << 31)){
249           sum++;
250           ret[i + j] ^= 1 << 31;
251         }
252       }
253       ret[i + size()] += (UInt)sum;
254     }
255     for (UInt i = size(); i < ret.size(); i++){
256       if (ret[i] & (1 << 31)){
257         ret[i] ^= 1 << 31;
258         ret[i + 1]++;
259       }
260     }
261     if (!ret.back())ret.pop_back();
262     return ret;
263   }
264   Number Number::operator * (const Number& num)const
265   {
266     Number ret = size() < num.size() ? num.mul(*this) : mul(num);
267     if (ret.size())ret.flag = flag ^ num.flag;
268     return ret;
269   }
270   Number Number::div(const Number& num, Number& mod)const
271   {
```

```
272   const UInt aSize = size(), bSize = num.size();
273   if (aSize < bSize){ mod = *this; return 0; }
274   Number ret;
275   ret.resize(aSize - bSize + 1);
276   mod.assign(begin() + aSize - bSize + 1, end());
277   ULL y = num.back();
278   int bit = 0;
279   for (int i = 16; i; i >>= 1){
280     if (y >> (bit + i))bit += i;
281   }
282   y = (y << (31 - bit)) + (num[bSize - 2] >> bit) + 1;
283   for (int i = ret.size() - 1; i >= 0; i--){
284     mod.shl31(1);
285     UInt oldSize = mod.size();
286     mod.resize(bSize + 1);
287     mod[0] = (*this)[i];
288     ULL x = link(mod[bSize], mod[bSize - 1]);
289     x = (x << (31 - bit)) | (mod[bSize - 2] >> bit);
290     if (!oldSize && mod[0])oldSize++;
291     mod.resize(oldSize);
292     if (ret[i] = (UInt)(x / y))mod.sub(num.mul(ret[i]));
293     if (mod.cmp(num) >= 0){
294       mod.sub(num);
295       ret[i]++;
296     }
297   }
298   if (!ret.back())ret.pop_back();
299   return ret;
300 }
301 Number Number::divInt(UInt num, UInt& mod)const
302 {
303   Number ret;
304   ret.resize(size());
305   ULL sum = 0;
306   for (int i = size() - 1; i >= 0; i--){
307     sum = link(sum, (*this)[i]);
308     ret[i] = (UInt)(sum / num);
309     sum %= num;
310   }
311   if (ret.size() && !ret.back())ret.pop_back();
312   mod = (UInt)sum;
313   return ret;
314 }
315 Number Number::operator / (const Number& num)const
316 {
317   UInt t;
```

```
318   Number ret = num.size() == 1 ? divInt(num[0], t) : div(num,
      ↪  Number());
319   if (ret.size())ret.flag = flag ^ num.flag;
320   return ret;
321 }
322 Number Number::operator % (const Number& num)const
323 {
324   Number ret;
325   if (num.size() == 1){
326     UInt t;
327     divInt(num[0], t);
328     ret = t;
329   }
330   else div(num, ret);
331   if (ret.size())ret.flag = flag;
332   return ret;
333 }
334 void Number::shr(UInt num)
335 {
336   if (!num)return;
337   UInt t = num / 31, k = num % 31;
338   if (size() <= t)clear();
339   else{
340     UInt newSize = size() - t;
341     for (UInt i = 0; i < newSize - 1; i++)
342       (*this)[i] = ((*this)[i + t] >> k) | ((*this)[i + t + 1] <<
          ↪  (31 - k)) & 0x7fffffff;
343     (*this)[newSize - 1] = back() >> k;
344     resize(newSize);
345     if (!back())pop_back();
346   }
347 }
348 void Number::shr31(UInt num)
349 {
350   if (size() <= num)clear();
351   else{
352     UInt newSize = size() - num;
353     for (UInt i = 0; i < newSize; i++)
354       (*this)[i] = (*this)[i + num];
355     resize(newSize);
356   }
357 }
358 void Number::shl(UInt num)
359 {
360   if (empty() || !num)return;
361   UInt t = (num + 30) / 31, k = (num + 30) % 31 + 1;
```

```
362   UInt oldSize = size();
363   resize(oldSize + t);
364   for (int i = oldSize - 1; i >= 0; i--)
365     (*this)[i + t] = ((*this)[i + 1] << k) | ((*this)[i] >> (31 -
      ↪  k)) & 0x7fffffff;
366   (*this)[t - 1] = (front() << k) & 0x7fffffff;
367   for (int i = t - 2; i >= 0; i--)
368     (*this)[i] = 0;
369   if (!back())pop_back();
370 }
371 void Number::shl31(UInt num)
372 {
373   if (empty())return;
374   UInt oldSize = size();
375   resize(oldSize + num);
376   for (int i = oldSize - 1; i >= 0; i--)
377     (*this)[i + num] = (*this)[i];
378   for (int i = num - 1; i >= 0; i--)
379     (*this)[i] = 0;
380 }
381 Number Number::operator >> (UInt num)const
382 {
383   bool f = false;
384   Number ret = *this;
385   if (flag){
386     UInt i, t;
387     for (i = 0; !(*this)[i]; i++);
388     t = i * 31;
389     f = t < num && ((t + 31 <= num) || ((1 << (num - t)) - 1) &
      ↪  (*this)[i]);
390   }
391   ret.shr(num);
392   if (f)ret.add(1);
393   return ret;
394 }
395 Number Number::operator << (UInt num)const
396 {
397   Number ret = *this;
398   ret.shl(num);
399   return ret;
400 }
401 void Number::and(const Number& num)
402 {
403   if (size() > num.size())resize(num.size());
404   for (UInt i = 0; i < size(); i++)
405     (*this)[i] &= num[i];
```

```
406   while (size() && !back())pop_back();
407 }
408 //为提高效率确保 *this 位数 >=num 位数
409 void Number::or(const Number& num)
410 {
411   if (size() < num.size())resize(num.size());
412   for (UInt i = 0; i < num.size(); i++)
413     (*this)[i] |= num[i];
414 }
415 Number Number::not(UInt k)const
416 {
417   Number ret;
418   ret.resize(k, ~(1 << 31));
419   for (int i = min(size(), k) - 1; i >= 0; i--)
420     ret[i] = (*this)[i] ^ ~(1 << 31);
421   while (ret.size() && !ret.back())ret.pop_back();
422   return ret;
423 }
424 //为提高效率确保 *this 位数 >=num 位数
425 void Number::xor(const Number& num)
426 {
427   if (size() < num.size())resize(num.size());
428   for (UInt i = 0; i < num.size(); i++)
429     (*this)[i] ^= num[i];
430   while (size() && !back())pop_back();
431 }
432 Number Number::operator & (const Number& num)const
433 {
434   Number ret;
435   if (flag != num.flag){
436     if (flag){
437       ret = not(num.size()); ret.add(1);
438       ret.and(num);
439     }
440     else{
441       ret = num.not(size()); ret.add(1);
442       ret.and(*this);
443     }
444   }
445   else if (flag){
446     Number temp;
447     if (size() < num.size()){ ret = num; temp = *this; }
448     else{ ret = *this; temp = num; }
449     ret.sub(1); temp.sub(1);
450     ret.or(temp); ret.add(1);
```

