ACM Template

Forgive Her

wfgu, Weis, 500kg

Last build at October 23, 2017 by LaTeX

Contents

1 Datastructure

1.1 KD tree

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

1.2 Binary indexed tree

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

1.3 Splay

```
//Node
struct Node {
```

```
3   int size,key;
4   Node *c[2], *p;
5 } mem[MaxN], *cur, *nil;
6 //Initialize functions without memory pool
7 Node *newNode(int v, Node *p) {
8   cur->c[0] = cur->c[1] = nil, cur->p = p;
9   cur->size = 1;
10  cur->key = v;
11  return cur++;
12 }
13 void Init() {
14  cur = mem;
15  nil = newNode(0, cur);
16  nil->size = 0;
17 }
18 //Splay tree
19 struct SplayTree {
20  Node *root;
21  void Init() {
22    root = nil;
23  }
24  void Pushup(Node *x) {
25    if (x == nil)   return;
26    Pushdown(x);
27    Pushdown(x->c[0]);
28    Pushdown(x->c[1]);
29    x->size = x->c[0]->size + x->c[1]->size + 1;
30  }
31  void Pushdown(Node *x) {
32    if (x == nil)   return;
33    //do something
34  }
35  void Rotate(Node *x, int f) {
36    if (x == nil)   return;
37    Node *y = x->p;
38    y->c[f ^ 1] = x->c[f], x->p = y->p;
39    if (x->c[f] != nil)
40      x->c[f]->p = y;
41    if (y->p != nil)
42      y->p->c[y->p->c[1] == y] = x;
43    x->c[f] = y, y->p = x;
44    Pushup(y);
45  }
46  void Splay(Node *x, Node *f) {
47    static Node *stack[maxn];
48    int top = 0;
49    stack[top++] = x;
50    for (Node *y = x; y != f; y = y->p)
51      stack[top++] = y->p;
52    while (top)
53      Pushdown(stack[--top]);
54    while (x->p != f) {
55      Node *y = x->p;
56      if (y->p == f)
57        Rotate(x, x == y->c[0]);
58      else {
59        int fd = y->p->c[0] == y;
60        if (y->c[fd] == x)
61          Rotate(x, fd ^ 1), Rotate(x, fd);
62        else
63          Rotate(y, fd), Rotate(x, fd);
64      }
65    }
66    Pushup(x);
```

```
67      if (f == nil)
68        root = x;
69    }
70    void Select(int k, Node *f) {
71      Node *x = root;
72      Pushdown(x);
73      int tmp;
74      while ((tmp = x->c[0]->size) != k) {
75        if (k < tmp)  x = x->c[0];
76        else
77          x = x->c[1], k -= tmp + 1;
78        Pushdown(x);
79      }
80      Splay(x, f);
81    }
82    void Select(int l, int r) {
83      Select(l, nil), Select(r + 2, root);
84    }
85    Node *Make_tree(int a[], int l, int r, Node *p) {
86      if (l > r)  return nil;
87      int mid = l + r >> 1;
88      Node *x = newNode(a[mid], p);
89      x->c[0] = Make_tree(a, l, mid - 1, x);
90      x->c[1] = Make_tree(a, mid + 1, r, x);
91      Pushup(x);
92      return x;
93    }
94    void Insert(int pos, int a[], int n) {
95      Select(pos, nil), Select(pos + 1, root);
96      root->c[1]->c[0] = Make_tree(a, 0, n - 1, root->c[1]);
97      Splay(root->c[1]->c[0], nil);
98    }
99    void Insert(int v) {
100     Node *x = root, *y = nil;
101     //Need pushdown
102     while (x != nil) {
103       y = x;
104       y->size++;
105       x = x->c[v >= x->key];
106     }
107     y->c[v >= y->key] = x = newNode(v, y);
108     Splay(x, nil);
109   }
110   void Remove(int l, int r) {
111     Select(l, r);
112     //Recycle(root->c[1]->c[0]);
113     root->c[1]->c[0] = nil;
114     Splay(root->c[1], nil);
115   }
116 };
```

1.4 Dynamic tree

```
1  struct SplayTree {
2    void Pushup(Node *x) {
3      if (x == nil)   return;
4      Pushdown(x);
5      Pushdown(x->c[0]);
6      Pushdown(x->c[1]);
7      x->size = x->c[0]->size + x->c[1]->size + 1;
8    }
9    void Pushdown(Node *x) {
10     if (x == nil)   return;
```

```
11      if (x->rev) {
12        x->rev = 0;
13        x->c[0]->rev ^= 1;
14        x->c[1]->rev ^= 1;
15        swap(x->c[0], x->c[1]);
16      }
17    }
18    bool isRoot(Node *x) {
19      return (x == nil) || (x->p->c[0] != x && x->p->c[1] != x);
20    }
21    void Rotate(Node *x, int f) {
22      if (isRoot(x))   return;
23      Node *y = x->p;
24      y->c[f ^ 1] = x->c[f], x->p = y->p;
25      if (x->c[f] != nil)
26        x->c[f]->p = y;
27      if (y != nil) {
28        if (y == y->p->c[1])
29          y->p->c[1] = x;
30        else if (y == y->p->c[0])
31          y->p->c[0] = x;
32      }
33      x->c[f] = y, y->p = x;
34      Pushup(y);
35    }
36    void Splay(Node *x) {
37      static Node *stack[MaxN];
38      int top = 0;
39      stack[top++] = x;
40      for (Node *y = x; !isRoot(y); y = y->p)
41        stack[top++] = y->p;
42      while (top)
43        Pushdown(stack[--top]);
44      while (!isRoot(x)) {
45        Node *y = x->p;
46        if (isRoot(y))
47          Rotate(x, x == y->c[0]);
48        else {
49          int fd = y->p->c[0] == y;
50          if (y->c[fd] == x)
51            Rotate(x, fd ^ 1), Rotate(x, fd);
52          else
53            Rotate(y, fd), Rotate(x, fd);
54        }
55      }
56      Pushup(x);
57    }
58    Node *Access(Node *u) {
59      Node *v = nil;
60      while (u != nil) {
61        Splay(u);
62        v->p = u;
63        u->c[1] = v;
64        Pushup(u);
65        u = (v = u)->p;
66        if (u == nil)
67          return v;
68      }
69    }
70    Node *LCA(Node *u, Node *v) {
71      Access(u);
72      return Access(v);
73    }
74    Node *Link(Node *u, Node *v) {
```

```
75      Access(u);
76      Splay(u);
77      u->rev = true;
78      u->p = v;
79    }
80    void ChangeRoot(Node *u) {
81      Access(u)->rev ^= 1;
82    }
83    Node *GetRoute(Node *u, Node *v) {
84      ChangeRoot(u);
85      return Access(v);
86    }
87  };
```

1.5 Partition tree

```
1   int n,m;
2   struct elem {
3     int v,index;
4   } a[120000];
5   int d[30][120000];
6   int s[30][120000];
7   bool cmp(elem a,elem b) {
8     if (a.v == b.v)
9       return a.index <= b.index;
10    return a.v < b.v;
11  }
12  void build(int depth,int l,int r) {
13    if (l == r)
14      return;
15    int mid = (l+r)/2;
16    int tl,tr;
17    tl = tr = 0;
18    for (int i = l; i <= r; i++) {
19      if (cmp(a[d[depth][i]],a[mid])) {
20        d[depth+1][l+tl] = d[depth][i];
21        tl++;
22      } else {
23        d[depth+1][mid+1+tr] = d[depth][i];
24        tr++;
25      }
26      s[depth][i] = tl;
27    }
28    build(depth+1,l,mid);
29    build(depth+1,mid+1,r);
30  }
31  int find(int depth,int dl,int dr,int fl,int fr,int k) {
32    if (fl == fr)
33      return a[d[depth][fl]].v;
34    int ls,rs;
35    int mid = (dl+dr)/2;
36    ls = (fl == dl)? 0 : s[depth][fl-1];
37    rs = s[depth][fr];
38    return (rs-ls < k)?
39           find(depth+1,mid+1,dr,mid+fl-dl-ls+1,mid+fr-dl-rs+1,k-(rs-ls))
40           : find(depth+1,dl,mid,dl+ls,dl+rs-1,k);
41  }
42  int main() {
43    while (scanf("%d%d",&n,&m) != EOF) {
44      for (int i = 1; i <= n; i++) {
45        scanf("%d",&a[i].v);
46        a[i].index = i;
```

```
    }
    sort(a+1,a+n+1,cmp);
    for (int i = 1; i <= n; i++)
      d[0][a[i].index] = i;
    build(0,1,n);
    int l,r,k;
    for (int i = 1; i <= m; i++) {
      scanf("%d%d%d",&l,&r,&k);
      printf("%d\n",find(0,1,n,l,r,k));
    }
  }
  return 0;
}
```

1.6 Treap

```
struct Treap {
  Treap* ch[2];
  int key, fix, size;

  Treap(int x) : key(x) {
    size = 1;
    fix = rand();
    ch[0] = ch[1] = nullptr;
  }
  int comp(int x) const {
    if (x == key) return -1;
    return x < key ? 0 : 1;
  }
  void maintain() {
    size = 1;
    if (ch[0] != nullptr) size += ch[0]->size;
    if (ch[1] != nullptr) size += ch[1]->size;
  }
};
bool fnd(Treap* o, int x) {
  while (o != nullptr) {
    int d = o->comp(x);
    if (d == -1) return true;
    o = o->ch[d];
  }
  return false;
}
void rotate(Treap* &o, int d) {
  Treap* k = o->ch[d ^ 1];
  o->ch[d ^ 1] = k->ch[d];
  k->ch[d] = o;
  o->maintain();
  k->maintain();
  o = k;
}
void insert(Treap* &o, int x) {;
  if (o == nullptr) {
    o = new Treap(x);
  } else {
    int d = o->comp(x);
    // int d = (x < o->key ? 0 : 1);
    insert(o->ch[d], x);
    if (o->ch[d]->fix > o->fix) rotate(o, d ^ 1);
  }
  o->maintain();
}
```

```
void remove(Treap* &o, int x) {
  int d = o->comp(x);
  if (d == -1) {
    Treap* u = o;
    if (o->ch[0] != nullptr && o->ch[1] != nullptr) {
      int dd = (o->ch[0]->fix > o->ch[1]->fix ? 1 : 0);
      rotate(o, dd);
      remove(o->ch[dd], x);
    } else {
      if (o->ch[0] == nullptr) o = o->ch[1];
      else o = o->ch[0];
      delete u;
      u = nullptr;
    }
  } else {
    remove(o->ch[d], x);
  }
  if (o != nullptr) o->maintain();
}
void clear(Treap* &o) {
  if (o->ch[0] != nullptr) clear(o->ch[0]);
  if (o->ch[1] != nullptr) clear(o->ch[1]);
  delete o;
  o = nullptr;
}
int Kth(Treap* o, int k) {
  if (o == nullptr || k <= 0 || k > o->size) return -1;
  int s = o->ch[0] == nullptr ? 0 : o->ch[0]->size;
  if (s + 1 == k) return o->key;
  else if (s >= k) return Kth(o->ch[0], k);
  else return Kth(o->ch[1], k - s - 1);
}
int Rnk(Treap* o, int x) {
  int r;
  if (o == nullptr) return 0;
  if (o->ch[0] == nullptr) r = 0;
  else r = o->ch[0]->size;
  if (x == o->key) return r + 1;
  if (x < o->key) return Rnk(o->ch[0], x);
  else return r + 1 + Rnk(o->ch[1], x);
}
```

2 Dynamic programming

2.1 RMQ

```
struct RMQ {
  void init(const vector<int> &a) {
    int n = a.size();
    for (int i = 0; i < n; ++i) dp[i][0] = a[i];
    for (int j = 1; (1 << j) <= n; ++j) {
      for (int i = 0; i + (1 << j) -1 < n; ++i) {
        dp[i][j] = min(dp[i][j - 1], dp[i + (1 << (j - 1))][j - 1]);
      }
    }
  }

  // [l, r]
  int query(int l, int r) {
    // // int k = 0;
    // while ((1 << k + 1) <= r - l + 1) ++k;
    int k = log(r - l + 1) / log(2.0);
```

```
17      return min(dp[l][k], dp[r - (1 << k) + 1][k]);
18    }
19  } rmq;
```

2.2 2D-LIS

```
1  #include<cstdio>
2  #include<map>
3  using namespace std;
4  map<int,int> mp[100001];
5  bool check(int idx,int x,int y) {
6    if (!idx) return 1;
7    if (mp[idx].begin()->first>=x) return 0;
8    map<int,int> ::iterator it=mp[idx].lower_bound(x);
9    it--;
10   if (it->second<y) return 1;
11   else return 0;
12 }
13 int main() {
14   int n;
15   scanf("%d",&n);
16   int l=0,r=0;
17   for (int i=0; i<n; i++) {
18     int x,y;
19     scanf("%d%d",&x,&y);
20     int tl=l,tr=r;
21     while (tl<tr) {
22       int mid=(tl+tr+1)/2;
23       if (check(mid,x,y))
24         tl=mid;
25       else
26         tr=mid-1;
27     }
28     if (tl==r) r++;
29     int idx=tl+1;
30     map<int,int> ::iterator itl=mp[idx].lower_bound(x),itr=itl;
31     while (itr!=mp[idx].end() && itr->second>y) itr++;
32     if (mp[idx].find(x)!=mp[idx].end())
33       y=min(y,mp[idx][x]);
34     if (itl!=itr) mp[idx].erase(itl,itr);
35     if (mp[idx].find(x)==mp[idx].end() || mp[idx][x]>y)
36       mp[idx][x]=y;
37   }
38   printf("%d\n",r);
39   return 0;
40 }
```

2.3 数位-DP

```
1  #include <bits/stdc++.h>
2  using namespace std;
3  typedef long long ll;
4  int a[20];
5  ll dp[20][state];  //不同题目状态不同
6  ll dfs(int pos, /*state变量*/, bool lead /*前导零*/,
7         bool limit /*数位上界变量*/)  //不是每个题都要判断前导零
8  {
9    //递归边界，既然是按位枚举，最低位是0，那么pos==-1说明这个数我枚举完了
10   if (pos == -1)
```

```
11      return 1; /*这里一般返回1，表示你枚举的这个数是合法的，那么这里就需要你在枚举时必
            须每一位都要满足题目条件，也就是说当前枚举到pos位，一定要保证前面已经枚举的数
            位是合法的。不过具体题目不同或者写法不同的话不一定要返回1
12                  */
13    //第二个就是记忆化(在此前可能不同题目还能有一些剪枝)
14    if (!limit && !lead && dp[pos][state] != -1) return dp[pos][state];
15    /*常规写法都是在没有限制的条件记忆化，这里与下面记录状态是对应，具体为什么是有条件
          的记忆化后面会讲*/
16    int up =
17        limit ? a[pos] : 9;  //根据limit判断枚举的上界up;这个的例子前面用213讲过了
18    ll ans = 0;
19    //开始计数
20    for (int i = 0; i <= up; i++)  //枚举，然后把不同情况的个数加到ans就可以了
21    {
22      if ()
23        ... else if ()... ans +=
24            dfs(pos - 1, /*状态转移*/, lead && i == 0,
25                limit && i == a[pos])  //最后两个变量传参都是这样写的
26      /*这里还算比较灵活，不过做几个题就觉得这里也是套路了
27      大概就是说，我当前数位枚举的数是i，然后根据题目的约束条件分类讨论
28      去计算不同情况下的个数，还有要根据state变量来保证i的合法性，比如题目
29      要求数位上不能有62连续出现,那么就是state就是要保存前一位pre,然后分类，
30      前一位如果是6那么这意味就不能是2，这里一定要保存枚举的这个数是合法*/
31    }
32    //计算完，记录状态
33    if (!limit && !lead) dp[pos][state] = ans;
34    /*这里对应上面的记忆化，在一定条件下时记录，保证一致性，当然如果约束条件不需要考虑
          lead，这里就是lead就完全不用考虑了*/
35    return ans;
36  }
37  ll solve(ll x) {
38    int pos = 0;
39    while (x)  //把数位都分解出来
40    {
41      a[pos++] =
42          x %
43          10;  //个人老是喜欢编号为[0,pos),看不惯的就按自己习惯来，反正注意数位边界就行
44      x /= 10;
45    }
46    return dfs(
47        pos - 1 /*从最高位开始枚举*/, /*一系列状态 */, true,
48        true);  //刚开始最高位都是有限制并且有前导零的，显然比最高位还要高的一位视为0嘛
49  }
50  int main() {
51    ll le, ri;
52    while (~scanf("%lld%lld", &le, &ri)) {
53      //初始化dp数组为-1,这里还有更加优美的优化,后面讲
54      printf("%lld\n", solve(ri) - solve(le - 1));
55    }
56  }
```

3 Geometry

3.1 2D

3.1.1 Point

```
1  //Use cross product instead of atan2
2  bool cmp(const Point& a,const Point& b) {
3    if (a.y*b.y <= 0) {
```

```
4       if (a.y > 0 || b.y > 0) return a.y < b.y;
5       if (a.y == 0 && b.y == 0) return a.x < b.x;
6     }
7     return a*b > 0;
8   }
```

3.1.2 Line

```
1   Point operator &(const Line& b) const {
2     Point res = s;
3     double t = ((s - b.s) * (b.s - b.e)) / ((s - e) * (b.s - b.e));
4     res.x += (e.x - s.x) * t;
5     res.y += (e.y - s.y) * t;
6     return res;
7   }
```