```
451     }
452     else if (size() < num.size()){ ret = *this; ret.and(num); }
453     else{ ret = num; ret.and(*this); }
454     ret.flag = flag & num.flag;
455     return ret;
456   }
457   Number Number::operator | (const Number& num)const
458   {
459     Number ret;
460     if (flag != num.flag){
461       Number temp;
462       if (flag){ ret = num.not(size()); temp = *this; }
463       else{ ret = not(num.size()); temp = num; }
464       temp.sub(1);
465       ret.and(temp); ret.add(1);
466     }
467     else if (flag){
468       Number temp;
469       if (size() < num.size()){ ret = *this; temp = num; }
470       else{ ret = num; temp = *this; }
471       ret.sub(1); temp.sub(1);
472       ret.and(temp); ret.add(1);
473     }
474     else if (size() < num.size()){ ret = num; ret.or(*this); }
475     else{ ret = *this; ret.or(num); }
476     ret.flag = flag | num.flag;
477     return ret;
478   }
479   Number Number::operator ^ (const Number& num)const
480   {
481     Number ret;
482     if (flag != num.flag){
483       if (flag){
484         ret = *this; ret.sub(1);
485         ret.xor(num);
486       }
487       else{
488         ret = num; ret.sub(1);
489         ret.xor(*this);
490       }
491       ret.add(1);
492     }
493     else if(flag){
494       Number temp;
495       if (size() < num.size()){ ret = num; temp = *this; }
496       else{ ret = *this; temp = num; }
```

```
497       ret.sub(1); temp.sub(1);
498       ret.xor(temp);
499     }
500     else if (size() < num.size()){ ret = num; ret.xor(*this); }
501     else{ ret = *this; ret.xor(num); }
502     ret.flag = flag ^ num.flag;
503     return ret;
504   }
```

9.3 math

```
#include<bits/stdc++.h>

const int maxn=1005;
const double eps=1e-8;
#define LL long long int

using namespace std;

int phi[maxn];

void swap(double& p,double& q)
{
  double t;
  t=p;p=q;q=t;
}

struct Matrix
{
  double a[maxn][maxn];
  //1-n 行表示第 1-n 个方程
  //每行第 1-n 个元素表示系数，第 n+1 个元素表示等号右边的常数
}q;

int ii,jj,nn;

LL det(LL a[][maxn], int n) {//求行列式值（整数版）
    int i, j;
    LL res = 1;
    for (i = 0; i < n; i++) {
        for (j = i + 1; j < n; j++) {
            while (a[j][i]) {
                LL f = a[i][i] / a[j][i];
                for (int k = i; k < n; k++) a[i][k] -= f *
                 ↪  a[j][k];
                for (int k = i; k < n; k++) swap(a[i][k],
                 ↪  a[j][k]);
                res = -res;
            }
        }
        if (a[i][i] == 0) return 0;
        res *= a[i][i];
    }
    return res < 0 ? -res : res;
}
```

```
43
44  double FF(double x)//需积分的函数，自行修改
45  {
46    return 1.0;
47  }
48
49  double simpson(double x,double y)
50  {
51    double z=x+(y-x)/2.0;
52    return (y-x)/6.0*(FF(x)+FF(y)+4*FF(z));
53  }
54
55  double asr(double x,double y,double eeps,double A)//eeps 为精度
56  {
57    double z=x+(y-x)/2.0;
58    double L=simpson(x,z);
59    double R=simpson(z,y);
60    if(fabs(L+R-A)<=15*eeps)return (L+R)+(L+R-A)/15.0;
61    else return asr(x,z,eeps/2.0,L)+asr(z,y,eeps/2.0,R);
62  }
63
64  double simpson_zsx(double x,double y,double eeps)//自适应辛普森主函
    ↪ 数
65  {
66    return asr(x,y,eeps,simpson(x,y));
67  }
68
69  void gauss_eli(struct Matrix& p,int n)//高斯消元
70  {
71    int i,j,k,r;
72    for(i=1;i<=n;i++)
73    {
74      r=i;
75      for(j=i+1;j<=n;j++)
76        if(fabs(p.a[j][i])>fabs(p.a[r][i]))r=j;
77      if(r!=i)for(j=1;j<=n+1;j++)swap(p.a[r][j],p.a[i][j]);
78      for(k=1;k<=i-1;k++)
79      {
80        if(p.a[i][k]==0)continue;
81        for(j=n+1;j>=k;j--)
82          p.a[i][j]-=p.a[k][j]/p.a[k][k]*p.a[i][k];
83      }
84    }
85    for(i=n;i>=1;i--)
86    {
```

```
87      for(j=i+1;j<=n;j++)
88        p.a[i][n+1]-=p.a[j][n+1]*p.a[i][j];
89      p.a[i][n+1]/=p.a[i][i];
90    }
91  }
92
93  LL gcd(LL a,LL b)
94  {
95    return b==0?a:gcd(b,a%b);
96  }
97
98  void tgcd(LL a,LL b,LL& d,LL& x,LL& y)//拓展欧几里德
99  {
100   if(!b){d=a;x=1;y=0;}
101   else{tgcd(b,a%b,d,y,x);y-=x*(a/b);}
102 }
103
104 LL pow_mod(LL a,LL p,LL n)//同余快速幂
105 {
106   if(p==0)return 1;
107   LL ans=pow_mod(a,p/2,n);
108   ans=(ans*ans)%n;
109   if(p%2==1)ans=(ans*a)%n;
110   return ans;
111 }
112
113 int euler_phi(int n)//求欧拉函数
114 {
115   int m=(int)sqrt(n+0.5);
116   int ans=n;
117   for(int i=2;i<=m;i++)
118     if(n%i==0)
119     {
120       ans=ans/i*(i-1);
121       while(n%i==0)n=n/i;
122     }
123   if(n>1)ans=ans/n*(n-1);
124   return ans;
125 }
126
127 void phi_table(int n)//欧拉函数表
128 {
129   memset(phi,0,n+1);
130   phi[1]=1;
131   for(int i=2;i<=n;i++)
132   {
```