3.1.3 Functions

```
1   Point nearestPointToLine(Point P, Line L) {
2     Point result;
3     double a, b, t;
4     a = L.e.x-L.s.x;
5     b = L.e.y-L.s.y;
6     t = ( (P.x-L.s.x)*a+(P.y-L.s.y)*b )/(a*a+b*b);
7     if (t >= 0 && t <= 1) {
8       result.x = L.s.x+a*t;
9       result.y = L.s.y+b*t;
10    }
11    return result;
12  }
13  //Segment
14  bool inter(Line l1,Line l2) {
15    return
16      max(l1.s.x,l1.e.x) >= min(l2.s.x,l2.e.x) &&
17      max(l2.s.x,l2.e.x) >= min(l1.s.x,l1.e.x) &&
18      max(l1.s.y,l1.e.y) >= min(l2.s.y,l2.e.y) &&
19      max(l2.s.y,l2.e.y) >= min(l1.s.y,l1.e.y) &&
20      sgn((l2.s-l1.s)*(l1.e-l1.s))*sgn((l2.e-l1.s)*(l1.e-l1.s)) <= 0 &&
21      sgn((l1.s-l2.s)*(l2.e-l2.s))*sgn((l1.e-l2.s)*(l2.e-l2.s)) <= 0;
22  }
23  bool onSeg(Line a,Point b) {
24    return ((a.s-b)*(a.e-b) == 0 &&
25            (b.x-a.s.x)*(b.x-a.e.x) <= 0 &&
26            (b.y-a.s.y)*(b.y-a.e.y) <= 0);
27  }
28  int inPoly(Point p,Point poly[], int n) {
29    int i, count;
30    Line ray, side;
31    count = 0;
32    ray.s = p;
33    ray.e.y  = p.y;
34    ray.e.x  = -1;//-∞
35    for (i = 0; i < n; i++) {
36      side.s = poly[i];
37      side.e = poly[(i+1)%n];
38      if(OnSeg(p, side))
39        return 1;
40      if (side.s.y == side.e.y)
41        continue;
42      if (OnSeg(side.s, ray)) {
43        if (side.s.y > side.e.y) count++;
```

```
    } else if (OnSeg(side.e, ray)) {
      if (side.e.y > side.s.y) count++;
    } else if (inter(ray, side)) {
      count++;
    }
  }
  return ((count % 2 == 1) ? 0 : 2);
}
Point centerOfPolygon(Point poly[],int n) {
  Point p, p0, p1, p2, p3;
  double m, m0;
  p1 = poly[0];
  p2 = poly[1];
  p.x = p.y = m = 0;
  for (int i = 2; i < n; i++) {
    p3 = poly[i];
    p0.x = (p1.x + p2.x + p3.x) / 3.0;
    p0.y = (p1.y + p2.y + p3.y) / 3.0;
    m0 = p1.x*p2.y+p2.x*p3.y+p3.x*p1.y-p1.y*p2.x-p2.y*p3.x-p3.y*p1.x;
    if (cmp(m + m0,0.0) == 0)
      m0 += eps;
    p.x = (m * p.x + m0 * p0.x) / (m + m0);
    p.y = (m * p.y + m0 * p0.y) / (m + m0);
    m = m + m0;
    p2 = p3;
  }
  return p;
}
```

3.1.4 Half plane intersection

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

```
29    while (head < tail && (((Q[head] & Q[head + 1]) - Q[tail].s) * (Q[tail].e - Q[tail
          ].s)) > EPS)
30      head++;
31    if (tail <= head + 1)   return;
32    for (int i = head; i < tail; i++)
33      res[resn++] = Q[i] & Q[i + 1];
34    if (head < tail + 1)
35      res[resn++] = Q[head] & Q[tail];
36  }
```

3.1.5 Convex hull

```
1   bool GScmp(Point a, Point b) {
2     if (fabs(a.x - b.x) < eps)
3       return a.y < b.y - eps;
4     return a.x < b.x - eps;
5   }
6   void GS(Point p[],int n,Point res[],int &resn) {
7     resn = 0;
8     int top = 0;
9     sort(p,p+n,GScmp);
10    if (conPoint(p,n)) {
11      res[resn++] = p[0];
12      return;
13    }
14    if (conLine(p,n)) {
15      res[resn++] = p[0];
16      res[resn++] = p[n-1];
17      return;
18    }
19    for (int i = 0; i < n;)
20      if (resn < 2 ||
21          (res[resn-1]-res[resn-2])*(p[i]-res[resn-1]) > 0)
22        res[resn++] = p[i++];
23      else
24        --resn;
25    top = resn-1;
26    for (int i = n-2; i >= 0;)
27      if (resn < top+2 ||
28          (res[resn-1]-res[resn-2])*(p[i]-res[resn-1]) > 0)
29        res[resn++] = p[i--];
30      else
31        --resn;
32    resn--;
33  }
```

3.1.6 Intersections of line and polygon

```
1   //Intersecting segment between [la,lb]
2   int Gao(int la,int lb,Line line) {
3     if (la > lb)
4       lb += n;
5     int l = la,r = lb,mid;
6     while (l < r) {
7       mid = l+r+1>>1;
8       if (cmp((line.e-line.s)*(p[la]-line.s),0)*cmp((line.e-line.s)*(p[mid]-line.s),0)
            >= 0)
9         l = mid;
10      else
11        r = mid-1;
12    }
```

```
13   return l%n;
14 }
15 double theta[maxn];
16 void Gettheta() {
17   for (int i = 0; i < n; i++) {
18     Point v = p[(i+1)%n]-p[i];
19     theta[i] = atan2(v.y,v.x);
20   }
21   for (int i = 1; i < n; i++)
22     if (theta[i-1] > theta[i]+eps)
23       theta[i] += 2*pi;
24 }
25 void Calc(Line l) {
26   double tnow;
27   Point v = l.e-l.s;
28   tnow = atan2(v.y,v.x);
29   if (cmp(tnow,theta[0]) < 0) tnow += 2*pi;
30   int pl = lower_bound(theta,theta+n,tnow)-theta;
31   tnow = atan2(-v.y,-v.x);
32   if (cmp(tnow,theta[0]) < 0) tnow += 2*pi;
33   int pr = lower_bound(theta,theta+n,tnow)-theta;
34   //Farest points with l on polygon
35   pl = pl%n;
36   pr = pr%n;
37   if (cmp(v*(p[pl]-l.s),0)*cmp(v*(p[pr]-l.s),0) >= 0)
38     return 0.0;
39   int xa = Gao(pl,pr,l);
40   int xb = Gao(pr,pl,l);
41   if (xa > xb)  swap(xa,xb);
42   //Intersecting with line P_xa -> P_xa+1 and P_xb -> P_xb+1
43   if (cmp(v*(p[xa+1]-p[xa]),0) == 0)  return 0.0;
44   if (cmp(v*(p[xb+1]-p[xb]),0) == 0)  return 0.0;
45   Point pa,pb;
46   //Intersections
47   pa = Line(p[xa],p[xa+1])&l;
48   pb = Line(p[xb],p[xb+1])&l;
49 }
```

3.2 3D

3.2.1 Point

```
1  Point3D operator *(const Point3D& b)const {
2    return Point3D(y*b.z-z*b.y,z*b.x-x*b.z,x*b.y-y*b.x);
3  }
4  //Rotate around V, notice that |V| = 1
5  Point3D Trans(Point3D pa,Point3D V,double theta) {
6    double s = sin(theta);
7    double c = cos(theta);
8    double x,y,z;
9    x = V.x;
10   y = V.y;
11   z = V.z;
12   Point3D pp =
13     Point3D(
14       (x*x*(1-c)+c)*pa.x+(x*y*(1-c)-z*s)*pa.y+(x*z*(1-c)+y*s)*pa.z,
15       (y*x*(1-c)+z*s)*pa.x+(y*y*(1-c)+c)*pa.y+(y*z*(1-c)-x*s)*pa.z,
16       (x*z*(1-c)-y*s)*pa.x+(y*z*(1-c)+x*s)*pa.y+(z*z*(1-c)+c)*pa.z);
17   return pp;
18 }
```

3.2.2 Functions

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

3.2.3 Convex hull

Don't forget **Randomshuffle**!

```
struct pt {
  double x, y, z;
  pt() {}
  pt(double _x, double _y, double _z): x(_x), y(_y), z(_z) {}
  pt operator - (const pt p1) {}
  pt operator * (pt p) {}
  double operator ^ (pt p) {}
};
struct _3DCH {
  struct fac {
    int a, b, c;
    bool ok;
  };
  int n;
  pt P[MAXV];
  int cnt;
  fac F[MAXV*8];
  int to[MAXV][MAXV];
  double vlen(pt a) {
    return sqrt(a.x*a.x+a.y*a.y+a.z*a.z);
  }
  double area(pt a, pt b, pt c) {
    return vlen((b-a)*(c-a));
  }
  double volume(pt a, pt b, pt c, pt d) {
    return (b-a)*(c-a)^(d-a);
  }
  double ptof(pt &p, fac &f) {
```

```
29      pt m = P[f.b]-P[f.a], n = P[f.c]-P[f.a], t = p-P[f.a];
30      return (m * n) ^ t;
31    }
32    void deal(int p, int a, int b) {
33      int f = to[a][b];
34      fac add;
35      if (F[f].ok) {
36        if (ptof(P[p], F[f]) > eps)
37          dfs(p, f);
38        else {
39          add.a = b, add.b = a, add.c = p, add.ok = 1;
40          to[p][b] = to[a][p] = to[b][a] = cnt;
41          F[cnt++] = add;
42        }
43      }
44    }
45    void dfs(int p, int cur) {
46      F[cur].ok = 0;
47      deal(p, F[cur].b, F[cur].a);
48      deal(p, F[cur].c, F[cur].b);
49      deal(p, F[cur].a, F[cur].c);
50    }
51    bool same(int s, int t) {
52      pt &a = P[F[s].a], &b = P[F[s].b], &c = P[F[s].c];
53      return fabs(volume(a, b, c, P[F[t].a])) < eps && fabs(volume(a, b, c,
54             P[F[t].b])) < eps && fabs(volume(a, b, c, P[F[t].c])) < eps;
55    }
56    void construct() {
57      cnt = 0;
58      if (n < 4)
59        return;
60      bool sb = 1;
61      for (int i = 1; i < n; i++) {
62        if (vlen(P[0] - P[i]) > eps) {
63          swap(P[1], P[i]);
64          sb = 0;
65          break;
66        }
67      }
68      if (sb)return;
69      sb = 1;
70      for (int i = 2; i < n; i++) {
71        if (vlen((P[0] - P[1]) * (P[1] - P[i])) > eps) {
72          swap(P[2], P[i]);
73          sb = 0;
74          break;
75        }
76      }
77      if (sb)return;
78      sb = 1;
79      for (int i = 3; i < n; i++) {
80        if (fabs((P[0] - P[1]) * (P[1] - P[2]) ^ (P[0] - P[i])) > eps) {
81          swap(P[3], P[i]);
82          sb = 0;
83          break;
84        }
85      }
86      if (sb)return;
87      fac add;
88      for (int i = 0; i < 4; i++) {
89        add.a = (i+1)%4, add.b = (i+2)%4, add.c = (i+3)%4, add.ok = 1;
90        if (ptof(P[i], add) > 0)
91          swap(add.b, add.c);
92        to[add.a][add.b] = to[add.b][add.c] = to[add.c][add.a] = cnt;
```

```
93        F[cnt++] = add;
94      }
95      for (int i = 4; i < n; i++) {
96        for (int j = 0; j < cnt; j++) {
97          if (F[j].ok && ptof(P[i], F[j]) > eps) {
98            dfs(i, j);
99            break;
100         }
101       }
102     }
103     int tmp = cnt;
104     cnt = 0;
105     for (int i = 0; i < tmp; i++) {
106       if (F[i].ok) {
107         F[cnt++] = F[i];
108       }
109     }
110   }
111   double area() {
112     double ret = 0.0;
113     for (int i = 0; i < cnt; i++) {
114       ret += area(P[F[i].a], P[F[i].b], P[F[i].c]);
115     }
116     return ret / 2.0;
117   }
118   double volume() {
119     pt O(0, 0, 0);
120     double ret = 0.0;
121     for (int i = 0; i < cnt; i++) {
122       ret += volume(O, P[F[i].a], P[F[i].b], P[F[i].c]);
123     }
124     return fabs(ret / 6.0);
125   }
126   int facetCnt_tri() {
127     return cnt;
128   }
129   int facetCnt() {
130     int ans = 0;
131     for (int i = 0; i < cnt; i++) {
132       bool nb = 1;
133       for (int j = 0; j < i; j++) {
134         if (same(i, j)) {
135           nb = 0;
136           break;
137         }
138       }
139       ans += nb;
140     }
141     return ans;
142   }
143   pt Fc[MAXV*8];
144   double V[MAXV*8];
145   pt Center() {
146     pt O(0,0,0);
147     for (int i = 0; i < cnt; i++) {
148       Fc[i].x = (O.x+P[F[i].a].x+P[F[i].b].x+P[F[i].c].x)/4.0;
149       Fc[i].y = (O.y+P[F[i].a].y+P[F[i].b].y+P[F[i].c].y)/4.0;
150       Fc[i].z = (O.z+P[F[i].a].z+P[F[i].b].z+P[F[i].c].z)/4.0;
151       V[i] = volume(O,P[F[i].a],P[F[i].b],P[F[i].c]);
152     }
153     pt res = Fc[0],tmp;
154     double m = V[0];
155     for (int i = 1; i < cnt; i++) {
156       if (fabs(m+V[i]) < eps)
```

```
157         V[i] += eps;
158       tmp.x = (m*res.x+V[i]*Fc[i].x)/(m+V[i]);
159       tmp.y = (m*res.y+V[i]*Fc[i].y)/(m+V[i]);
160       tmp.z = (m*res.z+V[i]*Fc[i].z)/(m+V[i]);
161       m += V[i];
162       res = tmp;
163     }
164     return res;
165   }
166 };
```

3.3 Circle

3.3.1 Functions

```
1  //Common area of two circle
2  double area(int x1,int y1,int x2,int y2,double r1,double r2) {
3    double s=dis(x2-x1,y2-y1);
4    if(r1+r2<s) return 0;
5    else if(r2-r1>s) return PI*r1*r1;
6    else if(r1-r2>s) return PI*r2*r2;
7    double q1=acos((r1*r1+s*s-r2*r2)/(2*r1*s));
8    double q2=acos((r2*r2+s*s-r1*r1)/(2*r2*s));
9    return (r1*r1*q1+r2*r2*q2-r1*s*sin(q1));
10 }
```

3.3.2 Union

```
1  for (int i = 1; i <= n; i++)
2    ans[i] = 0.0;
3  for (int i = 0; i < n; i++) {
4    tote = 0;
5    e[tote++] = Event(-pi,1);
6    e[tote++] = Event(pi,-1);
7    for (int j = 0; j < n; j++)
8      if (j != i) {
9        lab = Point(c[j].c.x-c[i].c.x,c[j].c.y-c[i].c.y);
10       AB = lab.Length();
11       AC = c[i].r;
12       BC = c[j].r;
13       if (cmp(AB+AC,BC) <= 0) {
14         e[tote++] = Event(-pi,1);
15         e[tote++] = Event(pi,-1);
16         continue;
17       }
18       if (cmp(AB+BC,AC) <= 0) continue;
19       if (cmp(AB,AC+BC) > 0)  continue;
20       theta = atan2(lab.y,lab.x);
21       fai = acos((AC*AC+AB*AB-BC*BC)/(2.0*AC*AB));
22       a0 = theta-fai;
23       if (cmp(a0,-pi) < 0)  a0 += 2*pi;
24       a1 = theta+fai;
25       if (cmp(a1,pi) > 0)  a1 -= 2*pi;
26       if (cmp(a0,a1) > 0) {
27         e[tote++] = Event(a0,1);
28         e[tote++] = Event(pi,-1);
29         e[tote++] = Event(-pi,1);
30         e[tote++] = Event(a1,-1);
31       } else {
32         e[tote++] = Event(a0,1);
33         e[tote++] = Event(a1,-1);
34       }
```

```
35      }
36    sort(e,e+tote,Eventcmp);
37    cur = 0;
38    for (int j = 0; j < tote; j++) {
39      if (cur != 0 && cmp(e[j].tim,pre[cur]) != 0) {
40        ans[cur] += Area(e[j].tim-pre[cur],c[i].r);
41        ans[cur] += xmult(Point(c[i].c.x+c[i].r*cos(pre[cur]),c[i].c.y+c[i].r*sin(pre[
              cur])),
42                          Point(c[i].c.x+c[i].r*cos(e[j].tim),c[i].c.y+c[i].r*sin(e[j].
                              tim)))/2.0;
43      }
44      cur += e[j].typ;
45      pre[cur] = e[j].tim;
46    }
47  }
48  for (int i = 1; i < n; i++)
49    ans[i] -= ans[i+1];
```

3.3.3 Area of intersection part with polygon

```
1   bool InCircle(Point a,double r) {
2     return cmp(a.x*a.x+a.y*a.y,r*r) <= 0;
3     //ϵ should big enough
4   }
5   double CalcArea(Point a,Point b,double r) {
6     Point p[4];
7     int tot = 0;
8     p[tot++] = a;
9     Point tv = Point(a,b);
10    Line tmp = Line(Point(0,0),Point(tv.y,-tv.x));
11    Point near = LineToLine(Line(a,b),tmp);
12    if (cmp(near.x*near.x+near.y*near.y,r*r) <= 0) {
13      double A,B,C;
14      A = near.x*near.x+near.y*near.y;
15      C = r;
16      B = C*C-A;
17      double tvl = tv.x*tv.x+tv.y*tv.y;
18      double tmp = sqrt(B/tvl);
19      p[tot] = Point(near.x+tmp*tv.x,near.y+tmp*tv.y);
20      if (OnSeg(Line(a,b),p[tot]) == true)  tot++;
21      p[tot] = Point(near.x-tmp*tv.x,near.y-tmp*tv.y);
22      if (OnSeg(Line(a,b),p[tot]) == true)  tot++;
23    }
24    if (tot == 3) {
25      if (cmp(Point(p[0],p[1]).Length(),Point(p[0],p[2]).Length()) > 0)
26        swap(p[1],p[2]);
27    }
28    p[tot++] = b;
29    double res = 0.0,theta,a0,a1,sgn;
30    for (int i = 0; i < tot-1; i++) {
31      if (InCircle(p[i],r) == true && InCircle(p[i+1],r) == true) {
32        res += 0.5*xmult(p[i],p[i+1]);
33      } else {
34        a0 = atan2(p[i+1].y,p[i+1].x);
35        a1 = atan2(p[i].y,p[i].x);
36        if (a0 < a1)  a0 += 2*pi;
37        theta = a0-a1;
38        if (cmp(theta,pi) >= 0) theta = 2*pi-theta;
39        sgn = xmult(p[i],p[i+1])/2.0;
40        if (cmp(sgn,0) < 0) theta = -theta;
41        res += 0.5*r*r*theta;
42      }
43    }
```

```
44    return res;
45  }
46  area2 = 0.0;
47  for (int i = 0; i < resn; i++) //counterclockwise
48    area2 += CalcArea(p[i],p[(i+1)%resn],r);
```

3.4 Matrix

3.4.1 基本矩阵

按向量 $\overrightarrow{(x,y,z)}$ 平移：

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

按比例 (x,y,z) 缩放：

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

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

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

以上矩阵变换都把点当作列向量，旋转角度的正负由右手定则决定

4 Graph

4.1 Dinic

```
1   struct Edge {int from, to, cap, flow;} ;
2   struct Dinic {
3     int n, m, s, t;
4     vector<Edge> edges;
5     vector<int> G[maxn];
6     bool vis[maxn];
7     int d[maxn];
8     int cur[maxn];
9
10    void init(int n) {
11      for (int i = 0; i <= n; ++i) G[i].clear();
12      edges.clear();
13    }
14
15    void add_edge(int from, int to, int cap) {
16      edges.push_back((Edge){from, to, cap, 0});
17      edges.push_back((Edge){to, from, 0, 0});
18      m = edges.size();
19      G[from].push_back(m - 2);
20      G[to].push_back(m - 1);
```

```
  }

  bool bfs() {
    memset(vis, 0, sizeof(vis));
    queue<int> Q;
    Q.push(s);
    d[s] = 0;
    vis[s] = 1;
    while (!Q.empty()) {
      int x = Q.front(); Q.pop();
      for (int i = 0; i < (int)G[x].size(); ++i) {
        Edge& e = edges[G[x][i]];
        if (!vis[e.to] && e.cap > e.flow) {
          vis[e.to] = 1;
          d[e.to] = d[x] + 1;
          Q.push(e.to);
        }
      }
    }
    return vis[t];
  }

  int dfs(int x, int a) {
    if (x == t || a == 0) return a;
    int flow = 0, f;
    for (int& i = cur[x]; i < G[x].size(); ++i) {
      Edge& e = edges[G[x][i]];
      if (d[x] + 1 == d[e.to] && (f = dfs(e.to, min(a, e.cap - e.flow))) > 0) {
        e.flow += f;
        edges[G[x][i] ^ 1].flow -= f;
        flow += f;
        a -= f;
        if (a == 0) break;
      }
    }
    return flow;
  }

  int run(int s, int t) {
    this->s = s; this->t = t;
    int flow = 0;
    while (bfs()) {
      memset(cur, 0, sizeof(cur));
      flow += dfs(s, inf);
    }
    return flow;
  }
} max_flow;
```

4.2 ISAP

```
struct Edge {
  int from, to, cap, flow;
};
struct ISAP {
  int n, m, s, t;  //结点数，边数（包括反向弧），源点编号，汇点编号
  vector<Edge> edges;
  vector<int> G[maxn];
  bool vis[maxn];
  int d[maxn];
  int cur[maxn];
  int p[maxn];
  int num[maxn];
```

```
13    void init(int n) {
14      this->n = n;
15      for (int i = 0; i <= n; ++i) G[i].clear();
16      edges.clear();
17    }
18    void add_edge(int from, int to, int cap) {
19      edges.push_back((Edge){from, to, cap, 0});
20      edges.push_back((Edge){to, from, 0, 0});
21      m = edges.size();
22      G[from].push_back(m - 2);
23      G[to].push_back(m - 1);
24    }
25    bool RevBFS() {
26      memset(vis, 0, sizeof(vis));
27      queue<int> Q;
28      Q.push(t);
29      d[t] = 0;
30      vis[t] = 1;
31      while (!Q.empty()) {
32        int x = Q.front();
33        Q.pop();
34        for (int i = 0; i < G[x].size(); i++) {
35          Edge &e = edges[G[x][i] ^ 1];
36          if (!vis[e.from] && e.cap > e.flow) {
37            vis[e.from] = 1;
38            d[e.from] = d[x] + 1;
39            Q.push(e.from);
40          }
41        }
42      }
43      return vis[s];
44    }
45    int Augment() {
46      int x = t, a = inf;
47      while (x != s) {
48        Edge &e = edges[p[x]];
49        a = min(a, e.cap - e.flow);
50        x = edges[p[x]].from;
51      }
52      x = t;
53      while (x != s) {
54        edges[p[x]].flow += a;
55        edges[p[x] ^ 1].flow -= a;
56        x = edges[p[x]].from;
57      }
58      return a;
59    }
60    int run(int s, int t) {
61      this->s = s; this->t = t;
62      int flow = 0;
63      RevBFS();
64      memset(num, 0, sizeof(num));
65      for (int i = 0; i < n; i++) {
66        num[d[i]]++;
67      }
68      int x = s;
69      memset(cur, 0, sizeof(cur));
70      while (d[s] < n) {
71        if (x == t) {
72          flow += Augment();
73          x = s;
74        }
75        int ok = 0;
76        for (int i = cur[x]; i < G[x].size(); i++) {
```

```
77          Edge &e = edges[G[x][i]];
78          if (e.cap > e.flow && d[x] == d[e.to] + 1) {
79            ok = 1;
80            p[e.to] = G[x][i];
81            cur[x] = i;
82            x = e.to;
83            break;
84          }
85        }
86        if (!ok) {
87          int m = n - 1;
88          for (int i = 0; i < G[x].size(); i++) {
89            Edge &e = edges[G[x][i]];
90            if (e.cap > e.flow) m = min(m, d[e.to]);
91          }
92          if (--num[d[x]] == 0) break;
93          num[d[x] = m + 1]++;
94          cur[x] = 0;
95          if (x != s) x = edges[p[x]].from;
96        }
97      }
98      return flow;
99    }
100 } max_flow;
```

4.3 Minimal cost maximal flow

```
1  struct Edge {
2    int from, to, cap, flow, cost;
3  };
4  struct MCMF {
5    int n, m, s, t;
6    vector<Edge> edges;
7    vector<int> G[maxn];
8    int inq[maxn];
9    int d[maxn];
10   int p[maxn];
11   int a[maxn];
12
13   void init(int n) {
14     this->n = n;
15     for (int i = 0; i < n; ++i) G[i].clear();
16     edges.clear();
17   }
18
19   void add_edge(int from, int to, int cap, int cost) {
20     edges.push_back((Edge) {from, to, cap, 0, cost});
21     edges.push_back((Edge) {to, from, 0, 0, -cost});
22     m = edges.size();
23     G[from].push_back(m - 2);
24     G[to].push_back(m - 1);
25   }
26
27   bool SPFA(int s, int t, int& flow, int& cost) {
28     for (int i = 0; i < n; ++i) d[i] = inf;
29     memset(inq, 0, sizeof(inq));
30     d[s] = 0; inq[s] = 1; p[s] = 0; a[s] = inf;
31
32     queue<int> Q;
33     Q.push(s);
34     while (!Q.empty()) {
35       int u = Q.front(); Q.pop();
36       inq[u] = 0;
```

```
37        for (int i = 0; i < (int)G[u].size(); ++i) {
38          Edge& e = edges[G[u][i]];
39          if (e.cap > e.flow && d[e.to]  > d[u] + e.cost) {
40            d[e.to] = d[u] + e.cost;
41            p[e.to] = G[u][i];
42            a[e.to] = min(a[u], e.cap - e.flow);
43            if (!inq[e.to]) { Q.push(e.to); inq[e.to] = 1; }
44          }
45        }
46      }
47
48      if (d[t] == inf) return false;
49      // if (d[t] >= 0) return false;
50
51      // add flow
52      flow += a[t];
53      cost += d[t] * a[t];
54      int u = t;
55      while (u != s) {
56        edges[p[u]].flow += a[t];
57        edges[p[u] ^ 1].flow -= a[t];
58        u = edges[p[u]].from;
59      }
60
61      return true;
62    }
63
64    int run(int s, int t) {
65      int flow = 0, cost = 0;
66      while (SPFA(s, t, flow, cost));
67      return cost;
68    }
69
70  } min_cost;
```

4.4 Johnson Minimal cost flow

```
1  #include <cstdio>
2  #include <cstring>
3  #include <algorithm>
4  #include <queue>
5  #include <stack>
6  using namespace std;
7  const int MAXN = 2003;
8  const int MAXM = 2000 * 1999 / 2 + 2000 * 3;
9  int N, L;
10 int head[MAXN];
11 struct Edge {
12   int to, next, flow, cost;
13 } edge[MAXM * 2];
14 int h[MAXN], dis[MAXN], pre[MAXN];
15 struct Heap {
16   int value[MAXN + 1], id[MAXN + 1];
17   int pos[MAXN];
18   int size;
19   void init() {
20     size = 1;
21   }
22   void swap2(int p, int q) {
23     swap(value[p], value[q]);
24     swap(id[p], id[q]);
25     pos[id[p]] = p;
26     pos[id[q]] = q;
```

```
27    }
28    void push_up(int p) {
29      while (p > 1 && value[p / 2] > value[p]) {
30        swap2(p, p / 2);
31        p /= 2;
32      }
33    }
34    void push_down(int p) {
35      while (p * 2 < size) {
36        int best = p;
37        if (p * 2 < size && value[p] > value[p * 2])
38          best = p * 2;
39        if (p * 2 + 1 < size && value[best] > value[p * 2 + 1])
40          best = p * 2 + 1;
41        if (p == best)
42          break;
43        swap2(p, best);
44        p = best;
45      }
46    }
47    void push(int _value, int _id) {
48      value[size] = _value;
49      id[size] = _id;
50      pos[_id] = size;
51      push_up(size++);
52    }
53    int top() {
54      return id[1];
55    }
56    void pop() {
57      value[1] = value[size - 1];
58      id[1] = id[--size];
59      pos[id[1]] = 1;
60      push_down(1);
61    }
62    void update(int _value, int _id) {
63      int p = pos[_id];
64      value[p] = _value;
65      push_up(p);
66    }
67  } heap;
68  bool inque[MAXN];
69  void init(int n) {
70    N = n;
71    L = 0;
72    memset(head, -1, 4 * n);
73  }
74  void add_edge(int u, int v, int flow, int cost) {
75    edge[L].to = v;
76    edge[L].flow = flow;
77    edge[L].cost = cost;
78    edge[L].next = head[u];
79    head[u] = L++;
80    edge[L].to = u;
81    edge[L].flow = 0;
82    edge[L].cost = -cost;
83    edge[L].next = head[v];
84    head[v] = L++;
85  }
86  void spfa(int s) {
87    memset(dis, 63, 4 * N);
88    memset(inque, 0, N);
89    memset(pre, -1, 4 * N);
90    dis[s] = 0;
```

```
91    queue <int> que;
92    que.push(s);
93    while (!que.empty()) {
94      int u = que.front();
95      inque[u] = 0;
96      que.pop();
97      for (int i = head[u]; i != -1; i = edge[i].next)
98        if (edge[i].flow) {
99          int v = edge[i].to;
100         if (dis[v] > dis[u] + edge[i].cost) {
101           dis[v] = dis[u] + edge[i].cost;
102           pre[v] = i;
103           if (!inque[v]) {
104             inque[v] = 1;
105             que.push(v);
106           }
107         }
108       }
109   }
110 }
111 void dijkstra(int s) {
112   for (int i = 0; i < N; ++i)
113     h[i] += dis[i];
114   memset(dis, 63, 4 * N);
115   memset(pre, -1, 4 * N);
116   memset(inque, 0, N);
117   dis[s] = 0;
118   inque[s] = 1;
119   heap.init();
120   heap.push(0, s);
121   while (heap.size > 1) {
122     int u = heap.top();
123     heap.pop();
124     for (int i = head[u]; i != -1; i = edge[i].next)
125       if (edge[i].flow) {
126         int v = edge[i].to;
127         if (dis[v] > dis[u] + edge[i].cost + h[u] - h[v]) {
128           dis[v] = dis[u] + edge[i].cost + h[u] - h[v];
129           pre[v] = i;
130           if (!inque[v]) {
131             heap.push(dis[v], v);
132             inque[v] = 1;
133           } else
134             heap.update(dis[v], v);
135         }
136       }
137   }
138 }
139 int minimumCostFlow(int s, int t, int &cost) {
140   int flow = 0;
141   memset(h, 0, 4 * N);
142   for (spfa(s); pre[t] != -1; dijkstra(s)) {
143     int maxs = edge[pre[t]].flow;
144     for (int i = pre[t]; i != -1; i = pre[edge[i ^ 1].to])
145       maxs = min(maxs, edge[i].flow);
146     for (int i = pre[t]; i != -1; i = pre[edge[i ^ 1].to]) {
147       edge[i].flow -= maxs;
148       edge[i ^ 1].flow += maxs;
149       cost += edge[i].cost * maxs;
150     }
151     flow += maxs;
152   }
153   return flow;
154 }
```

```
155 int main() {
156   return 0;
157 }
```

4.5 Bi-connect

```
1  struct edges {
2    int to,next;
3    bool cut,visit;
4  } edge[MAXM<<1];
5  int head[MAXN],low[MAXN],dpt[MAXN],L;
6  bool visit[MAXN],cut[MAXN];
7  void init(int n) {
8    L=0;
9    memset(head,-1,4*n);
10   memset(visit,0,n);
11 }
12 void add_edge(int u,int v) {
13   edge[L].cut=edge[L].visit=0;
14   edge[L].to=v;
15   edge[L].next=head[u];
16   head[u]=L++;
17 }
18 int idx;
19 stack<int> st;
20 int bcc[MAXM];
21 void dfs(int u,int fu,int deg) {
22   cut[u]=0;
23   visit[u]=1;
24   low[u]=dpt[u]=deg;
25   int tot=0;
26   for (int i=head[u]; i!=-1; i=edge[i].next) {
27     int v=edge[i].to;
28     if (edge[i].visit)
29       continue;
30     st.push(i/2);
31     edge[i].visit=edge[i^1].visit=1;
32     if (visit[v]) {
33       low[u]=dpt[v]>low[u]?low[u]:dpt[v];
34       continue;
35     }
36     dfs(v,u,deg+1);
37     edge[i].cut=edge[i^1].cut=(low[v]>dpt[u] || edge[i].cut);
38     if (u!=fu) cut[u]=low[v]>=dpt[u]?1:cut[u];
39     if (low[v]>=dpt[u] || u==fu) {
40       while (st.top()!=i/2) {
41         int x=st.top()*2,y=st.top()*2+1;
42         bcc[st.top()]=idx;
43         st.pop();
44       }
45       bcc[i/2]=idx++;
46       st.pop();
47     }
48     low[u]=low[v]>low[u]?low[u]:low[v];
49     tot++;
50   }
51   if (u==fu && tot>1) cut[u]=1;
52 }
53 int main() {
54   int n,m;
55   while (scanf("%d%d",&n,&m)!=EOF) {
56     init(n);
```

```
57      for (int i=0; i<m; i++) {
58        int u,v;
59        scanf("%d%d",&u,&v);
60        add_edge(u,v);
61        add_edge(v,u);
62      }
63      idx=0;
64      for (int i=0; i<n; i++)
65        if (!visit[i])
66          dfs(i,i,0);
67    }
68    return 0;
69  }
```

4.6 Cut and bridge

```
1  vector<int> G[maxn];
2  int dfn[maxn], low[maxn], dfs_clock;
3  //割点答案
4  bool iscut[maxn];
5  //桥答案
6  vector<pair<int,int> > bridge;
7  void init()
8  {
9    dfs_clock = 1;
10   memset(dfn, 0,sizeof(dfn));
11   for (int i = 1; i <= n; i++)
12   {
13     G[i].clear();
14   }
15   memset(iscut,0,sizeof(iscut));
16   bridge.clear();
17 }
18 void addedge(int u, int v)
19 {
20   G[u].push_back(v);
21   G[v].push_back(u);
22 }
23 void dfs(int u, int fa)
24 {
25   low[u] = dfn[u] = dfs_clock++;
26   int cnt = 0;
27   for (int v: G[u])
28   {
29     if (v != fa)
30     {
31       if (!dfn[v])
32       {
33         dfs(v, u);
34         cnt++;
35         low[u] = min(low[u], low[v]);
36         //判断割点 u?=1用于判断树根
37         if (u == 1 && cnt > 1) iscut[u] = true;
38         if (u != 1 && low[v] >= dfn[u]) iscut[u] = true;
39         //判断桥
40         if (low[v] > dfn[u]) bridge.push_back({u, v});
41       }
42       else
43       {
44         low[u] = min(low[u], dfn[v]);
45       }
46     }
```

```
47    }
48    if (cnt <= 1 && u == 1) iscut[u] = false;
49  }
```