```
133       if(phi[i])continue;
134       for(int j=i;j<=n;j+=i)
135       {
136         if(!phi[j])phi[j]=j;
137         phi[j]=phi[j]/i*(i-1);
138       }
139     }
140   }
141
142   LL inv(LL a,LL n)//a 关于 n 的逆元
143   {
144     LL d,x,y;
145     tgcd(a,n,d,x,y);
146     return d==1?(x+n)%n:-1;
147   }
148
149   // x mod m0=a0,x mod m =a,noSolution return 0
150   //初始可令 m0 = 1 ,a0 = 0
151   //布尔值返回是否有解
152   //m0,m 可以不互质
153   //若有多个方程，做多次此剩余定理
154   //m0，a0 返回答案
155   bool _china(LL &m0,LL &a0,LL m,LL a)
156   {
157       LL g,x,y;
158       LL c=abs(a-a0);
159       tgcd(m0,m,g,x,y);
160       if ( c % g ) return 0;
161       x*=(a-a0)/g;
162       x%=m/g;
163       a0=x*m0+a0;
164       m0*=m/g;
165       a0%=m0;
166       if(a0<0) a0+=m0;
167       return 1;
168   }
169
170   LL china(int n,int* a,int* m)//中国剩余定理
171   {
172     LL M=1,d,y,x=0;
173     for(int i=0;i<n;i++)M*=m[i];
174     for(int i=0;i<n;i++)
175     {
176       LL w=M/m[i];
177       tgcd(m[i],w,d,d,y);
```

```
178       x=(x+y*w*a[i])%M;
179     }
180     return (x+M)%M;
181   }
182
183   int log_mod(int a,int b,int n)//求解模方程 a^x=b(mod n),n 为素数,
      ↪ 无解返回-1
184   {
185     int m,v,e=1,i;
186     m=(int)sqrt(n+0.5);
187     v=inv(pow_mod(a,m,n),n);
188     map<int,int> x;
189     x[1]=0;
190     for(i=1;i<m;i++)
191     {
192       e=(e*a)%n;
193       if(!x.count(e))x[e]=i;
194     }
195     for(i=0;i<m;i++)
196     {
197       if(x.count(b))return (i*m+x[b]);
198       b=(b*v)%n;
199     }
200     return -1;
201   }
```

9.4 Matrix Fast Mi

```
struct mat
{
    int n;
    LL num[105][105];

    void init0(int t)
    {
        n=t;
        for(int i=0;i<=n;i++)
            for(int j=0;j<=n;j++)
                num[i][j]=0;
    }

    void init1(int t)
    {
        n=t;
        for(int i=0;i<=n;i++)
            for(int j=0;j<=n;j++)
                if(i!=j)num[i][j]=0;else num[i][j]=1;
    }

    mat operator * (const struct mat p)const
    {
        struct mat ans;
        ans.init0(n);
        for(int i=1;i<=n;i++)
            for(int j=1;j<=n;j++)
                for(int k=1;k<=n;k++)

                    ↪ ans.num[i][j]=(ans.num[i][j]+num[i][k]*p.num[k][j])%mod;
        //printf("??");ans.testprint();
        return ans;
    }

    mat operator ^ (int t)const
    {
        struct mat ans,now;
        ans.init1(n);
        now.n=n;
        for(int i=0;i<=n;i++)
            for(int j=0;j<=n;j++)
                now.num[i][j]=num[i][j];
        while(t>0)
        {
```

```
44                 if(t&1)ans=ans*now;
45                 now=now*now;
46                 t>>=1;
47             }
48             return ans;
49         }
50 
51 };
```

9.5 NTT CRT

```cpp
#include<cstdio>
#include<cmath>
#include<algorithm>
#include<vector>
#include<cstring>
using namespace std;
int len, bit;
int MOD, w[2][32];
inline int add(int a, int b){
  return a + b - (a + b >= MOD ? MOD : 0);
}
inline int sub(int a, int b){
  return a - b + (a - b < 0 ? MOD : 0);
}
inline int mul(int a, int b){
  return (long long)a * b % MOD;
}
int power(int a, int b){
  int ret = 1;
  for (int t = a; b; b >>= 1){
    if (b & 1)ret = mul(ret, t);
    t = mul(t, t);
  }
  return ret;
}
int cal_root(int mod)
{
  for (int i = 2;; i++){
    if (power(i, (mod - 1) / 2) == mod - 1)
      return i;
  }
}
void fft_init(int n, int mod)
{
  MOD = mod;
  bit = (int)log2(n - 0.5) + 2;
  len = 1 << bit;
  w[0][0] = power(cal_root(mod), (mod - 1) / len);
  int i;
  for (i = 1; i < bit; i++)
    w[0][i] = mul(w[0][i - 1], w[0][i - 1]);
  i--;
  w[1][i] = w[0][i];
  for (i--; i >= 0; i--)
```

```
45      w[1][i] = mul(w[1][i + 1], w[0][i]);
46    }
47    void bitReverse(int a[]) {
48      for (int i = 1, j = len / 2; i < len - 1; i++) {
49        if (i < j) swap(a[i], a[j]);
50        int k = len / 2;
51        while (j >= k) { j -= k; k >>= 1; }
52        if (j < k) j += k;
53      }
54    }
55    void fft_main(int a[], bool reverse)
56    {
57      bitReverse(a);
58      for (int i = 1, s = 1; s < len; i++, s <<= 1){
59        int step = w[reverse][bit - i];
60        for (int j = 0; j < len; j += 2 * s){
61          int cur = 1;
62          for (int k = j; k < j + s; k++){
63            int u = a[k], t = mul(cur, a[k + s]);
64            a[k] = add(u, t);
65            a[k + s] = sub(u, t);
66            cur = mul(cur, step);
67          }
68        }
69      }
70      if (reverse){
71        int t = power(len, MOD - 2);
72        for (int i = 0; i < len; i++)
73          a[i] = mul(a[i], t);
74      }
75    }
76    //确保数组中的数小于 mod(mod<2^30)，数组需留足 2^(logn 向上取整 +1)
      ↪ 的空间，后面填充 0
77    //并且 mod 为形如 m*2^k+1 的素数，2^k>=2*n
78    void fft(int a[], int b[], int n, int mod)
79    {
80      fft_init(n, mod);
81      fft_main(a, 0); fft_main(b, 0);
82      for (int i = 0; i < len; i++)
83        a[i] = mul(a[i], b[i]);
84      fft_main(a, 1);
85    }
86    //确保 mod 两两互质，retmod 任意
87    void chineseRemainder(const int mod[], int *a[], int ret[], int
      ↪ num, int n, int retMod)
```