4.7 Stoer-Wagner

```
1   int map[maxn][maxn];
2   int n;
3   void contract(int x,int y) {
4     int i,j;
5     for (i=0; i<n; i++)
6       if (i!=x) map[x][i]+=map[y][i],map[i][x]+=map[i][y];
7     for (i=y+1; i<n; i++) for (j=0; j<n; j++) {
8         map[i-1][j]=map[i][j];
9         map[j][i-1]=map[j][i];
10      }
11    n--;
12  }
13  int w[maxn],c[maxn];
14  int sx,tx;
15  int mincut() {
16    int i,j,k,t;
17    memset(c,0,sizeof(c));
18    c[0]=1;
19    for (i=0; i<n; i++) w[i]=map[0][i];
20    for (i=1; i+1<n; i++) {
21      t=k=-1;
22      for (j=0; j<n; j++) if (c[j]==0&&w[j]>k)
23          k=w[t=j];
24      c[sx=t]=1;
25      for (j=0; j<n; j++) w[j]+=map[t][j];
26    }
27    for (i=0; i<n; i++) if (c[i]==0) return w[tx=i];
28  }
29  int main() {
30    int i,j,k,m;
31    while (scanf("%d%d",&n,&m)!=EOF) {
32      memset(map,0,sizeof(map));
33      while (m--) {
34        scanf("%d%d%d",&i,&j,&k);
35        map[i][j]+=k;
36        map[j][i]+=k;
37      }
38      int mint=999999999;
39      while (n>1) {
40        k=mincut();
41        if (k<mint) mint=k;
42        contract(sx,tx);
43      }
44      printf("%d\n",mint);
45    }
46    return 0;
47  }
```

4.8 Euler path

```
1   //Directed graph
2   void solve(int x) {
3     int i;
4     if (!match[x]) {
5       path[++l]=x;
```

```
6       return ;
7     }
8     for (i=1; i<=n; i++)
9       if (b[x][i]) {
10        b[x][i]--;
11        match[x]--;
12        solve(i);
13      }
14    path[++l]=x;
15  }
16  //Undirected graph
17  void solve(int x) {
18    int i;
19    if (!match[x]) {
20      path[++l]=x;
21      return ;
22    }
23    for (i=1; i<=n; i++)
24      if (b[x][i]) {
25        b[x][i]--;
26        b[i][x]--;
27        match[x]--;
28        match[i]--;
29        solve(i);
30      }
31    path[++l]=x;
32  }
```

4.9 Strongly connected component

```
1   int dfsnum[2000];
2   int low[2000];
3   int stack[2000];
4   int top;
5   int ans;
6   int an;
7   int be[2000];
8   int flag[2000];
9   void dfs(int x) {
10    dfsnum[x] = low[x] = ans++;
11    stack[++top] = x;
12    flag[x] = 1;
13    for (int i = head[x]; i != -1; i = edge[i].next) {
14      int y = edge[i].to;
15      if (dfsnum[y] == -1) {
16        dfs(y);
17        low[x] = min(low[x],low[y]);
18      } else if (flag[y] == 1)
19        low[x] = min(low[x],dfsnum[y]);
20    }
21    if (dfsnum[x] == low[x]) {
22      while (stack[top] != x) {
23        flag[stack[top]] = 0;
24        be[stack[top]] = an;
25        top--;
26      }
27      flag[x] = 0;
28      be[x] = an++;
29      top--;
30    }
31  }
32  void SC() {
33    memset(dfsnum,-1,sizeof(dfsnum));
```

```
34    memset(flag,0,sizeof(flag));
35    top = 0;
36    an = 0;
37    ans = 0;
38    for (int i = 0; i < n; i++)
39      if (dfsnum[i] == -1)
40        dfs(i);
41  }
```

4.10 Match

4.10.1 Bipartite graph

```
1   bool check(int u) {
2     for (int i=head[u]; i!=-1; i=edge[i].next) {
3       int v=edge[i].to;
4       if (!use[v]) {
5         use[v]=1;
6         if (pre[v]==-1 || check(pre[v])) {
7           pre[v]=u;
8           return 1;
9         }
10      }
11    }
12    return 0;
13  }
14  int match() {
15    int ret=0;
16    memset(pre,-1,sizeof(pre));
17    for (int u=1; u<=N; u++) {
18      memset(use,0,sizeof(use));
19      if (check(u))
20        ret++;
21    }
22    return ret;
23  }
```

4.10.2 Edmonds

```
1   int N;
2   bool Graph[MaxN+1][MaxN+1];
3   int Match[MaxN+1];
4   bool InQueue[MaxN+1],InPath[MaxN+1],InBlossom[MaxN+1];
5   int Head,Tail;
6   int Queue[MaxN+1];
7   int Start,Finish;
8   int NewBase;
9   int Father[MaxN+1],Base[MaxN+1];
10  int Count;
11  void CreateGraph() {}
12  void Push(int u) {
13    Queue[Tail] = u;
14    Tail++;
15    InQueue[u] = true;
16  }
17  int Pop() {
18    int res = Queue[Head];
19    Head++;
20    return res;
21  }
22  int FindCommonAncestor(int u,int v) {
23    memset(InPath,false,sizeof(InPath));
```

```
24    while (true) {
25      u = Base[u];
26      InPath[u] = true;
27      if (u == Start) break;
28      u = Father[Match[u]];
29    }
30    while (true) {
31      v = Base[v];
32      if (InPath[v]) break;
33      v = Father[Match[v]];
34    }
35    return v;
36  }
37  void ResetTrace(int u) {
38    int v;
39    while (Base[u] != NewBase) {
40      v = Match[u];
41      InBlossom[Base[u]] = InBlossom[Base[v]] = true;
42      u = Father[v];
43      if (Base[u] != NewBase) Father[u] = v;
44    }
45  }
46  void BlossomContract(int u,int v) {
47    NewBase = FindCommonAncestor(u,v);
48    memset(InBlossom,false,sizeof(InBlossom));
49    ResetTrace(u);
50    ResetTrace(v);
51    if (Base[u] != NewBase) Father[u] = v;
52    if (Base[v] != NewBase) Father[v] = u;
53    for (int tu = 1; tu <= N; tu++)
54      if (InBlossom[Base[tu]]) {
55        Base[tu] = NewBase;
56        if (!InQueue[tu]) Push(tu);
57      }
58  }
59  void FindAugmentingPath() {
60    memset(InQueue,false,sizeof(InQueue));
61    memset(Father,0,sizeof(Father));
62    for (int i = 1; i <= N; i++)
63      Base[i] = i;
64    Head = Tail = 1;
65    Push(Start);
66    Finish = 0;
67    while (Head < Tail) {
68      int u = Pop();
69      for (int v = 1; v <= N; v++)
70        if (Graph[u][v] && (Base[u] != Base[v]) && (Match[u] != v)) {
71          if ((v == Start) ||
72              ((Match[v] > 0) && (Father[Match[v]] > 0)))
73            BlossomContract(u,v);
74          else if (Father[v] == 0) {
75            Father[v] = u;
76            if (Match[v] > 0)
77              Push(Match[v]);
78            else {
79              Finish = v;
80              return;
81            }
82          }
83        }
84    }
85  }
86  void AugmentPath() {
87    int u,v,w;
```

```
  u = Finish;
  while (u > 0) {
    v = Father[u];
    w = Match[v];
    Match[v] = u;
    Match[u] = v;
    u = w;
  }
}
void Edmonds() {
  memset(Match,0,sizeof(Match));
  for (int u = 1; u <= N; u++)
    if (Match[u] == 0) {
      Start = u;
      FindAugmentingPath();
      if (Finish > 0) AugmentPath();
    }
}
void PrintMatch() {}
int main() {
  CreateGraph();
  Edmonds();
  PrintMatch();
}
```

4.10.3 KM

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

```
37          if (visx[i]) lx[i]-=lack;
38          if (visy[i]) ly[i]+=lack;
39        }
40      }
41    }
42    int cost=0;
43    for (int i=0; i<ycnt; i++)
44      cost+=map[matchy[i]][i];
45  }
```

4.11 Clique

```
1  bool am[100][100];
2  int ans;
3  int c[100];
4  int U[100][100];
5  int n;
6  bool dfs(int rest,int num) {
7    if (!rest) {
8      if (num>=ans)
9        return 1;
10     else
11       return 0;
12   }
13   int pre=-1;
14   for (int i=0; i<rest && rest-i+num>=ans; i++) {
15     int idx=U[num][i];
16     if (num+c[idx]<ans)
17       return 0;
18     int nrest=0;
19     for (int j=i+1; j<rest; j++)
20       if (am[idx][U[num][j]])
21         U[num+1][nrest++]=U[num][j];
22     if (dfs(nrest,num+1))
23       return 1;
24   }
25   return 0;
26 }
27 int main() {
28   while (scanf("%d",&n),n) {
29     for (int i=0; i<n; i++)
30       for (int j=0; j<n; j++)
31         scanf("%d",&am[i][j]);
32     ans=0;
33     for (int i=n-1; i>=0; i--) {
34       int rest=0;
35       for (int j=i+1; j<n; j++)
36         if (am[i][j])
37           U[0][rest++]=j;
38       ans+=dfs(rest,0);
39       c[i]=ans;
40     }
41     printf("%d\n",ans);
42   }
43   return 0;
44 }
```

最大团的压位做法 by Claris

```
1  typedef unsigned long long U;
2  typedef long long ll;
3  const int N=45;
```

```
4  //0为有边，1为无边
5  int n,K,x,i,j,ans;bool flag;U g[N];double res;
6  inline int ctz(U s){return s?__builtin_ctzll(s):64;}
7  void BornKerbosch(U cur,U allow,U forbid){
8    if(!allow&&!forbid){
9      ans=max(ans,__builtin_popcountll(cur));
10     return;
11   }
12   if(!allow)return;
13   int pivot=ctz(allow|forbid);
14   U z=allow&~g[pivot];
15   for(int u=ctz(z);u<n;u+=ctz(z>>(u+1))+1){
16     BornKerbosch(cur|(1ULL<<u),allow&g[u],forbid&g[u]);
17     allow^=1ULL<<u,forbid|=1ULL<<u;
18   }
19 }
20 int main(){
21   scanf("%d",&n);
22   for(i=0;i<n;i++)g[i]=(1ULL<<n)-1-(1ULL<<i);
23   for(i=0;i<n;i++)for(j=0;j<n;j++){
24     scanf("%d",&x);
25 //0为有边，1为无边
26     if(!x&&i!=j)g[i]^=1ULL<<j;
27   }
28   BornKerbosch(0,(1ULL<<n)-1,0);
29   //ans为最大团大小
30   printf("%d",ans);
31 }
```

4.12 Spanning tree

4.12.1 Count the number of spanning tree

```
1 Matrix laplacian;
2 laplacian.clear();
3 for (int i = 0; i < n; i++)
4   for (int j = 0; j < n; j++)
5     if (i != j && G[i][j]) {
6       laplacian.a[i][j] = -1;
7       laplacian.a[i][i]++;
8     }
9 printf("%d\n",laplacian.det(n-1));
```

4.12.2 Spanning tree on directed graph

```
1  struct Edge {
2    int u,v,cost;
3  };
4  Edge e[1001*1001];
5  int pre[1001],id[1001],visit[1001],in[1001];
6  int zhuliu(int root,int n,int m,Edge e[]) {
7    int res = 0,u,v;
8    while (true) {
9      for (int i = 0; i < n; i++)
10       in[i] = inf;
11     for (int i = 0; i < m; i++)
12       if (e[i].u != e[i].v && e[i].cost < in[e[i].v]) {
13         pre[e[i].v] = e[i].u;
14         in[e[i].v] = e[i].cost;
15       }
16     for (int i = 0; i < n; i++)
```

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

4.13 Kth shortest path

```
#include<cstdio>
#include<cstring>
#include<queue>
using namespace std;
int K;
class states {
public:
  int cost,id;
};
int dist[1000];
class cmp {
public:
  bool operator ()(const states &i,const states &j) {
    return i.cost>j.cost;
  }
};
class cmp2 {
public:
  bool operator ()(const states &i,const states &j) {
    return i.cost+dist[i.id]>j.cost+dist[j.id];
  }
};
struct edges {
```

```
24    int to,next,cost;
25  } edger[100000],edge[100000];
26  int headr[1000],head[1000],Lr,L;
27  void dijkstra(int s) {
28    states u;
29    u.id=s;
30    u.cost=0;
31    dist[s]=0;
32    priority_queue<states,vector<states>,cmp> q;
33    q.push(u);
34    while (!q.empty()) {
35      u=q.top();
36      q.pop();
37      if (u.cost!=dist[u.id]) continue;
38      for (int i=headr[u.id]; i!=-1; i=edger[i].next) {
39        states v=u;
40        v.id=edger[i].to;
41        if (dist[v.id]>dist[u.id]+edger[i].cost) {
42          v.cost=dist[v.id]=dist[u.id]+edger[i].cost;
43          q.push(v);
44        }
45      }
46    }
47  }
48  int num[1000];
49  void init(int n) {
50    Lr=L=0;
51    memset(head,-1,4*n);
52    memset(headr,-1,4*n);
53    memset(dist,63,4*n);
54    memset(num,0,4*n);
55  }
56  void add_edge(int u,int v,int x) {
57    edge[L].to=v;
58    edge[L].cost=x;
59    edge[L].next=head[u];
60    head[u]=L++;
61    edger[Lr].to=u;
62    edger[Lr].cost=x;
63    edger[Lr].next=headr[v];
64    headr[v]=Lr++;
65  }
66  int a_star(int s,int t) {
67    if (dist[s]==0x3f3f3f3f)
68      return -1;
69    priority_queue<states,vector<states>,cmp2> q;
70    states tmp;
71    tmp.id=s;
72    tmp.cost=0;
73    q.push(tmp);
74    while (!q.empty()) {
75      states u=q.top();
76      q.pop();
77      num[u.id]++;
78      if (num[t]==K)
79        return u.cost;
80      for (int i=head[u.id]; i!=-1; i=edge[i].next) {
81        int v=edge[i].to;
82        tmp.id=v;
83        tmp.cost=u.cost+edge[i].cost;
84        q.push(tmp);
85      }
86    }
87    return -1;
```

```
}
int main() {
  int n,m;
  scanf("%d%d",&n,&m);
  init(n);
  for (int i=0; i<m; i++) {
    int u,v,x;
    scanf("%d%d%d",&u,&v,&x);
    add_edge(u-1,v-1,x);
  }
  int s,t;
  scanf("%d%d%d",&s,&t,&K);
  if (s==t)
    K++;
  dijkstra(t-1);
  printf("%d\n",a_star(s-1,t-1));
}
```

4.14 LCA

```
vector<int> G[maxn];
int root;
int fa[mxlg][maxn];
int depth[maxn];

void dfs(int u, int p, int d) {
  fa[0][u] = p;
  depth[u] = d;
  for (int i = 0; i < G[u].size(); ++i) {
    if (G[u][i] != p) dfs(G[u][i], u, d + 1);
  }
}

void init(int V) {
  dfs(root, -1, 0); // root = 1 ususally...
  for (int k = 0; k + 1 < mxlg; ++k) {
    for (int v = 0; v < V; ++v) {
      if (fa[k][v] < 0) fa[k + 1][v] = -1;
      else fa[k + 1][v] = fa[k][fa[k][v]];
    }
  }
}

int lca(int u, int v) {
  if (depth[u] > depth[v]) swap(u, v);
  for (int k = 0; k < mxlg; k++) {
    if ((depth[v] - depth[u]) >> k & 1) {
      v = fa[k][v];
    }
  }
  if (v == u) return u;
  for (int k = mxlg - 1; k >= 0; k--) {
    if (fa[k][u] != fa[k][v]) {
      u = fa[k][u];
      v = fa[k][v];
    }
  }
  return fa[0][u];
}
```

4.15 Virtual-tree

```
void build(vector<int>& V) {
  sort(V.begin(), V.end(), comp);

  int top = 0, tot = 0;
  S[top++] = 1;
  for (int i = 0; i < k; ++i) {
    int v = V[i];
    int lca = get_lca(v, S[top - 1]);
    key_node[v] = 1;
    if (!vis[v]) used[tot++] = v;
    vis[v] = 1;
    if (!vis[lca]) {
      vis[lca] = 1;
      used[tot++] = lca;
    }
    if (lca != S[top - 1]) {
      int x, y;
      while (top >= 2 && depth[S[top -2]] >= depth[lca]) {
        x = S[top - 2];
        y = S[top - 1];
        add_virtual_edge(x, y);
        --top;
      }

      if (S[top - 1] != lca) {
        x = lca;
        y = S[--top];
        add_virtual_edge(x, y);
        S[top++] = lca;
      }
    }
    S[top++] = v;
  }
  while (top > 1) {
    add_virtual_edge(S[top - 1], S[top - 2]);
    --top;
  }

  // work...

  for (int i = 0; i < tot; ++i) {
    virtual_tree[used[i]].clear();
    key_node[used[i]] = 0;
    vis[used[i]] = 0;
  }
  key_nodes.clear();
}
```

4.16 Stable marriage problem

假定有 n 个男生和 个女生，理想的拍拖状态就是对于每对情侣 (a, b)，找不到另一对情侣 (c, d) 使得 c 更喜欢 b,b 也更喜欢 c，同理，对 a 来说也没有 (e, f) 使得 a 更喜欢 e 而 e 更喜欢 a, 当然最后会有一些人落单。这样子一个状态可以称为理想拍拖状态，它也有一个专业的名词叫稳定婚姻。

求解这个问题可以用一个专有的算法，延迟认可算法，其核心就是让每个男生按自己喜欢的顺序逐个向女生表白，例如 leokan 向一个女生求爱，这个过程中，若这个女生没有男朋友，那么这个女生就暂时成为 leokan 的女朋友，或这个女生喜欢她现有男朋友的程度没有喜欢 leokan 高，这个女生也暂时成为 leokan 的女朋友，而她原有的男朋友则再将就找下一个次喜欢的女生来当女朋友。

```
#include<string.h>
#include<stdio.h>
#define N 1050
```

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

5 Math

5.1 Hill climbing

Hill climbing is an useful function to get the maximum value if you don't know what to do! just make a function and follow the instruction below:

```
for (s = 1; s > 1e-6; f = 0)
{
  if (F(dx, dy) > F(dx + s, dy)) dx += s, f = 1;
  else if (F(dx, dy) > F(dx - s, dy)) dx -= s, f = 1;
  else if (F(dx, dy) > F(dx, dy + s)) dy += s, f = 1;
  else if (F(dx, dy) > F(dx, dy - s)) dy -= s, f = 1;
  if (!f) s *= 0.7;
}
```

5.2 Linear Seq

杜教的递推板子，目测大概需要暴力递推阵的大小的两倍。

```
const ll moder = 998244353;
typedef vector<int> VI;
ll p_m(ll base, ll index)
{
  ll ret = 1;
  while(index)
  {
    if (index & 1) ret = ret * base % moder;
    base = base * base % moder;
    index >>= 1;
  }
  return ret;
}
int n;
namespace linear_seq
{
  const int N = 10000 + 10;
  ll res[N], base[N], _c[N], _md[N];
  vector<int> Md;
  void mul(ll *a, ll *b, int k)
  {
    for (int i = 0; i < k+k; i++) _c[i] = 0;
    for (int i = 0; i < k; i++)
      if (a[i])
        for (int j = 0; j < k; j++)
          _c[i+j] = (_c[i+j] + a[i] * b[j]) % moder;
    for (int i = k + k - 1; i >= k; i--)
      if (_c[i])
        for (int j = 0; j < Md.size(); j++)
          _c[i-k+Md[j]] = (_c[i-k+Md[j]] - _c[i] * _md[Md[j]]) % moder;
    for (int i = 0; i < k; i++) a[i] = _c[i];
  }
  int solve(ll n, VI a, VI b)
  {
    ll ans = 0, pnt = 0;
    int k = a.size();
    for (int i = 0; i < k; i++) _md[k-1-i] = -a[i];
    _md[k] = 1;
    Md.clear();
    for (int i = 0; i < k; i++) if (_md[i] != 0) Md.push_back(i);
    for (int i = 0; i < k; i++) res[i] = base[i] = 0;
    res[0] = 1;
    while((1LL << pnt) <= n) pnt++;
    for (int p = pnt; p >= 0; p--)
    {
      mul(res,res,k);
      if ((n>>p) & 1)
      {
        for (int i = k - 1; i >= 0; i--) res[i+1] = res[i];
        res[0] = 0;
        for (int j = 0; j < Md.size(); j++) res[Md[j]] = (res[Md[j]] - res[k] * _md[
           Md[j]]) % moder;
      }
    }
    for (int i = 0; i < k; i++) ans = (ans + res[i] * b[i]) % moder;
    if (ans < 0) ans += moder;
    return ans;
  }
  VI BM(VI s)
  {
```

```
60      VI C(1,1), B(1,1);
61      int L = 0,m = 1, b = 1;
62      for (int n = 0; n < s.size(); n++)
63      {
64        ll d = 0;
65        for (int i = 0; i <= L; i++) d = (d + (ll)C[i]*s[n-i])%moder;
66        if (d == 0)++m;
67        else if (2 * L <= n)
68        {
69          VI T = C;
70          ll c = moder - d * p_m(b, moder - 2) % moder;
71          while(C.size() < B.size() + m) C.push_back(0);
72          for (int i = 0; i < B.size(); i++) C[i+m] = (C[i+m] + c * B[i]) % moder;
73          L = n + 1 - L; B = T; b = d; m = 1;
74        }
75        else
76        {
77          ll c = moder - d * p_m(b, moder - 2) % moder;
78          while(C.size() < B.size() + m) C.push_back(0);
79          for (int i = 0; i < B.size(); i++) C[i+m] = (C[i+m] + c * B[i]) % moder;
80          ++m;
81        }
82      }
83      return C;
84    }
85    int gao(VI a, ll n)
86    {
87      VI c = BM(a);
88      c.erase(c.begin());
89      for (int i = 0; i < c.size(); i++) c[i] = (moder - c[i]) % moder;
90      return solve(n,c,VI(a.begin(), a.begin() + c.size()));
91    }
92  }
93  int main() {
94          while(~scanf("%d",&n))
95          printf("%d\n",linear_seq::gao(VI{1, 4, 12, 33, 88, 232, 609},n));
96  }
```

5.3 FFT

5.3.1 Bit operation

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

```
1  // Transforms the interval [x, y) in a.
2  void transform(int x, int y) {
3    if ( x == y - 1) {
4      return;
5    }
6    int l2 = ( y - x ) / 2;
7    int z = x + l2;
8    transform(x, z);
9    transform(z, y);
10   for (int i=x; i<z; i++) {
11     int x1 = a[i];
12     int x2 = a[i+l2];
13     a[i] = (x1 - x2 + MOD) % MOD;
14     a[i+l2] = (x1 + x2) % MOD;
15   }
16 }
17 // Reverses the transform in
```

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

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

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

5.3.2 Standard

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

```
35      s = 0;
36      e = s + t;
37      while (s < len) {
38        vir a, b, wo(cos(PI / t), sin(PI / t)), wn(1, 0);
39        for (j = s; j < s + t; j++) {
40          a = x[j];
41          b = x[j + t] * wn;
42          x[j] = a + b;
43          x[j + t] = a - b;
44          wn = wn * wo;
45        }
46        s = e + t;
47        e = s + t;
48      }
49    }
50  }
51  void dit_fft(vir *x, int len, int loglen) {
52    int i, j, s, e, t = 1 << loglen;
53    for (i = 0; i < loglen; i++) {
54      t >>= 1;
55      s = 0;
56      e = s + t;
57      while (s < len) {
58        vir a, b, wn(1, 0), wo(cos(PI / t), -sin(PI / t));
59        for (j = s; j < s + t; j++) {
60          a = x[j] + x[j + t];
61          b = (x[j] - x[j + t]) * wn;
62          x[j] = a;
63          x[j + t] = b;
64          wn = wn * wo;
65        }
66        s = e + t;
67        e = s + t;
68      }
69    }
70    change(x, len, loglen);
71    for (i = 0; i < len; i++)
72      x[i].re /= len;
73  }
```

5.3.3 Usage

```
1   vir x1[MAXN], x2[MAXN];
2   void solve(long long *a, int lena, long long *b, int lenb, long long *ret, int& len)
      {
3     int len1 = lena << 1;
4     int len2 = lenb << 1;
5     len = 1;
6     int loglen = 0;
7     while (len < len1 || len < len2) {
8       len <<= 1;
9       loglen++;
10    }
11    for (int i = 0; i < lena; i++)
12      x1[i] = vir(a[i], 0);
13    for (int i = lena; i < len; i++)
14      x1[i] = vir(0, 0);
15    for (int i = 0; i < lenb; i++)
16      x2[i] = vir(b[i], 0);
17    for (int i = lenb; i < len; i++)
18      x2[i] = vir(0, 0);
19    fft(x1, len, loglen);
20    fft(x2, len, loglen);
```

```
  for (int i = 0; i < len; i++)
    x1[i] = x1[i] * x2[i];
  dit_fft(x1, len, loglen);
  for (int i = 0; i < len; i++)
    ret[i] = (long long)(x1[i].re + 0.5);
}
```

5.4 Euler function

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

5.5 Ex-GCD

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

5.6 Möbius

两个公式:

$$F(n)=\sum_{d|n} f(d) \Longrightarrow f(n)=\sum_{d|n} \mu(d)F(\frac{n}{d}) \tag{1}$$

$$F(n)=\sum_{n|d} f(d) \Longrightarrow f(n)=\sum_{n|d} \mu(\frac{d}{n})F(d) \tag{2}$$

$$\mu(n)=\begin{cases}1 & n=1\\ (-1)^k & n=p_1p_2...p_k\\ 0 & \text{其余情况}\end{cases}$$

5.6.1 Möbius 用于容斥

容斥原理: 在集合 S 中至少具有 $P_1P_2...P_m$ 中一个元素的个数是：

$$|S_1 \cup S_2 \cup S_3 ... \cup S_n| = \sum |S_i| - \sum |S_i \cup S_j| + ... + \sum (-1)^{m+1} |S_1 \cup S_2 ... \cup S_m|$$

常用转化式:

$$\sum_{i=1}^{n} \lfloor \frac{n}{i} \rfloor = \sum_{i=1}^{n} d(i), d(n)是n的正因子数目\ (埃筛)$$

$$[x = 1] = \sum_{d|x} \mu(d)$$

```
const int maxn = 1000000 + 100;
int primes[maxn], ptot;
int mu[maxn];
bool nprime[maxn];
void init()
{
  nprime[1] = true;
  mu[1] = 1;
  for (int i = 2; i < maxn; i++)
  {
    if (!nprime[i])
    {
      primes[ptot++] = i;
      mu[i] = -1;
    }
    for (int j = 0; j < ptot && i * primes[j] < maxn; j++)
    {
      nprime[i * primes[j]] = true;
      if (i % primes[j] == 0)
      {
        mu[i * primes[j]] = -mu[i];
        break;
      }
    }
  }
}
```

5.7 Prime

5.7.1 Get primes

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

5.7.2 Get factors

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

```
63    x = rand()%(n-1)+1;
64    y = x;
65    while(true) {
66      i++;
67      x = (muti_mod(x,x,n) + c) % n;
68      d = gcd(y-x, n);
69      if (1 < d && d < n) return d;
70      if( y == x) return n;
71      if(i == k) {
72        y = x;
73        k <<= 1;
74      }
75    }
76  }
77  void findFactor(bint n,int k) {
78    if(n==1)return;
79    if(miller_rabin(n, TIME)) {
80      factor[++fac_top] = n;
81      return;
82    }
83    bint p = n;
84    while(p >= n)
85      p = pollard_rho(p,k--);
86    findFactor(p,k);
87    findFactor(n/p,k);
88  }
```

5.7.3 区间筛

```
1   const int maxn = 1000000 + 10;
2   int primes[maxn];
3   int ptot;
4   bool nprime[maxn];
5   void intervalprime(int L, int U)
6   {
7     int i, j;
8     int SU = sqrt((double) U);
9     int d = U - L + 1;
10    for (int i = 0; i < d; i++) nprime[i] = 0;
11    //去偶数，可删(改下面起始点为2，步长为1)
12    for (int i = (L % 2 != 0); i < d; i+= 2) nprime[i] = 1;
13
14    for (int i = 3; i <= SU; i += 2)
15    {
16      if (i > L && nprime[i - L]) continue;
17      j = (L / i) * i;
18      if (j < L) j += i;
19      j = j - L;
20      for (; j < d; j += i) nprime[j] = 1;
21    }
22    if (L <= 1) nprime[1 - L] = 1;
23    if (L <= 2) nprime[2 - L] = 0;
24    ptot = 0;
25    for (int i = 0; i < d; i++) if (!nprime[i]) primes[ptot++] = i + L;
26  }
```

5.8 Simpson

```
1   double Simp(double l,double r) {
2     double h = (r-l)/2.0;
3     return h*(calc(l)+4*calc((l+r)/2.0)+calc(r))/3.0;
```

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

5.9 Chinese remainder theorem

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

5.10 Lucas

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

5.11 Primitive root

```
int getPriRoot(int p) {
  if (p==2) return 1;
  int phi = p - 1;
  getFactor(phi);
  for (int g = 2; g < p; ++g) {
```

```
6       bool flag=1;
7       for (int i = 0; flag && i < N; ++i)
8         if (power(g, phi/fac[i], p) == 1)
9           flag=0;
10      if (flag)
11        return g;
12    }
13  }
```

5.12 Inverse element

```
1  void getInv2(int x) {
2    inv[1]=1;
3    for (int i=2; i<=x; i++)
4      inv[i]=(mod-(mod/i)*inv[mod%i]%mod)%mod;
5  }
```

5.13 Calculator

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

```
1  #include <iostream>
2  #include <cstdio>
3  #include <cstring>
4  #include <algorithm>
5  #include <string>
6  using namespace std;
7
8  char s[100000];
9  int n,cur;
10 const string OP = "+-*";
11
12 char next_char() {
13   if (cur >= n) return EOF;
14   return s[cur];
15 }
16
17 int get_priority(char ch) {
18   if (ch =='*')  return 2;
19   return 1;
20 }
21
22 int P();
23
24 int calc(int a,char op,int b) {
25   if (op =='+')
26     return a+b;
27   if (op =='-')
28     return a-b;
29   if (op =='*')
30     return a*b;
31 }
32
33 int calc_exp(int p) {
34   int a = P();
35   while ((OP.find(next_char()) != OP.npos) &&
36          (get_priority(next_char()) >= p)) {
37     char op = next_char();
```

```
38      cur++;
39      a = calc(a,op,calc_exp(get_priority(op)+1));
40    }
41    return a;
42  }
43
44  int totvar,m,var[26],varid[26];
45
46  int P() {
47    if (next_char() =='-') {
48      cur++;
49      return -P();
50    } else if (next_char() =='+') {
51      cur++;
52      return P();
53    } else if (next_char() =='(') {
54      cur++;
55      int res = calc_exp(0);
56      cur++;
57      return res;
58    } else {
59      cur++;
60      return var[varid[s[cur-1]-'a']];
61    }
62  }
63
64  int id[26],minid;
65
66  int main() {
67    while (true) {
68      scanf("%d%d",&totvar,&var[0]);
69      if (totvar == 0 && var[0] == 0)  break;
70      for (int i = 1; i < totvar; i++)
71        scanf("%d",&var[i]);
72      scanf("%d",&m);
73      scanf("%s",s);
74      for (int i = 0; i < 26; i++)
75        id[i] = -1;
76      minid = 0;
77      n = strlen(s);
78      for (int i = 0; i < n; i++)
79        if (s[i] >='a' && s[i] <='z') {
80          if (id[s[i]-'a'] == -1) {
81            id[s[i]-'a'] = minid;
82            minid++;
83          }
84          s[i] ='a'+id[s[i]-'a'];
85        }
86      for (int i = 0; i < totvar; i++)
87        varid[i] = i;
88      int res = 0;
89      do {
90        cur = 0;
91        int tmp = calc_exp(0);
92        if (tmp == m) {
93          res++;
94          break;
95        }
96      } while (next_permutation(varid,varid+totvar));
97      //puts(s);
98      if (res > 0)
99        puts("YES");
```

```
100     else
101       puts("NO");
102   }
103   return 0;
104 }
```