```
{
  int kk[30], mulMod[30][30], mulModr[30], mulretMod[30];
  for (int i = 0; i < num; i++){
    MOD = mod[i]; mulMod[i][0] = 1;
    for (int j = 1; j <= i; j++)
      mulMod[i][j] = mul(mulMod[i][j - 1], mod[j - 1]);
    mulModr[i] = power(mulMod[i][i], MOD - 2);
  }
  mulretMod[0] = 1; MOD = retMod;
  for (int i = 1; i < num; i++)
    mulretMod[i] = mul(mulretMod[i - 1], mod[i - 1]);
  for (int i = 0; i < n; i++){
    for (int j = 1; j < num; j++){
      MOD = mod[j];
      int sum = a[0][i] % MOD;
      for (int k = 1; k < j; k++)
        sum = add(sum, mul(mulMod[j][k], kk[k]));
      kk[j] = mul(sub(a[j][i] % MOD, sum), mulModr[j]);
    }
    MOD = retMod;
    ret[i] = a[0][i] % MOD;
    for (int j = 1; j < num; j++)
      ret[i] = add(ret[i], mul(kk[j] % MOD, mulretMod[j]));
  }
}
//附满足条件大整数：167772161, 469762049, 754974721
```

10 Prime

10.1 Euler Prime

```cpp
#include<cstdio>
#define MAXN 10000001
int minFactor[MAXN];
int prime[2000000], primeNum;
int phi[MAXN];
void calPrime()
{
  for (int i = 2; i < MAXN; i++){
    if (!minFactor[i]){
      prime[primeNum++] = i;
      minFactor[i] = primeNum;
    }
    for (int j = 1; j <= minFactor[i]; j++){
      int t = i * prime[j - 1];
      if (t >= MAXN)break;
      minFactor[t] = j;
    }
  }
}
void calPhi()
{
  phi[1] = 1;
  for (int i = 2; i < MAXN; i++){
    if (!minFactor[i]){
      prime[primeNum++] = i;
      minFactor[i] = primeNum;
      phi[i] = i - 1;
    }
    for (int j = 1;; j++){
      int t = i * prime[j - 1];
      if (t >= MAXN)break;
      minFactor[t] = j;
      if (j == minFactor[i]){
        phi[t] = phi[i] * prime[j - 1];
        break;
      }
      phi[t] = phi[i] * (prime[j - 1] - 1);
    }
  }
}
```

10.2 Miller-Rabin Pollard

```cpp
#include<cstdio>
#include<typeinfo>
#include<cstdlib>
#include<algorithm>
using namespace std;
typedef unsigned long long ULL;
typedef unsigned int UInt;
const UInt base1[] = { 2, 7, 61, 0 };
const UInt base2[] = { 2, 325, 9375, 28178, 450775, 9780504, 1795265022, 0 };
const UInt prime[] = { 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53 };
template <typename T>
inline T add(T a, T b, T mod){
  return a + b - (a + b >= mod ? mod : 0);
}
inline UInt mul(UInt a, UInt b, UInt mod){
  return (ULL)a * b % mod;
}
ULL mul(ULL a, ULL b, ULL mod){
  ULL ret = 0;
  for (ULL t = a; b; b >>= 1){
    if (b & 1)ret = add(ret, t, mod);
    t <<= 1;
    if (t >= mod)t -= mod;
  }
  return ret;
}
template <typename T>
T power(T a, T b, T mod){
  T ret = 1;
  for (T t = a; b; b >>= 1){
    if (b & 1)ret = mul(ret, t, mod);
    t = mul(t, t, mod);
  }
  return ret;
}
//n 为小于 2^63 的非 1 奇数，正确性 100%
template <typename T>
bool millerRabin(T n)
{
  int s = 0;
  T r = n;
  for (r--; !(r & 1); r >>= 1)s++;
```

```
43    for (const UInt *base = typeid(T) == typeid(UInt) ? base1 :
      ↪  base2; *base; base++){
44      T t = power(*base % n, r, n);
45      if (t == 0 || t == 1 || t == n - 1)continue;
46      for (int j = 1; j < s; j++){
47        t = mul(t, t, n);
48        if (t == 1)return false;
49        if (t == n - 1)break;
50      }
51      if (t != n - 1)return false;
52    }
53    return true;
54  }
55  template <typename T>
56  bool checkPrime(T n)
57  {
58    if (n == 1)return false;
59    for (int i = 0; i < sizeof(prime) / sizeof(int); i++){
60      if (n % prime[i] == 0)return n == prime[i];
61    }
62    return millerRabin(n);
63  }
64  template <typename T>
65  T gcd(T x, T y){
66    return y ? gcd(y, x % y) : x;
67  }
68  template <typename T>
69  T pollard(T n)
70  {
71    if (millerRabin(n))return n;
72    while (1){
73      T x = rand() % n, y = x, c = rand() % (n - 1) + 1;
74      for (UInt i = 1, j = 2;; i++){
75        if (i == j){ j *= 2; y = x; }
76        x = add(mul(x, x, n), c, n);
77        T d = gcd(x - y + n, n);
78        if (d != 1){
79          if (d != n)return d;
80          break;
81        }
82      }
83    }
84  }
85  ULL factor[64];
86  int factorNum;
87  void calFactorInternal(ULL n)
```

```
88  {
89    ULL d;
90    d = n >> 32 ? pollard(n) : pollard((UInt)n);
91    if (d == n){ factor[factorNum++] = d; return; }
92    calFactorInternal(d);
93    calFactorInternal(n / d);
94  }
95  void calFactor(ULL n)
96  {
97    factorNum = 0;
98    for (int i = 0; i < sizeof(prime) / sizeof(int); i++){
99      while (n % prime[i] == 0){
100       n /= prime[i];
101       factor[factorNum++] = prime[i];
102     }
103   }
104   if (n != 1)calFactorInternal(n);
105   sort(factor, factor + factorNum);
106 }
```

11 String

11.1 AC Automation