5.14 Linear programming

```
1  #define MAXM 20  //max num of basic varibles
2  #define INF 1E200
3
4  double A[MAXM+5][MAXN+MAXM+5];
5  double b[MAXM+5],c[MAXN+MAXM+5];
6  int N[MAXN+5],B[MAXM+5];
7  double X[MAXN+MAXM+5],V;
8  int n,m,R,C,nCnt,bCnt;
9  int v1[MAXN],v2[MAXN];
10
11 int fcmp(double a,double b) {
12   if(fabs(a-b)<1E-7) return 0;
13   if(a>b) return 1;
14   return -1;
15 }
16
17 void Pivot(int l,int e) {
18   double t=A[l][e],p=c[e];
19   b[l]=b[l]/t;
20   for(int i=1; i<=C; i++)
21     A[l][i]/=t;
22   V=V-c[e]*b[l];
23   for(int i=1; i<=R; i++) {
24     if(i==l||fcmp(A[i][e],0.0)==0)
25       continue;
26     t=A[i][e];
27     b[i]=b[i]-t*b[l];
28     for(int j=1; j<=C; j++)
29       A[i][j]=A[i][j]-t*A[l][j];
30   }
31   for(int i=1; i<=C; i++)
32     c[i]=c[i]-p*A[l][i];
33   for(int i=1; i<=nCnt; i++) {
34     if(N[i]==e) {
35       N[i]=B[l];
36       break;
37     }
38   }
39   B[l]=e;
40 }
41
42 bool Process(double P[]) {
43   while(true) {
44     int e=-1;
45     double mV=-INF;
46     for(int i=1; i<=nCnt; i++)
47       if(fcmp(P[N[i]],mV)==1)
48         mV=P[N[i]],e=N[i];
49
50     if(fcmp(mV,0.0)<=0) break;
51     int l=-1;
52     mV=INF;
53     for(int i=1; i<=bCnt; i++) {
54       if(fcmp(A[i][e],0.0)==1) {
```

```
55          double t=b[i]/A[i][e];
56          if(fcmp(mV,t)==1||(fcmp(mV,t)==0&&(l==-1||B[l]>B[i])))
57            mV=t,l=i;
58        }
59      }
60      if(l==-1) return false;
61      Pivot(l,e);
62    }
63    return true;
64  }
65
66  bool initSimplex() {
67    nCnt=bCnt=0;
68    for(int i=1; i<=n; i++)
69      N[++nCnt]=i;
70    for(int i=1; i<=m; i++)
71      B[++bCnt]=i+n,A[i][n+i]=1.0;
72    R=bCnt,C=bCnt+nCnt;
73    double minV=INF;
74    int p=-1;
75    for(int i=1; i<=m; i++)
76      if(fcmp(minV,b[i])==1)
77        minV=b[i],p=i;
78    if(fcmp(minV,0.0)>=0)
79      return true;
80    N[++nCnt]=n+m+1;
81    R++,C++;
82    for(int i=0; i<=C; i++)
83      A[R][i]=0.0;
84    for(int i=1; i<=R; i++)
85      A[i][n+m+1]=-1.0;
86    Pivot(p,n+m+1);
87    if(!Process(A[R])) return false;
88    if(fcmp(b[R],0.0)!=0)
89      return false;
90    p=-1;
91    for(int i=1; i<=bCnt&&p==-1; i++)
92      if(B[i]==n+m+1) p=i;
93    if(p!=-1) {
94      for(int i=1; i<=nCnt; i++) {
95        if(fcmp(A[p][N[i]],0.0)!=0) {
96          Pivot(p,N[i]);
97          break;
98        }
99      }
100   }
101   bool f=false;
102   for(int i=1; i<=nCnt; i++) {
103     if(N[i]==n+m+1) f=true;
104     if(f&&i+1<=nCnt)
105       N[i]=N[i+1];
106   }
107   nCnt--;
108   R--,C--;
109   return true;
110 }
111
112 //-1: no solution 1: no bound 0: has a solution -V
113 int Simplex() {
114   if(!initSimplex())
115     return -1;
116   if(!Process(c))
117     return 1;
118   for(int i=1; i<=nCnt; i++)
```

```
119     X[N[i]]=0.0;
120   for(int i=1; i<=bCnt; i++)
121     X[B[i]]=b[i];
122   return 0;
123 }
124
125 int main() {
126   //n = 1;m=1;
127   //V= 0.0;
128   //c[1] = 1.0;
129   //A[1][1] = 1.0;
130   //b[1] = 5.0;
131   //Simplex();
132   //printf("V = %.3f\n",V);
133
134   while(scanf("%d",&v1[1]) == 1) {
135     for(int i = 2; i<=6; i++)
136       scanf("%d",&v1[i]);
137     n = 4;
138     m = 6;
139     for(int i = 0 ; i<=m+1; i++)
140       for(int j=0; j<=n+m+2; j++)
141         A[i][j] = c[j] = 0;
142     memset(b,0,sizeof(b));
143     V = 0.0;
144     /*
145     n 为未知数个数
146     m 为约束个数
147     目标：siama(c[i]*xi)
148     约束：sigma(A[i][j]*xj) <=b[i]; j = 1 ... n
149     解存在X里面
150     */
151     b[1] = v1[1] ;
152     A[1][1] = 1;
153     A[1][4] = 1;
154     b[2] = v1[2] ;
155     A[2][1] = 1;
156     A[2][3] = 1;
157     b[3] = v1[3] ;
158     A[3][3] = 1;
159     A[3][4] = 1;
160     b[4] = v1[4] ;
161     A[4][2] = 1;
162     A[4][3] = 1;
163     b[5] = v1[5] ;
164     A[5][2] = 1;
165     A[5][4] = 1;
166     b[6] = v1[6] ;
167     A[6][1] = 1;
168     A[6][2] = 1;
169     c[1] = 1;
170     c[2] = 1;
171     c[3] = 1;
172     c[4] = 1;
173     Simplex();
174     //printf("V = %.3f\n",V);
175     printf("%.3f␣%.3f␣%.3f␣%.3f\n",X[1],X[2],X[3],X[4]);
176
177   }
178   return 0;
179 }
```

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

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

5.16 Partition ways of an integer

$O(n\sqrt{n})$

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

5.17 Pell's equation

```
import java.math.BigInteger;
import java.util.*;
public class Main {
  public static class Fraction {
    public BigInteger num,den;
    public Fraction() {
      num=BigInteger.ZERO;
      den=BigInteger.ONE;
    }
    public Fraction(int _num,int _den) {
      num=BigInteger.valueOf(_num);
      den=BigInteger.valueOf(_den);
    }
    public Fraction(BigInteger _num,BigInteger _den) {
      num=_num;
      den=_den;
    }
    public Fraction gen() {
      BigInteger g=num.gcd(den);
```

```
20        return new Fraction(num.divide(g),den.divide(g));
21      }
22      public Fraction add(Fraction x) {
23        return new Fraction(x.num.multiply(den).add(num.multiply(x.den)),x.den.multiply
              (den)).gen();
24      }
25      public Fraction reciprocal() {
26        return new Fraction(den,num);
27      }
28      public void out() {
29        System.out.println(num+" /" +den);
30      }
31    }
32    public static BigInteger sqrt(BigInteger a) {
33      BigInteger b=a;
34      while (a.compareTo(b.multiply(b))<0)
35        b=b.multiply(b).add(a).divide(b.multiply(BigInteger.valueOf(2)));
36      return b;
37    }
38    public static boolean check(Fraction x,int n) {
39      return x.num.multiply(x.num).add(x.den.multiply(x.den.multiply(BigInteger.valueOf
            (n))).negate()).compareTo(BigInteger.ONE)==0;
40    }
41    static int p[]=new int[1000];
42    static int l;
43    public static void main(String[] args) {
44      BigInteger ans=BigInteger.ZERO;
45      int idx=0;
46      for (int n=2,r=2; n<=1000; n++) {
47        if (n==r*r) {
48          r++;
49          continue;
50        }
51        int tmp=calc(n,0,1),a=tmp,b=n-tmp*tmp;
52        p[0]=tmp;
53        l=1;
54        while (true) {
55          tmp=calc(n,a,b);
56          p[l++]=tmp;
57          a=a-tmp*b;
58          Fraction x=getFrac();
59          if (check(x,n)) {
60            if (ans.compareTo(x.num)<0) {
61              ans=x.num;
62              idx=n;
63            }
64            break;
65          }
66          a=-a;
67          b=(n-a*a)/b;
68        }
69      }
70      System.out.println(idx);
71    }
72    private static Fraction getFrac() {
73      Fraction ret=new Fraction(p[l-1],1);
74      for (int i=l-2; i>=0; i--)
75        ret=new Fraction(p[i],1).add(ret.reciprocal());
76      return ret;
77    }
78    private static int calc(int n, int a, int b) {
79      for (long i=2;; i++)
80        if ((i*b-a)*(i*b-a)>n)
```

```
81          return (int)i-1;
82      }
83  }
```

5.18 Polya

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

对于有 n 个位置的手镯，有 n 种旋转置换和 n 种翻转置换

对于旋转置换：
$C(f_i)=gcd(n,i)$,i 表示一次转过 i 颗宝石，$i=0$ 时 $c=n$；

对于翻转置换：

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

如果 n 为奇数： $C(f)=\frac{n}{2}+1$

5.19 拉格朗日插值法

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

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

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

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

这个组合系数可以通过高斯消元暴出来，前提是要猜到它满足递推关系。

5.20 正多面体顶点着色

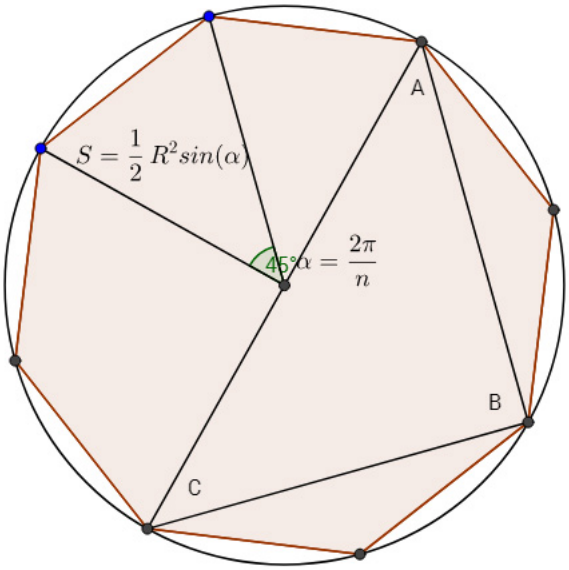
正四面体：$N=\frac{(n^4+11\times n^2)}{12}$
正六面体：$N=\frac{(n^8+17\times n^4+6\times n^2)}{24}$
正八面体：$N=\frac{(n^6+3\times n^4+12\times n^3+8\times n^2)}{24}$
正十二面体：$N=\frac{(n^{20}+15\times n^{10}+20\times n^8+24\times n^4)}{60}$
正二十面体：$N=\frac{(n^{12}+15\times n^6+44\times n^4)}{60}$

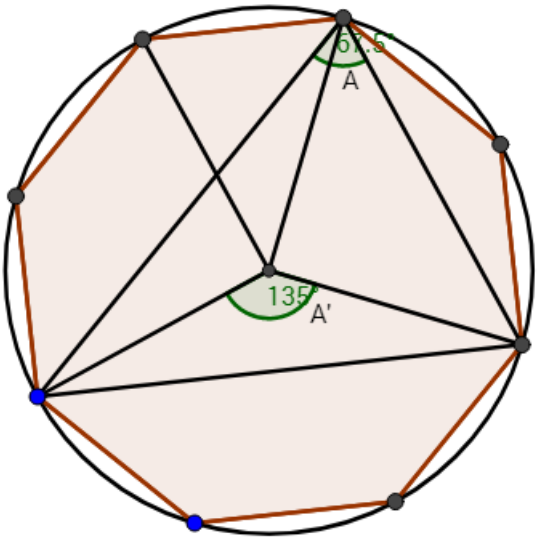
5.21 求和公式

$\sum k = \frac{n\times(n+1)}{2}$
$\sum 2k-1 = n^2$
$\sum k^2 = \frac{n\times(n+1)\times(2n+1)}{6}$
$\sum (2k-1)^2 = \frac{n\times(4n^2-1)}{3}$
$\sum k^3 = (\frac{n\times(n+1)}{2})^2$
$\sum (2k-1)^3 = n^2 \times (2n^2-1)$
$\sum k^4 = \frac{n\times(n+1)\times(2n+1)\times(3n^2+3n-1)}{30}$
$\sum k^5 = \frac{n^2\times(n+1)^2\times(2n^2+2n-1)}{12}$
$\sum k\times(k+1) = \frac{n\times(n+1)\times(n+2)}{3}$
$\sum k\times(k+1)\times(k+2) = \frac{n\times(n+1)\times(n+2)\times(n+3)}{4}$
$\sum k\times(k+1)\times(k+2)\times(k+3) = \frac{n\times(n+1)\times(n+2)\times(n+3)\times(n+4)}{5}$
$\sum i\times\binom{n}{i} = n\times 2^{n-1}$ (子集长度和)
$\sum_{i=1}^{n}\lfloor\frac{n}{i}\rfloor = \sum_{i=1}^{n} d(i), d(n)$是$n$的正因子数目 (埃筛)
$[x=1] = \sum_{d|x}\mu(d)$

5.22 几何公式

n-Polygon: $\frac{n}{2} * R^2 * sin(\frac{2\pi}{n})$
已知任意三点求边数最少的（同时也是面积最小）正多边形：由于 $A = \frac{A'}{2}$ 所以有 $A = k\frac{\alpha}{2}$ ，结论: $\alpha = 2gcd(A,B,C)$ 其中 A, B, C 为三角形内角





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

5.23 小公式

Pick 公式：$A = E\times 0.5 + I - 1$（A 是多边形面积，E 是边界上的整点，I 是多边形内部的整点）

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

求 $\binom{n}{k}$ 中素因子 P 的个数：

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

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

部分错排公式：
$n+m$ 个数中 m 个数必须错排求排列数

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

$dp[m]$ 为所求解

6 Search

6.1 Dancing links

```
struct DLX {
  int h,n,m,tot;
  int U[MaxN*MaxM],D[MaxN*MaxM],L[MaxN*MaxM],R[MaxN*MaxM],Row[MaxN*MaxM],Col[MaxN*
      MaxM];
  int S[MaxM],O[MaxN];
  bool hasans;
  void init() {
    h = 0;
    hasans = false;
    tot = m+n;
    for (int i = 0; i <= m; i++) {
      D[i] = U[i] = Col[i] = i;
      Row[i] = S[i] = 0;
      L[i] = (i+m)%(m+1);
      R[i] = (i+1)%(m+1);
    }
    for (int i = 1; i <= n; i++) {
      R[i+m] = L[i+m] = i+m;
      Row[i+m] = i;
      Col[i+m] = 0;
    }
  }
  void insert(int x,int y) {
    tot++;
    Row[tot] = x;
    Col[tot] = y;
    S[y]++;
    int colPos,rowPos;
    colPos = y;
    while (true) {
      colPos = D[colPos];
      if (colPos == y || Row[colPos] > x)    break;
    }
    colPos = U[colPos];
    if (Row[colPos] == x)   return;
    U[tot] = colPos;
    D[tot] = D[colPos];
    U[D[tot]] = D[U[tot]] = tot;
    rowPos = x+m;
```

```
39      while (true) {
40        rowPos = R[rowPos];
41        if (rowPos == x+m || Col[rowPos] > y)    break;
42      }
43      rowPos = L[rowPos];
44      if (Col[rowPos] == y)   return;
45      L[tot] = rowPos;
46      R[tot] = R[rowPos];
47      L[R[tot]] = R[L[tot]] = tot;
48    }
49    void print(int deep) {
50      for (int i = 0; i < deep; i++)
51        printf("%d ", O[i]);
52      printf("\n");
53    }
54    void cover(int col) {
55      L[R[col]] = L[col];
56      R[L[col]] = R[col];
57      for (int i = D[col]; i != col; i = D[i])
58        for (int j = R[i]; j != i; j = R[j])
59          if (Col[j] != col) {
60            U[D[j]] = U[j];
61            D[U[j]] = D[j];
62            S[Col[j]]--;
63          }
64    }
65    void resume(int col) {
66      for (int i = U[col]; i != col; i = U[i])
67        for (int j = L[i]; j != i; j = L[j])
68          if (Col[j] != col) {
69            S[Col[j]]++;
70            U[D[j]] = j;
71            D[U[j]] = j;
72          }
73      L[R[col]] = col;
74      R[L[col]] = col;
75    }
76    void initDFS() {
77      for (int i = 1; i <= n; i++) {
78        L[R[i+m]] = L[i+m];
79        R[L[i+m]] = R[i+m];
80      }
81    }
82    void DFS(int deep) {
83      if (hasans == true) return;
84      if (R[0] == 0)  {
85        hasans = true;
86        print(deep);
87        return;
88      };
89      int tc = R[0];
90      for (int i = R[0]; i != 0; i = R[i])
91        if (S[i] < S[tc])   tc = i;
92      cover(tc);
93      for (int i = D[tc]; i != tc; i = D[i]) {
94        int temp = O[deep];
95        O[deep] = Row[i];
96        for (int j = R[i]; j != i; j = R[j])
97          cover(Col[j]);
98        DFS(deep+1);
99        for (int j = L[i]; j != i; j = L[j])
100         resume(Col[j]);
101       O[deep] = temp;
```

```
    }
    resume(tc);
  }
}
```

6.1.1 Usage

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

6.2 Dancing links (A-star)

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

```
43      for (int i = 0; i < n; i++)
44        USE[i] = _USE[i];
45      return ;
46    }
47    int i,now = INF,c;
48    for (i = R[0]; i; i = R[i])
49      if (now > SUM[i])
50        now = SUM[c = i];
51    for(i = D[c]; i != c; i = D[i]) {
52      DLX_Remove_Repeat(i);
53      for(int j = R[i]; j != i; j = R[j])
54        DLX_Remove_Repeat(j);
55      _USE[n] = ROW[i];
56      DLX_Dfs(n + 1);
57      for(int j = L[i]; j != i; j = L[j])
58        DLX_Resume_Repeat(j);
59      DLX_Resume_Repeat(i);
60    }
61  }
62  void solve() {
63    //ANS = m
64    DLX_Dfs(0);
65  }
66  };
```

7 String

7.1 Aho-Corasick automation

```
1   #include <bits/stdc++.h>
2   using namespace std;
3   const int charsize = 26;
4   const int maxn = 500000;
5   struct Node { // you need to modify a lot ... before using it
6     int tot;
7     int root;
8     int next[maxn][charsize];
9     int fail[maxn];
10    int end[maxn]; // the cont of the word
11
12    inline int getid(const char& c) {}
13    int newnode() {
14      for (int i = 0; i < charsize; ++i) {
15        next[tot][i] = -1;
16      }
17      end[tot++] = 0;
18      return tot - 1;
19    }
20
21    void init() {
22      tot = 0;
23      root = newnode();
24    }
25
26    void insert(const char* str) {
27      int now = root;
28      while(*str) {
29        int charid = getid(*str);
30        if (next[now][charid] == -1) next[now][charid] = newnode();
31        now = next[now][charid];
32        ++str;
33      }
```

```
    ++end[now];
  }

  void build() {
    queue<int> q;
    fail[root] = root;
    for (int i = 0; i < charsize; ++i) {
      if (next[root][i] == -1) {
        next[root][i] = root;
      } else {
        fail[next[root][i]] = root;
        q.push(next[root][i]);
      }
    }

    while (!q.empty()) {
      int now = q.front(); q.pop();
      for (int i = 0; i < charsize; ++i) {
        if (next[now][i] == -1) {
          next[now][i] = next[fail[now]][i];
        } else {
          fail[next[now][i]] = next[fail[now]][i];
          q.push(next[now][i]);
        }
      }
    }
  }

  int solve(const char* str) {
    int ret = 0, k = 0;
    while (*str) {
      int charid = getid(*str);
      k = next[k][charid];
      int j = k;
      while (j) {
        ret += end[j];
        end[j] = 0;
        j = fail[j];
      }
      ++str;
    }
    return ret;
  }

} AC;
```

7.2 KMP

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

```
int fail[maxn];
void get_fail(const string& t) {
  fail[0] = -1;
  int n = t.size();
  int j = 0;
  int k = -1;
  while (j < n) {
    if (k == -1 || t[j] == t[k]) {
      fail[++j] = ++k;
    } else {
      k = fail[k];
    }
  }
```

```
}
int KMP(const string& s, const string& t) {
  get_fail(t);
  int sn = s.size();
  int tn = t.size();
  int i = 0;
  int j = 0;
  while (i < sn && j < tn) {
    if (j == -1 || s[i] == t[j]) {
      ++i; ++j;
    } else {
      j = fail[j];
    }
  }
  if (j == tn) return i - tn + 1;
  else return -1;
}
```

7.3 Extend-KMP

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

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

7.4 Manacher

```
const int maxn = 110000;

char Ma[maxn*2];
```

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

7.5 Suffix array

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

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

```
}
```

7.5.1 Longest common prefix

```
int lcp(int x,int y) {
  if (x > y)  swap(x,y);
  if (x == y)
    return len-sa[x];//NOTICE!
  x++;
  int k = lent[y-x+1];
  return min(f[x][k],f[y-(1<<k)+1][k]);
}
//Interval
void getinterval(int pos,int comlen,int& pl,int& pr) {
  int l,r,mid,cp;
  l = 0;
  r = pos;
  while (l < r) {
    mid = l+r>>1;
    cp = lcp(mid,pos);
    if (cp < comlen)
      l = mid+1;
    else
      r = mid;
  }
  pl = l;
  l = pos;
  r = len-1;
  while (l < r) {
    mid = l+r+1>>1;
    cp = lcp(pos,mid);
    if (cp < comlen)
      r = mid-1;
    else
      l = mid;
  }
  pr = l;
}
```

7.6 Smallest represention

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

7.7 Hash

```
typedef long long ll;
typedef unsigned long long ull;
const int MAXN = 1000000 + 100;
//ll gao[MAXN];
ll BL, BR, ML, MR;
ull psl[MAXN],psr[MAXN];
//call this before everything
void init(){
    int maxx = 1e9;
    srand(time(0));
    BL = (ll)maxx + rand() % maxx;
    BR = (ll)maxx + rand() % maxx;
    ML = (ll)maxx + rand() % maxx;
    MR = (ll)maxx + rand() % maxx;
}
//n is the max length you need
void Hash2(int n){
    for(int i = 0; i <= n ; i ++){
        psl[i] = (i == 0 ? 1 : psl[i - 1] * BL) % ML;
    }
    for(int i = 0 ; i <= n; i ++){
        psr[i] = (i == 0 ? 1 : psr[i - 1] * BR) % MR;
    }
}
struct _hash{
    //read your string in here.
    char str[MAXN];
    ull hs[MAXN];
    void build(){
        int n = strlen(str);
        ll l = 0, r = 0;
        for(int i = 0 ; i < n ; i ++){
            l = (l * BL + str[i]) % ML;
            r = (r * BR + str[i]) % MR;
            if(l < 0) l+= ML;
            if(r < 0)r += MR;
            hs[i + 1] = (l << 32) | r;
        }
    }
    //hash(str[b: e])
    ll get(int b,int e){
        ull el = (hs[e] >>32);
        ull er = (hs[e] & 0xffffffffULL);
        ull bl = (hs[b] >> 32);
        ull br = (hs[b] & 0xffffffffULL);
        ll l = el - bl * psl[e-b] % ML;
        ll r = er - br *psr[e-b] % MR;
        if(l < 0) l += ML;
        if(r < 0) r += MR;
        return (l << 32) | r;
    }
}Hash;
```

8 Tool

8.1 Fast IO

```
// 加强版
const int MAXBUF = 10000;
char buf[MAXBUF], *ps = buf, *pe = buf + 1;
inline void rnext() {
  if (++ps == pe)
```

```
6       pe = (ps = buf) +
7            fread(buf, sizeof(char), sizeof(buf) / sizeof(char), stdin);
8   }
9
10  template <class T>
11  inline bool in(T &ans) {
12    ans = 0;
13    T f = 1;
14    if (ps == pe) return false;  // EOF
15    do {
16      rnext();
17      if ('-' == *ps) f = -1;
18    } while (!isdigit(*ps) && ps != pe);
19    if (ps == pe) return false;  // EOF
20    do {
21      ans = (ans << 1) + (ans << 3) + *ps - 48;
22      rnext();
23    } while (isdigit(*ps) && ps != pe);
24    ans *= f;
25    return true;
26  }
27
28  const int MAXOUT = 10000;
29  char bufout[MAXOUT], outtmp[50], *pout = bufout, *pend = bufout + MAXOUT;
30  inline void write() {
31    fwrite(bufout, sizeof(char), pout - bufout, stdout);
32    pout = bufout;
33  }
34  inline void out_char(char c) {
35    *(pout++) = c;
36    if (pout == pend) write();
37  }
38  inline void out_str(const char *s) {
39    while (*s) {
40      *(pout++) = *(s++);
41      if (pout == pend) write();
42    }
43  }
44  template <class T>
45  inline void out_int(T x) {
46    if (!x) {
47      out_char('0');
48      return;
49    }
50    if (x < 0) x = -x, out_char('-');
51    int len = 0;
52    while (x) {
53      outtmp[len++] = x % 10 + 48;
54      x /= 10;
55    }
56    outtmp[len] = 0;
57    for (int i = 0, j = len - 1; i < j; i++, j--) swap(outtmp[i], outtmp[j]);
58    out_str(outtmp);
59  }
60  // how to use
61  int main() {
62    int t, ca = 1;
63    in(t);
64    while (t--) {
65      int n;
66      in(n);
67
68      out_str("Case #");
```

```
69      out_int(ca++);
70      out_str(" : ");
71      out_int(n), out_char('\n');
72    }
73    write();  // 一定要记得 write()啊！！！！！
74    return 0;
75  }
```

8.2 日期函数

```
1  int days[12]={31,28,31,30,31,30,31,31,30,31,30,31};
2  struct date{
3   int year,month,day;
4  };
5  //是否闰年
6  inline bool isleap(int year)
7  {
8    return  (year%4==0&&year%100!=0)||year%400==0;
9  }
10 //判合法性
11 inline bool islegal(date a){
12  if (a.month<0||a.month>12) return 0;
13  if (a.month==2)
14    return a.day>0&&a.day<=28+isleap(a.year);
15  return a.day>0&&a.day<=days[a.month-1];
16 }
17 //比较日期大小，正/负表示大于/小于，0表示相等
18 //如果用于sort等，请把-改成<
19 inline int datecmp(date a,date b){
20  if (a.year!=b.year)
21    return a.year-b.year;
22  if (a.month!=b.month)
23    return a.month-b.month;
24  return a.day-b.day;
25 }
26 //日期转天数偏移({0,1,1}第1天)
27 int date2int(date a){
28  int ret=a.year*365+(a.year-1)/4-(a.year-1)/100+(a.year-1)/400,i;
29  days[1]+=isleap(a.year);
30  for (i=0;i<a.month-1;ret+=days[i++]);
31  days[1]=28;
32  return ret+a.day;
33 }
34 //天数偏移转日期
35 date int2date(int a){
36  date ret;
37  ret.year=a/146097*400;
38  for (a%=146097;a>=365+isleap(ret.year);a-=365+isleap(ret.year),ret.year++);
39  days[1]+=isleap(ret.year);
40  for (ret.month=1;a>=days[ret.month-1];a-=days[ret.month-1],ret.month++);
41  days[1]=28;
42  ret.day=a+1;
43  return ret;
44 }
```

8.3 Bit compression

```
1  int bit[5];
2  inline int getbit26(int sta, int pos) {
3    return sta / bit[pos] % bit[1];
4  }
```

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

8.4 Random

```
// std::mt19937 rng_engine{randutils::auto_seed_128{}.base()};
std::mt19937 rng_engine{(size_t)(new char)};
std::uniform_int_distribution<int> dist{1, 1000};//1-1000 inclusive
int rand_integer = dist(rng_engine);
```

8.5 Hash map

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

```
42        return ele[tmp].val;
43    }
44    void set(int key1, int key2, long long val) {
45      int tmp = fint(key1, key2);
46      if (tmp == -1) {
47        insert(key1, key2);
48        ele[N - 1].val = val;
49      } else
50        ele[tmp].val = val;
51    }
52    void add(int key1, int key2, long long val) {
53      int tmp = fint(key1, key2);
54      if (tmp == -1) {
55        insert(key1, key2);
56        ele[N - 1].val += val;
57      } else
58        ele[tmp].val += val;
59    }
60  };
```

8.6 树链剖分

```
1  // with Segtree or splay...
2  void dfs(int u, int p, int d) {
3    sz[u] = 1;
4    fa[u] = p;
5    deep[u] = d;
6    for (int i = head[u]; i != -1; i = edge[i].next) {
7      int v = edge[i].v;
8      if (v == p) continue;
9      dfs(v, u, d + 1);
10     sz[u] += sz[v];
11     if (son[u] == -1 || sz[v] > sz[son[u]]) son[u] = v;
12   }
13 }
14
15 void link(int u, int first) {
16   top[u] = first;
17   tid[u] = ++tot;
18   arc[tid[u]] = u;
19   if (son[u] != -1) link(son[u], first);
20   for (int i = head[u]; i != -1; i = edge[i].next) {
21     int v = edge[i].v;
22     if (v == fa[u] || v == son[u]) continue;
23     link(v, v);
24   }
25 }
26 int solve(int u, int v) {
27   int ret =  0;
28   while (top[u] != top[v]) {
29     if (deep[top[u]] < deep[top[v]]) swap(u, v);
30     ret += st.query(id[top[u]], id[u] + 1);
31     u = fa[top[u]];
32   }
33   if (u == v) return ret; // 取决是否边权下放
34   if (deep[u] > deep[v]) swap(u, v);
35   ret += st.query(id[son[u]], id[v] + 1);
36   return ret;
37 }
```

8.7 单调栈

```
for (int i = 0; i < n; i++) l[i] = r[i] = i;
for (int i = 1; i < n; i++) {
  int now = i;
  while (now >= 1 && a[i] <= a[now - 1]) now = l[now - 1];
  l[i] = now;
}
for (int i = n - 2; i >= 0; i--) {
  int now = i;
  while (now < n - 1 && a[i] <= a[now + 1]) now = r[now + 1];
  r[i] = now;
}
```

8.8 单调队列

```
struct Deque {
  int val, idx;
  Deque(int v = 0, int x = 0) : val(v), idx(x) {}
} Q[maxn];
int head, tail;
vector<int> Max, Min;
int n, k;
void solve_min() {
  Min.clear();
  head = 1;
  tail = 0;
  for (int i = 1; i < k; ++i) {
    while (head <= tail && Q[tail].val >= a[i]) {
      --tail;
    }
    ++tail;
    Q[tail].val = a[i];
    Q[tail].idx = i;
  }
  for (int i = k; i <= n; ++i) {
    while (head <= tail && Q[tail].val >= a[i]) {
      --tail;
    }
    ++tail;
    Q[tail].val = a[i];
    Q[tail].idx = i;
    while (head <= tail && Q[head].idx < i - k + 1) {
      ++head;
    }
    Min.push_back(Q[head].val);
  }
}
void solve_max() {
  Max.clear();
  head = 1;
  tail = 0;
  for (int i = 1; i < k; ++i) {
    while (head <= tail && Q[tail].val <= a[i]) {
      --tail;
    }
    ++tail;
    Q[tail].val = a[i];
    Q[tail].idx = i;
  }
  for (int i = k; i <= n; ++i) {
    while (head <= tail && Q[tail].val <= a[i]) {
      --tail;
    }
    ++tail;
```

```
    Q[tail].val = a[i];
    Q[tail].idx = i;
    while (head <= tail && Q[head].idx < i - k + 1) {
      ++head;
    }
    Max.push_back(Q[head].val);
  }
}
```

8.9 Big-integer

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

const int MAXN = 410;

struct bign {
  int len, s[MAXN];
  bign() {
    memset(s, 0, sizeof(s));
    len = 1;
  }
  bign(int num) { *this = num; }
  bign(const char *num) { *this = num; }
  bign operator=(const int num) {
    char s[MAXN];
    sprintf(s, "%d", num);
    *this = s;
    return *this;
  }
  bign operator=(const char *num) {
    for (int i = 0; num[i] == '0'; num++)
      ;  //È¥Ç°µ¼0
    len = strlen(num);
    for (int i = 0; i < len; i++) s[i] = num[len - i - 1] - '0';
    return *this;
  }
  bign operator+(const bign &b) const  //+
  {
    bign c;
    c.len = 0;
    for (int i = 0, g = 0; g || i < max(len, b.len); i++) {
      int x = g;
      if (i < len) x += s[i];
      if (i < b.len) x += b.s[i];
      c.s[c.len++] = x % 10;
      g = x / 10;
    }
    return c;
  }
  bign operator+=(const bign &b) {
    *this = *this + b;
    return *this;
  }
  void clean() {
    while (len > 1 && !s[len - 1]) len--;
  }
```

```
52  bign operator*(const bign &b)  //*
53  {
54    bign c;
55    c.len = len + b.len;
56    for (int i = 0; i < len; i++) {
57      for (int j = 0; j < b.len; j++) {
58        c.s[i + j] += s[i] * b.s[j];
59      }
60    }
61    for (int i = 0; i < c.len; i++) {
62      c.s[i + 1] += c.s[i] / 10;
63      c.s[i] %= 10;
64    }
65    c.clean();
66    return c;
67  }
68  bign operator*=(const bign &b) {
69    *this = *this * b;
70    return *this;
71  }
72  bign operator-(const bign &b) {
73    bign c;
74    c.len = 0;
75    for (int i = 0, g = 0; i < len; i++) {
76      int x = s[i] - g;
77      if (i < b.len) x -= b.s[i];
78      if (x >= 0)
79        g = 0;
80      else {
81        g = 1;
82        x += 10;
83      }
84      c.s[c.len++] = x;
85    }
86    c.clean();
87    return c;
88  }
89  bign operator-=(const bign &b) {
90    *this = *this - b;
91    return *this;
92  }
93  bign operator/(const bign &b) {
94    bign c, f = 0;
95    for (int i = len - 1; i >= 0; i--) {
96      f = f * 10;
97      f.s[0] = s[i];
98      while (f >= b) {
99        f -= b;
100       c.s[i]++;
101     }
102   }
103   c.len = len;
104   c.clean();
105   return c;
106 }
107 bign operator/=(const bign &b) {
108   *this = *this / b;
109   return *this;
110 }
111 bign operator%(const bign &b) {
112   bign r = *this / b;
113   r = *this - r * b;
114   return r;
115 }
```

```
116   bign operator%=(const bign &b) {
117     *this = *this % b;
118     return *this;
119   }
120   bool operator<(const bign &b) {
121     if (len != b.len) return len < b.len;
122     for (int i = len - 1; i >= 0; i--) {
123       if (s[i] != b.s[i]) return s[i] < b.s[i];
124     }
125     return false;
126   }
127   bool operator>(const bign &b) {
128     if (len != b.len) return len > b.len;
129     for (int i = len - 1; i >= 0; i--) {
130       if (s[i] != b.s[i]) return s[i] > b.s[i];
131     }
132     return false;
133   }
134   bool operator==(const bign &b) { return !(*this > b) && !(*this < b); }
135   bool operator!=(const bign &b) { return !(*this == b); }
136   bool operator<=(const bign &b) { return *this < b || *this == b; }
137   bool operator>=(const bign &b) { return *this > b || *this == b; }
138   string str() const {
139     string res = "";
140     for (int i = 0; i < len; i++) res = char(s[i] + '0') + res;
141     return res;
142   }
143 };
144
145 istream &operator>>(istream &in, bign &x) {
146   string s;
147   in >> s;
148   x = s.c_str();
149   return in;
150 }
151
152 ostream &operator<<(ostream &out, const bign &x) {
153   out << x.str();
154   return out;
155 }
```

8.10 Code::blocks settings

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

8.11 Bit operation

8.11.1 基本操作

注意括号

8.11.2 枚举长为 n 含 k 个 1 的 01 串