```cpp
#include<cstdio>
#include<cstring>
#include<queue>
using namespace std;
#define LETTER 26
struct Trie{
  int num, fail, match;
  int next[LETTER];
}trie[500001];
int cnt;
void init(){
  cnt = 1;
  memset(trie, 0, 2 * sizeof(Trie));
  trie[0].fail = 1;
}
inline int convert(char ch){ return ch - 'a'; }
void insert(char *s)
{
  int cur = 0;
  for (int i = 0; s[i]; i++){
    int &pos = trie[cur].next[convert(s[i])];
    if (!pos){
      pos = ++cnt;
      memset(&trie[cnt], 0, sizeof(Trie));
    }
    cur = pos;
  }
  trie[cur].num++;
}
void makeFail()
{
  queue<int> q; q.push(0);
  while (!q.empty()){
    int t = q.front(); q.pop();
    for (int i = 0; i < LETTER; i++){
      int &cur = trie[t].next[i];
      if (cur){
        q.push(cur);
        trie[cur].fail = trie[trie[t].fail].next[i];
        trie[cur].match = trie[cur].num ? cur :
        ↪  trie[trie[cur].fail].match;
```

```
41          }
42          else cur = trie[trie[t].fail].next[i];
43        }
44      }
45    }
46    int search(char *s)
47    {
48      int ret = 0, cur = 0;
49      for (int i = 0; s[i]; i++){
50        cur = trie[cur].next[convert(s[i])];
51        for (int temp = trie[cur].match; temp; temp =
          ↪  trie[trie[temp].fail].match)
52          ret += trie[temp].num;
53      }
54      return ret;
55    }
```

11.2 AC-Auto(Compressed)

```
#define LETTER 26
struct Trie{
  int num, next, fail;
}trie[1000000];
int cnt;
int pool[LETTER * 200000], poolEnd;
void init()
{
  cnt = 0;
  trie[0].num = 0;
  trie[0].next = -1;
  memset(pool, 0, sizeof(pool));
  poolEnd = 0;
}
inline int convert(char ch){ return ch - 'a'; }
inline bool oneBranch(int value){ return value < LETTER; }
inline int child(int i, int ch){
  if (oneBranch(trie[i].next))return trie[i].next == ch ? i + 1 :
   ↪  0;
  return pool[trie[i].next + ch];
}
void insert(char *s)
{
  int pos = 0, i;
  for (i = 0; s[i]; i++){
    int t = trie[pos].next;
    if (oneBranch(t)){
      if (t == convert(s[i]))pos++;
      else{
        trie[pos].next = (poolEnd += LETTER);
        if (t != -1)pool[trie[pos].next + t] = pos + 1;
        break;
      }
    }
    else if (pool[t + convert(s[i])])
      pos = pool[t + convert(s[i])];
    else break;
  }
  if (s[i]){
    pool[trie[pos].next + convert(s[i])] = ++cnt;
    for (i++; s[i]; i++, cnt++){
      trie[cnt].num = 0;
      trie[cnt].next = convert(s[i]);
    }
```

```
44      trie[cnt].num = 1;
45      trie[cnt].next = -1;
46    }
47    else trie[pos].num++;
48  }
49  int getFailPoint(int father, int ch)
50  {
51    while (father){
52      father = trie[father].fail;
53      int pos = child(father, ch);
54      if (pos)return pos;
55    }
56    return 0;
57  }
58  void makeFail()
59  {
60    queue<int> q; q.push(0);
61    trie[0].fail = 0;
62    while (!q.empty()){
63      int t = q.front(); q.pop();
64      if (oneBranch(trie[t].next)){
65        if (trie[t].next != -1){
66          trie[t + 1].fail = getFailPoint(t, trie[t].next);
67          q.push(t + 1);
68        }
69      }
70      else for (int i = 0; i < LETTER; i++){
71        int cur = pool[trie[t].next + i];
72        if (cur){
73          trie[cur].fail=getFailPoint(t, i);
74          q.push(cur);
75        }
76      }
77    }
78  }
79  //统计匹配总次数，包括母串多次匹配同一模式串或多个模式串相同
80  int search(char *s)
81  {
82    int ret = 0, cur = 0;
83    for (int i = 0; s[i]; i++){
84      int ch = convert(s[i]);
85      for (; cur && !child(cur, ch); cur = trie[cur].fail);
86      cur = child(cur, ch);
87      for (int temp = cur; temp; temp = trie[temp].fail)
88        ret += trie[temp].num;
```

```
89      }
90      return ret;
91  }
```

11.3 KMP

```
int nxt[maxn];
char origin_string[maxn];
char target_string[maxn];
void get_nxt()
{
  int n=strlen(target_string);
  nxt[0]=0;nxt[1]=0;
  for (int i=1;i<n;i++)
  {
    int j=nxt[i];
    while(j&&target_string[i]!=target_string[j]) j=nxt[j];
    nxt[i+1]=target_string[i]==target_string[j]?j+1:0;
  }
}
int kmp()
{
  int n=strlen(origin_string);
  int m=strlen(target_string);
  int j=0,cnt=0;
  for (int i=0;i<n;i++)
  {
    while(j&&origin_string[i]!=target_string[j]) j=nxt[j];
    if (origin_string[i]==target_string[j]) j++;
    if (j==m) {cnt++;j=nxt[j];}
  }
  return cnt;
}
int main()
{
  int _;
  scanf("%d",&_);
  while(_--)
  {
    scanf("%s",target_string);
    scanf("%s",origin_string);
    get_nxt();
    printf("%d\n",kmp());
  }
  return 0;
}
```

11.4 LCS

```cpp
#include<algorithm>
using namespace std;
int maxCommonSubstring(char *s1, char *s2)
{
  init();
  suffixAutomation(s1);
  int match = 0, ret = 0, cur = 0;
  for (int i = 0; s2[i]; i++){
    char c = convert(s2[i]);
    if (!st[cur].next[c]){
      while (cur != -1 && !st[cur].next[c])
        cur = st[cur].link;
      if (cur == -1){ match = cur = 0; continue; }
      match = st[cur].len;
    }
    cur = st[cur].next[c];
    ret = max(ret, ++match);
  }
  return ret;
}
```

11.5 manacher

```
char s1[1000005],s2[2000010];
int p[2000010];
int i,j,k,l,m,n;
int mx,id;

int min(int x,int y)
{
  return x<y?x:y;
}

int main()
{
  gets(s1);l=0;
  while(s1[0]!='E')
  {
    l++;
    n=strlen(s1);
    s2[0]='$';k=0;
    for(i=0;i<n;i++)
    {
      s2[++k]='#';
      s2[++k]=s1[i];
    }
    s2[++k]='#';s2[++k]='\0';
    memset(p,0,sizeof(p));
    mx=0;id=0;
    for(i=1;s2[i]!='\0';i++)
    {
      p[i]=mx>i?min(p[2*id-i],mx-i):1;
      while(s2[i+p[i]]==s2[i-p[i]])p[i]++;
      if(i+p[i]>mx)
      {
        mx=i+p[i];id=i;
      }
    }
    mx=0;
    for(i=1;s2[i]!='\0';i++)if(p[i]-1>mx)mx=p[i]-1;
    printf("Case %d: %d\n",l,mx);
    gets(s1);
  }
  return 0;
}
```

11.6 palindrome automation

```
struct node
{
  int len,sum;
  node* fail,*next[26];
}mem[100005],*headf,*heads,*last;
int tot,now;
char s[100005];

void init()
{
  memset(mem,0,sizeof(mem));
  headf=mem;last=heads=mem+1;
  headf->fail=heads;heads->len=-1;
  tot=1;now=0;
}
void add(int x,int p)
{
  node* cur=last;
  for (;s[p-cur->len-1]!=s[p];cur=cur->fail);
  if (!cur->next[x])
  {
    node* ths=&mem[++tot];
    last=cur->next[x]=ths;
    ths->len=cur->len+2;
    if (cur==heads) ths->fail=headf;
    else
    {
      for (cur=cur->fail;s[p-cur->len-1]!=s[p];cur=cur->fail);
      ths->fail=cur->next[x];
    }
    ths->sum=ths->fail->sum+1;
  }
  else last=cur->next[x];
}
//HDU 5157
long long l[100005],r[100005];
int main()
{
  while(~scanf("%s",s))
  {
    int n=strlen(s);
    init();
    for (int i=0;i<n;i++) {add(s[i]-'a',i);l[i]=last->sum;}
    reverse(s,s+n);
```

```
45      init();
46      for (int i=0;i<n;i++) {add(s[i]-'a',i);r[i]=last->sum+r[i-1];}
47      long long ans=0;
48      for (int i=0;i<n-1;i++) ans+=l[i]*r[n-i-2];
49      printf("%I64d\n",ans);
50    }
51    return 0;
52  }
```

11.7 SAM(Compressed)

```cpp
#include<cstdio>
#include<cstring>
using namespace std;
#define MAXN 1000001
#define LETTER 26
int pool[MAXN * LETTER / 3], poolNum;
struct State{
  int len, link;
  char ch[4];
  int next[3];
  int* child(char c){
    if (ch[0] == -1)
      return pool[next[0] + c] ? &pool[next[0] + c] : 0;
    for (int i = ch[0]; i; i--){
      if (ch[i] == c)return &next[i - 1];
    }
    return 0;
  }
  void insert(char c, int pos){
    if (ch[0] == 3){
      ch[0] = -1;
      memset(pool + poolNum, 0, sizeof(int)*LETTER);
      for (int i = 3; i; i--)
        pool[poolNum + ch[i]] = next[i - 1];
      next[0] = poolNum;
      poolNum += LETTER;
    }
    if (ch[0] == -1)pool[next[0] + c] = pos;
    else{
      next[ch[0]] = pos;
      ch[++ch[0]] = c;
    }
  }
}st[MAXN * 2];
int cnt, last;
void init()
{
  last = cnt = 0;
  st[cnt].len = 0; st[cnt].link = -1;
  st[cnt].ch[0] = 0;
  poolNum = 0;
}
inline int convert(char ch){ return ch - 'a'; }
void add(char c)
```

```
{
  c = convert(c);
  int cur = ++cnt, i, *tmp;
  st[cur].len = st[last].len + 1;
  st[cur].ch[0] = 0;
  for (i = last; i != -1 && !(tmp = st[i].child(c)); i =
  ↪  st[i].link)
    st[i].insert(c, cur);
  if (i == -1)st[cur].link = 0;
  else{
    int j = *tmp;
    if (st[i].len + 1 == st[j].len)
      st[cur].link = j;
    else{
      int copy = ++cnt;
      st[copy].len = st[i].len + 1;
      if (st[j].ch[0] == -1){
        st[copy].ch[0] = -1;
        st[copy].next[0] = poolNum;
        memcpy(pool + poolNum, pool + st[j].next[0],
        ↪  sizeof(int)*LETTER);
        poolNum += LETTER;
      }
      else{
        for (int i = 0; i < 4; i++)
          st[copy].ch[i] = st[j].ch[i];
        for (int i = st[j].ch[0]; i; i--)
          st[copy].next[i - 1] = st[j].next[i - 1];
      }
      st[copy].link = st[j].link;
      for (; i != -1 && (tmp = st[i].child(c)) && *tmp == j; i =
      ↪  st[i].link)
        *tmp = copy;
      st[j].link = st[cur].link = copy;
    }
  }
  last = cur;
}
void suffixAutomation(char *s)
{
  init();
  for (int i = 0; s[i]; i++)
    add(s[i]);
}
```

11.8 SAM

```cpp
#include<cstdio>
#include<cstring>
using namespace std;
#define MAXN 1000001
#define LETTER 26
struct State{
  int len, link;
  int next[LETTER];
}st[MAXN * 2];
int tree[MAXN];
int cnt, last;
void init()
{
  last = cnt = 0;
  st[cnt].len = 0; st[cnt].link = -1;
  memset(st[0].next, 0, sizeof(st[0].next));
}
inline int convert(char ch){ return ch - 'a'; }
void add(char c)
{
  c = convert(c);
  int cur = ++cnt, i;
  st[cur].len = st[last].len + 1;
  memset(st[cur].next, 0, sizeof(st[cur].next));
  for (i = last; i != -1 && !st[i].next[c]; i = st[i].link)
    st[i].next[c] = cur;
  if (i == -1)st[cur].link = 0;
  else{
    int j = st[i].next[c];
    if (st[i].len + 1 == st[j].len)
      st[cur].link = j;
    else{
      int copy = ++cnt;
      st[copy].len = st[i].len + 1;
      memcpy(st[copy].next, st[j].next, sizeof(st[j].next));
      st[copy].link = st[j].link;
      for (; i != -1 && st[i].next[c] == j; i = st[i].link)
        st[i].next[c] = copy;
      st[j].link = st[cur].link = copy;
    }
  }
  last = cur;
}
void suffixAutomation(char *s)
```

```
45  {
46    init();
47    for (int i = 0; s[i]; i++)
48      add(s[i]);
49  }
50  void suffixTree(char *s)
51  {
52    init();
53    for (int i = strlen(s) - 1; i >= 0; i--){
54      add(s[i]);
55      tree[i] = last;
56    }
57  }
```

11.9 Suffix Array

```cpp
#include<cstdio>
#include<cstring>
using namespace std;
#define MAXN 2000001
#define CHAR 256
int bucket[MAXN];
//#define DC3
#ifndef DC3
int r[2][MAXN];
int tmp[MAXN], sa[MAXN];//存储第 i 小的字符位置
int *rk;//存储位置 i 上的字符是第几小的
int suffixArray(unsigned char *s)
{
  int len = 0, m = 0;
  int *r1 = r[0], *r2 = r[1];
  memset(bucket, 0, sizeof(int)*CHAR);
  while (bucket[s[len]]++, s[len++]);
  for (int i = 1; i < CHAR; i++)
    bucket[i] += bucket[i - 1];
  for (int i = len - 1; i >= 0; i--)
    sa[--bucket[s[i]]] = i;
  for (int i = 0, k = 0; i < CHAR; i++){
    if (bucket[i]>k){ k = bucket[i]; m++; }
    bucket[i] = m;
  }
  for (int i = 0; i < len; i++)
    r1[i] = bucket[s[i]];
  for (int i = 1; m < len; i *= 2){
    for (int j = 0; j < i; j++)
      tmp[j] = len - i + j;
    for (int j = i, k = 0; j < len; k++){
      if (sa[k] >= i)tmp[j++] = sa[k] - i;
    }
    memset(bucket, 0, sizeof(int)*m);
    for (int j = 0; j < len; j++)
      bucket[r1[j]]++;
    for (int j = 1; j < m; j++)
      bucket[j] += bucket[j - 1];
    for (int j = len - 1; j >= 0; j--)
      sa[--bucket[r1[tmp[j]]]] = tmp[j];
    r2[sa[0]] = 0; m = 0;
    for (int j = 1; j < len; j++){
      if (r1[sa[j]] != r1[sa[j - 1]] || r1[sa[j] + i] != r1[sa[j -
      ↪  1] + i])m++;
```

```
      r2[sa[j]] = m;
    }
    int *t = r1; r1 = r2; r2 = t;
    m++;
  }
  rk = r1;
  return len;
}
#else
int pool[3 * MAXN + 200], sa[3 * MAXN], rk[MAXN];
int tmp[MAXN];
void sort(int src[], int sa1[], int sa2[], int len, int m)
{
  memset(bucket, 0, sizeof(int)*m);
  for (int i = 0; i < len; i++)
    bucket[tmp[i] = src[sa1[i]]]++;
  for (int i = 1; i < m; i++)
    bucket[i] += bucket[i - 1];
  for (int i = len - 1; i >= 0; i--)
    sa2[--bucket[tmp[i]]] = sa1[i];
}
bool cmp(int r[], int r2[], int i, int j)
{
  if (r[i] != r[j])return r[i] < r[j];
  if (j % 3 == 1)return r2[i + 1] < r2[j + 1];
  if (r[i + 1] != r[j + 1])return r[i + 1] < r[j + 1];
  return r2[i + 2] < r2[j + 2];
}
#define F(pos) (pos % 3 == 1 ? pos / 3 : pos / 3 + t2)
#define G(pos) (pos < t2 ? pos * 3 + 1 : (pos - t2) * 3 + 2)
void merge(int r[], int sa[], int len, int m)
{
  int t1 = 0, t2 = (len + 1) / 3, t3 = len / 3, len2 = t2 + t3;
  int *r2 = r + len + 2, *sa2 = sa + len, *od = sa2 + len2;
  for (int i = 0, j = 0; i < len; i += 3, j += 2){
    sa[j] = i + 1;
    sa[j + 1] = i + 2;
  }
  r[len] = r[len + 1] = 0;
  sort(r + 2, sa, od, len2, m);
  sort(r + 1, od, sa, len2, m);
  sort(r, sa, od, len2, m);
  int k = 0;
  r2[F(od[0])] = 0;
  for (int i = 1; i < len2; i++){
    int pos1 = od[i], pos2 = od[i - 1];
```

```
90      if (r[pos1] != r[pos2] || r[pos1 + 1] != r[pos2 + 1] || r[pos1
        ↪  + 2] != r[pos2 + 2])k++;
91      r2[F(od[i])] = k;
92    }
93    if (++k < len2)merge(r2, sa2, len2, k);
94    else for (int i = 0; i < len2; i++)
95      sa2[r2[i]] = i;
96    if (len % 3 == 1)r2[t1++] = len - 1;
97    for (int i = 0; i < len2; i++){
98      if (sa2[i] < t2)r2[t1++] = sa2[i] * 3;
99    }
100   sort(r, r2, od, t1, m);
101   for (int i = 0; i < len2; i++)
102     r2[sa2[i] = G(sa2[i])] = i;
103   int ii = 0, jj = 0, kk = 0;
104   while (ii < t1 && jj < len2)
105     sa[kk++] = cmp(r, r2, od[ii], sa2[jj]) ? od[ii++] : sa2[jj++];
106   for (; ii < t1; ii++)sa[kk++] = od[ii];
107   for (; jj < len2; jj++)sa[kk++] = sa2[jj];
108 }
109 int suffixArray(unsigned char *s)
110 {
111   int len = 0;
112   while (pool[len] = s[len], s[len++]);
113   merge(pool, sa, len, CHAR);
114   return len;
115 }
116 #endif
117 int h[MAXN];
118 int getH(unsigned char *s)
119 {
120   int len = suffixArray(s) - 1;
121 #ifdef DC3
122   for (int i = 0; i <= len; i++)
123     rk[sa[i]] = i;
124 #endif
125   for (int i = 0, k = 0; i < len; i++){
126     int j = sa[rk[i] - 1];
127     while (s[i + k] == s[j + k])k++;
128     h[rk[i] - 1] = k;
129     if (k)k--;
130   }
131   return len;
132 }
```

11.10 Suffix Tree

```
#include<vector>
using namespace std;
int bucket[MAXN], order[MAXN];
int id[MAXN];
bool leave[MAXN];
vector<int> tree[MAXN];
void sortState()//ºÃÃÃÃÃÃÃÃÃÃÃÃ
{
  int size = st[last].len;
  memset(bucket, 0, sizeof(int)*(size + 1));
  for (int i = 0; i <= cnt; i++)
    bucket[st[i].len]++;
  for (int i = 1; i <= size; i++)
    bucket[i] += bucket[i - 1];
  for (int i = cnt; i; i--)
    order[--bucket[st[i].len]] = i;
}
void suffixTree(char *s)
{
  int len = strlen(s);
  init();
  memset(leave, 0, len * sizeof(bool) * 2);
  for (int i = len - 1; i >= 0; i--){
    add(s[i]);
    id[i] = last;
    leave[last] = true;
  }
  for (int i = 0; i <= cnt; i++)
    tree[i].clear();
  for (int i = cnt; i; i--)
    tree[st[i].link].push_back(i);
}
```

11.11 Trie(Compressed)

```cpp
#include<cstdio>
#include<cstring>
using namespace std;
#define LETTER 26
struct Trie{
  int num, next;
}trie[1000000];
int cnt;
int pool[LETTER * 200000], poolEnd;
void init()
{
  cnt = 0;
  trie[0].num = 0;
  trie[0].next = -1;
  memset(pool, 0, sizeof(pool));
  poolEnd = 0;
}
inline int convert(char ch){ return ch - 'a'; }
inline char convert2(int value){ return value + 'a'; }
inline bool oneBranch(int value){ return value < LETTER; }
inline int child(int i, int ch){
  if (oneBranch(trie[i].next))return trie[i].next == ch ? i + 1 :
  ↪  0;
  return pool[trie[i].next + ch];
}
void insert(char *s)
{
  int pos = 0, i;
  for (i = 0; s[i]; i++){
    int t = trie[pos].next;
    if (oneBranch(t)){
      if (t == convert(s[i]))pos++;
      else{
        trie[pos].next = (poolEnd += LETTER);
        if (t != -1)pool[trie[pos].next + t] = pos + 1;
        break;
      }
    }
    else if (pool[t + convert(s[i])])
      pos = pool[t + convert(s[i])];
    else break;
  }
  if (s[i]){
    pool[trie[pos].next + convert(s[i])] = ++cnt;
```

```
    for (i++; s[i]; i++, cnt++){
      trie[cnt].num = 0;
      trie[cnt].next = convert(s[i]);
    }
    trie[cnt].num = 1;
    trie[cnt].next = -1;
  }
  else trie[pos].num++;
}
int search(char* s)
{
  int pos = 0;
  for (int i = 0; s[i]; i++){
    pos = child(pos, convert(s[i]));
    if (!pos)return -1;
  }
  return trie[pos].num;
}
char temp[100];
void dfs(int i, int h)
{
  if (trie[i].num){
    temp[h] = 0;
    printf("%s %d\n", temp, trie[i].num);
  }
  int t = trie[i].next;
  if (oneBranch(t)){
    if (t == -1)return;
    temp[h] = convert2(t);
    dfs(i + 1, h + 1);
  }
  else for (int j = 0; j < LETTER; j++){
    if (pool[t + j]){
      temp[h] = convert2(j);
      dfs(pool[t + j], h + 1);
    }
  }
}
```

11.12 Trie

```cpp
#include<cstdio>
#include<cstring>
#define LETTER 26
struct Trie{
  int num;
  int next[LETTER];
}trie[500001];
int cnt;
void init(){
  cnt = 0;
  memset(trie, 0, sizeof(Trie));
}
inline int convert(char ch){ return ch - 'a'; }
inline char convert2(int value){ return value + 'a'; }
void insert(char *s)
{
  int cur = 0;
  for (int i = 0; s[i]; i++){
    int &pos = trie[cur].next[convert(s[i])];
    if (!pos){
      pos = ++cnt;
      memset(&trie[cnt], 0, sizeof(Trie));
    }
    cur = pos;
  }
  trie[cur].num++;
}
int search(char *s)
{
  int cur = 0;
  for (int i = 0; s[i]; i++){
    cur = trie[cur].next[convert(s[i])];
    if (!cur)return -1;
  }
  return trie[cur].num;
}
char temp[1001];
void dfs(int i, int h)
{
  if (trie[i].num){
    temp[h] = 0;
    printf("%s %d\n", temp, trie[i].num);
  }
  for (int j = 0; j < LETTER; j++){
```

```
45      if (trie[i].next[j]){
46        temp[h] = convert2(j);
47        dfs(trie[i].next[j], h + 1);
48      }
49    }
50  }
```

12 Others

12.1 Attention

比赛：测试 PE 是否被判定为 WA，以及是否开了 O2 优化（详情咨询陈铮）

要点：
发票抬头写西安交通大学!!!
留住所有发票！不能为收据，必须为机打发票！
去赛场报到时会拿到参赛费的发票，务必带回！
住宿不要超过三天
比赛不要乱吃！

香港：自行注意

请假条：
套模板之后交由徐宏喆老师或李玟老师签字，然后拿到计算机系办公室交由张华老师盖章，原件放在机房！
以及拿复印件请假。

12.2 Better

卡常数优化

getchar(),gets()
cache
手写队列、栈会比 STL 库快
尽量少用 memset，需要给多少清零就给多少清
利用 & 减少寻址（int &x=a[][][][]）
尽量减少初始化

12.3 operations

i 插入模式
esc 正常模式

:e xxx.cpp 编辑 xxx.cpp
:w 保存
:q 退出
:wq 保存并退出

ggvG 全选
y 复制
p 粘贴
u 撤销
"+y 复制到外部
"+p 从外部粘贴
正常模式下按 v 再按 ijkl 可选定指定区域
(vim 的剪切板和外部的剪切板貌似是分开的 ?)

(在 vimrc 正常工作的情况下)
F9 编译
F5 运行

(你们貌似都不用多屏 ?)
:split xxx.txt 横屏打开新文件 xxx.txt
:vsplit xxx.txt 竖屏打开新文件 xxx.txt
Ctrl+ww 移动到下一个窗口

12.4 Read

```
inline void read(int &x) {
    char ch=getchar();
    while(ch<'0'||ch>'9') ch=getchar();
    x=0;
    while(ch<='9'&&ch>='0'){
        x=x*10+ch-'0';
        ch=getchar();
    }
}
```

12.5 Tags

二分 离线 倒跑 并查集 DFS BFS

贪心 DP 递推 莫队

前缀和 快速幂 倍增！ 差分数列

单调栈

12.6 vimrc

```
map <F9> :! g++ % -o %< -g -lm -Wall && size %<.exe <CR>
map <F5> :! gdb %< <CR>
set cindent
set smartindent
set autoindent
set number
set ruler
set mouse=a
set bs=2
set tabstop=4
set softtabstop=4
set shiftwidth=4
set autoread
"set expandtab

syntax on
colo evening
```

12.7 Wrong

变量名打错！(数组开小)
忘了给变量或数组清零
程序逻辑错误（先 XX，再 XX）
n，m 打反