```
1 int n = 5,k = 3;
2 for (int s = (1 << k)-1,u = 1 << n; s < u;) {
3   for (int i = 0;i < n;i++)
4     printf("%d",(((s>>(n-1-i))&1) == 1));
5   printf("\n");
6
7   int b = s & -s;
```

功能	示例	位运算
返回 lsb 之后的 0 的个数	$(1100010) \to 1D$	__builtin_ctz(x)[x==0 时 UB]
统计二进制 1 的个数	$(1100110 \to 4D)$	__builtin_popcount(x)
取最后一个 1 的 pos+1(ffs)	$(1000010 \to 2D)$	__builtin_ffs(x)
取最后一个 1 的 mask(lsb)	$(1000010 \to 10)$	(x & (-x))
去掉最后一位	$(101101 \to 10110)$	x shr 1
在最后加一个 0	$(101101 \to 1011010)$	x shl 1
在最后加一个 1	$(101101 \to 1011011)$	x shl 1+1
把最后一位变成 1	$(101100 \to 101101)$	x or 1
把最后一位变成 0	$(101101 \to 101100)$	x or 1-1
最后一位取反	$(101101 \to 101100)$	x xor 1
把右数第 k 位变成 1	$(101001 \to 101101, k = 3)$	x or (1 shl (k-1))
把右数第 k 位变成 0	$(101101 \to 101001, k = 3)$	x and not (1 shl (k-1))
右数第 k 位取反	$(101001 \to 101101, k = 3)$	x xor (1 shl (k-1))
取末三位	$(1101101 \to 101)$	x and 7
取末 k 位	$(1101101 \to 1101, k = 5)$	x and (1 shl k-1)
取右数第 k 位	$(1101101 \to 1, k = 4)$	x shr (k-1) and 1
把末 k 位变成 1	$(101001 \to 101111, k = 4)$	x or (1 shl k-1)
末 k 位取反	$(101001 \to 100110, k = 4)$	x xor (1 shl k-1)
把右边连续的 1 变成 0	$(100101111 \to 100100000)$	x and (x+1)
把右起第一个 0 变成 1	$(100101111 \to 100111111)$	x or (x+1)
把右边连续的 0 变成 1	$(11011000 \to 11011111)$	x or (x-1)
取右边连续的 1	$(100101111 \to 1111)$	(x xor (x+1)) shr 1
去掉右起第一个 1 的左边	$(100101000 \to 1000)$	x and (x xor (x-1))

```
8   s = (s+b)|(((s^(s+b))>>2)/b);
9 }
```

8.12 Profile of Vim

```
1  set nu
2  set history=1000
3
4  set tabstop
5  set shiftwidth=4
6  set smarttab
7
8  set cindent
9
10 colo evening
11
12 set nobackup
13 set noswapfile
14
15 set mouse=a
16
17 map <F5> : call CompileRun()<CR>
18 func! CompileRun()
19     exec "w"
20     exec "!g++ % -Wall -std=c++11 -o %<"
21     exec "!./%<"
22 endfunc
```

9 Appendix

9.1 Template by elfness

9.1.1 AC machine

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

```
58        ret+=q->ct;
59        q->ct=-1;
60        q=q->fail;
61      }
62    }
63    return ret;
64  }
65  int i,n,_;
66  int main() {
67    for(i=0; i<26; i++)VD.ne[i]=NULL;
68    VD.ct=0;
69    scanf("%d",&_);
70    while(_--) {
71      scanf("%d",&n);
72      init();
73      for(i=0; i<n; i++) {
74        scanf("%s",s);
75        build();
76      }
77      pre();
78      scanf("%s",s);
79      printf("%d\n",doit());
80    }
81  }
```

9.1.2 E-KMP

```
1  #include<cstdio>
2  #include<cstring>
3  #include<cstdlib>
4  #include<cmath>
5  #include<algorithm>
6  #include<iostream>
7  using namespace std;
8  typedef long long LL;
9  void e_kmp(char *s,char *t,int *has,int *e_has) {
10   int sp,p,mx,tn;
11   for(sp=p=mx=0; s[p]>0; p++) {
12     if(mx==p||p+e_has[p-sp]>=mx ) {
13       for(tn=mx-p; s[mx]==t[tn]; tn++)mx++;
14       has[sp=p]=mx-p;
15       if(mx==p)sp=++mx;
16     } else has[p]=e_has[p-sp];
17   }
18 }
19 const int V=1001000;
20 char t[V],s[V];
21 int e_has[V],has[V],tn;
22 int main() {
23   scanf("%s%s",s,t);
24   tn=strlen(t);
25   t[tn]=-1;
26   e_has[0] = tn;
27   e_kmp(t+1,t,e_has+1,e_has);
28   e_kmp(s,t,has,e_has);
29 }
```

9.1.3 KM (list)

```
1  #include<cstdio>
2  #include<cstring>
```

```
3  #include<cstdlib>
4  #include<cmath>
5  #include<algorithm>
6  using namespace std;
7  const int V=1200;
8  const int En=21000;
9  const int oo=1000000000;
10 struct Edge {
11   int num,ne,w;
12 } e[En];
13 int p[V],K;
14 void add(int x,int y,int z) {
15   e[K].num=y;
16   e[K].w=z;
17   e[K].ne=p[x];
18   p[x]=K++;
19 }
20 bool sx[V],sy[V];
21 int lx[V],ly[V],mat[V];
22 bool path(int u) {
23   sx[u]=true;
24   for(int i=p[u]; i!=-1; i=e[i].ne) {
25     int v=e[i].num;
26     if(!sy[v]&&lx[u]+ly[v]==e[i].w) {
27       sy[v]=true;
28       if(mat[v]==-1||path(mat[v])) {
29         mat[v]=u;
30         return true;
31       }
32     }
33   }
34   return false;
35 }
36 int N;
37 int KM() {
38   int i,j;
39   for(i=0; i<N; i++) {
40     lx[i]=-oo;
41     for(j=p[i]; j!=-1; j=e[j].ne)
42       lx[i]=max(lx[i],e[j].w);
43   }
44   for(i=0; i<N; i++)ly[i]=0,mat[i]=-1;
45   for(int u=0; u<N; u++)
46     while(1) {
47       for(i=0; i<N; i++)sx[i]=0,sy[i]=0;
48       if(path(u))break;
49       int dx=oo;
50       for(i=0; i<N; i++)if(sx[i])
51           for(j=p[i]; j!=-1; j=e[j].ne)
52             if(!sy[e[j].num])
53               dx=min(dx,lx[i]+ly[e[j].num]-e[j].w);
54       if(dx==oo)return -1;
55       for(i=0; i<N; i++)if(sx[i])lx[i]-=dx;
56       for(i=0; i<N; i++)if(sy[i])ly[i]+=dx;
57     }
58   int ret=0;
59   for(i=0; i<N; i++)ret+=lx[i]+ly[i];
60   return -ret;
61 }
62 int _,ca,n,m,i,x,y,z,te;
63 int main() {
64   scanf("%d",&_);
65   ca=0;
```

```
66    while(_--) {
67      ca++;
68      scanf("%d%d",&n,&m);
69      N=n;
70      for(i=0; i<n; i++)p[i]=-1;
71      K=0;
72      for(i=0; i<m; i++) {
73        scanf("%d%d%d",&x,&y,&z);
74        x--;
75        y--;
76        add(x,y,-z);
77        add(y,x,-z);
78      }
79      te=KM();
80      printf("Case %d: ",ca);
81      if(te==-1)puts("NO");
82      else printf("%d\n",te);
83    }
84  }
```

9.1.4 Nearest point pair

```
1   /*
2    * nearestPointPair.cpp
3    *
4    *  Created on: 2011-10-10
5    *      Author: Fish
6    */
7
8   #include <cstdio>
9   #include <cstring>
10  #include <cstdlib>
11  #include <cmath>
12  #include <algorithm>
13
14  using namespace std;
15
16  const int MaxN = 120000;
17  const int Log = 20;
18
19  struct Point {
20    double x, y;
21    Point() {
22    }
23    Point(double x, double y) :
24      x(x), y(y) {
25    }
26    Point operator-(const Point& p) const {
27      return Point(x - p.x, y - p.y);
28    }
29    double norm() const {
30      return hypot(x, y);
31    }
32    void init() {
33      scanf("%lf%lf", &x, &y);
34    }
35  } p[MaxN];
36  int x[MaxN], y[Log][MaxN], tmp[MaxN], n;
37  bool vst[MaxN];
38
39  bool comp_x(const int& i, const int& j) {
40    return p[i].x < p[j].x;
```

```
41 }
42
43 bool comp_y(const int& i, const int& j) {
44   return p[i].y < p[j].y;
45 }
46
47 double dfs(int k, int l, int r) {
48   double ret = 1e100;
49   if (r - l <= 2) {
50     for (int i = l; i < r; i++)
51       for (int j = i + 1; j <= r; j++)
52         ret = min(ret, (p[x[i]] - p[x[j]]).norm());
53     return ret;
54   }
55
56   int mid = (l + r) >> 1;
57   int lp = l, rp = mid + 1;
58   for (int i = l; i <= r; i++)
59     vst[x[i]] = i <= mid;
60   for (int i = l; i <= r; i++)
61     if (vst[y[k][i]])
62       y[k + 1][lp++] = y[k][i];
63     else
64       y[k + 1][rp++] = y[k][i];
65   double lhs = dfs(k + 1, l, mid);
66   double rhs = dfs(k + 1, mid + 1, r);
67   double mx = (p[x[mid + 1]].x + p[x[mid]].x) / 2.0;
68   ret = min(lhs, rhs);
69
70   lp = 0;
71   for (int i = l; i <= r; i++)
72     if (fabs(mx - p[y[k][i]].x) < ret)
73       tmp[lp++] = y[k][i];
74
75   for (int i = 0; i < lp; i++)
76     for (int j = 1; j < 8 && i + j < lp && (p[tmp[i + j]].y - p[tmp[i]].y) < ret; j
         ++)
77       ret = min(ret, (p[tmp[i]] - p[tmp[i + j]]).norm());
78
79   return ret;
80 }
81
82 int main() {
83 #ifdef __FISH__
84   freopen("data.in", "r", stdin);
85   freopen("nlogn.out", "w", stdout);
86 #endif
87   while (scanf("%d", &n) == 1 && n) {
88     for (int i = 0; i < n; i++) {
89       p[i].init();
90       x[i] = y[0][i] = i;
91     }
92     sort(x, x + n, comp_x);
93     sort(y[0], y[0] + n, comp_y);
94     printf("%.2f\n", dfs(0, 0, n - 1) / 2.0);
95     // printf("%.6f\n", dfs(0, 0, n - 1));
96   }
97
98   return 0;
99 }
```

9.1.5 SA

```
#include<cstdio>
#include<cstring>
#include<cstdlib>
#include<cmath>
#include<algorithm>
#include<iostream>
#include<vector>
#include<string>
using namespace std;
typedef long long LL;
const int N=100100;
char s[N];  /// 长度+1，对于非字符串，加一个小于最小值的元素，
int sa[N];  /// 倍增算法，结果 下标 1-n，第 i 大的是 sa[i]
int rk[N];  /// 第 i 位开始的后缀，的排名为 rk[i]
int wa[N],wb[N],wv[N],rmq[20][N];
int sn,to[N];
bool cmp(int *y,int a,int b,int L) {
  return y[a]==y[b]&&y[a+L]==y[b+L];
}
void da(char *s,int *sa,int len,int dn) {
  int i,j,p;
  int *x,*y,*t;
  x=wa;
  y=wb;
  for(i=0; i<dn; i++)rk[i]= 0;
  for(i=0; i<len; i++)rk[x[i]=s[i]]++;
  for(i=0; i<dn; i++)rk[i+1]+=rk[i];
  for(i=len-1; i>=0; i--)sa[--rk[x[i]]]=i;
  for(j=1,p=1; p<len; j*=2,dn=p) {
    for(p=0; p<j; p++)y[p]=len-j+p;
    for(i=0; i<len; i++)if(sa[i]>=j)y[p++]=sa[i]-j;
    for(i=0; i<len; i++)wv[i]=x[y[i]];
    for(i=0; i<dn; i++)rk[i]=0;
    for(i=0; i<len; i++)rk[wv[i]]++;
    for(i=0; i<dn; i++)rk[i+1]+=rk[i];
    for(i=len-1; i>=0; i--)sa[--rk[wv[i]]]=y[i];
    swap(x,y);
    x[sa[0]]=0;
    for(p=i=1; i<len; i++) {
      p+=!cmp(y,sa[i],sa[i-1],j);
      x[sa[i]]=p-1;
    }
  }
}
void find_height(char *s,int *sa,int len) {
  int *h=rmq[0];
  int i,j,k=0;
  for(i=1; i<=len; i++)
    rk[sa[i]] = i;
  for(i=0; i<len; i++) {
    if(k>0)k--;
    j=sa[rk[i]-1];
    while(s[i+k]==s[j+k])k++;
    h[rk[i]]=k;
  }
}
void RMQ(int n) {
  int i,j;
  int rn=(int)floor(log(n*2.0)/log(2.0));
  for(i=1; i<rn; i++)
    for(j=0; j<n+2-(1<<(i-1)); j++)
      rmq[i][j]=min(rmq[i-1][j],rmq[i-1][j+(1<<(i-1))]);
}
```

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

9.1.6 SAP

```
#include<cstdio>
#include<cstring>
#include<cstdlib>
#include<cmath>
#include<algorithm>
using namespace std;
const int V=220;
const int En=200000;
const int oo=0x3f3f3f3f;
struct Edge {
  int num,ne,c;
} e[En];
int d[V],p[V],pre[V],low[V];
```

```
int gap[V],cur[V];
int N,K,st,ed;
void add(int x,int y,int c) {
  e[K].num=y;
  e[K].c=c;
  e[K].ne=p[x];
  p[x]=K++;
  e[K].num=x;
  e[K].c=0;
  e[K].ne=p[y];
  p[y]=K++;
}
int sap() {
  int ret=0;
  bool fail;
  for(int i=0; i<=N; i++) {
    low[i]=gap[i]=d[i]=0;
    cur[i]=p[i];
  }
  low[st]=oo;
  gap[0]=N;
  int u=st;
  while(d[st]<N) {
    fail=true;
    for(int i=cur[u]; i!=-1; i=e[i].ne) {
      int v=e[i].num;
      cur[u]=i;
      if(e[i].c&&d[u]==d[v]+1) {
        pre[v]=i;
        low[v]=min(low[u],e[i].c);
        u=v;
        if(u==ed) {
          do {
            e[pre[u]].c-=low[ed];
            e[pre[u]^1].c+=low[ed];
            u=e[pre[u]^1].num;
          } while(u!=st);
          ret+=low[ed];
        }
        fail=false;
        break;
      }
    }
    if(fail) {
      gap[d[u]]--;
      if(!gap[d[u]])return ret;
      d[u]=N;
      for(int i=p[u]; i!=-1; i=e[i].ne)
        if(e[i].c)d[u]=min(d[u],d[e[i].num]+1);
      gap[d[u]]++;
      cur[u]=p[u];
      if(u!=st)u=e[pre[u]^1].num;
    }
  }
  return ret;
}
```

9.1.7 一般图最大匹配

```
#include <stdio.h>
#include <string.h>
#include <algorithm>
#include <vector>
```

```
5  #define maxn 300
6  #define maxm 90010
7
8  using namespace std;
9
10 int match[maxn];              //标记是否匹配
11 int st[maxn],aim[maxm],nxt[maxm],ln;  //边表
12 int q[maxn];                 //bfs队列
13 int level[maxn];              //离根深度的奇偶性
14 vector<int> ar[maxn];          //存每个点到根的路径
15 vector<int> a;                //找到的一条增广路
16 int n;
17 void init() {
18   for(int i=0; i<n; i++)st[i]=-1;
19   ln=0;
20 }
21 void in_edge(int x,int y) {
22   aim[ln]=y;
23   nxt[ln]=st[x];
24   st[x]=ln++;
25 }
26 int lca(int p,int q) {          //求p和q的最近公共祖先
27   int ret=0;
28   while (ret<ar[p].size() && ret<ar[q].size() && ar[p][ret]==ar[q][ret]) ret++;
29   return ret-1;
30 }
31 int FindAlterRoad(int sp) {
32   int qn=1;
33   memset(level,-1,sizeof(level));
34   level[q[0]=sp]=1;
35   ar[sp].clear();
36   ar[sp].push_back(sp);
37   for (int p=0; p<qn; p++) {
38     int x=q[p];
39     for (int i=st[x]; i!=-1; i=nxt[i]) {
40       int u=aim[i];
41       if (match[u]==u) continue;
42       if (level[u]==-1) {      //u是未访问的点
43         if (match[u]==-1) {    //u是未匹配的,找到增广路
44           a=ar[x];
45           a.push_back(u);
46           return 1;
47         } else {          //u是已匹配的点
48           int v=match[u];
49           if (level[v]!=-1) continue;
50           ar[v]=ar[x];
51           ar[v].push_back(u);
52           ar[v].push_back(v);
53           level[u]=0;
54           level[v]=1;
55           q[qn++]=v;
56         }
57       } else if (level[u]==1) {      //u和x同为偶点.形成花
58         int root=lca(u,x);
59         vector<int> tmp=ar[x];
60         for (int i=ar[u].size()-1; i>root; i--) {
61           int y=ar[u][i];
62           tmp.push_back(y);
63           if (level[y]==0) {
64             level[y]=1;
65             ar[y]=tmp;
66             level[y]=1;
67             q[qn++]=y;
```

```
68            }
69          }
70          tmp=ar[u];
71          for (int i=ar[x].size()-1; i>root; i--) {
72            int y=ar[x][i];
73            tmp.push_back(y);
74            if (level[y]==0) {
75              level[y]=1;
76              ar[y]=tmp;
77              level[y]=1;
78              q[qn++]=y;
79            }
80          }
81        }
82      }
83    }
84    return 0;
85  }
86  int MaximumMatch() {
87    int ret=0;                  //最大匹配数
88    memset(match,-1,sizeof(match));
89    for (int i=0; i<n; i++)
90      if (match[i]==-1)
91        if (FindAlterRoad(i)) {
92          for (int i=0; i<a.size(); i+=2) {
93            int u=a[i],v=a[i+1];
94            match[u]=v;
95            match[v]=u;
96          }
97          ret++;
98        } else match[i]=i;
99    return ret;
100 }
```

9.1.8 上下界最大流

```
1  /*
2  Author: elfness@UESTC
3  */
4  #include<cstdio>
5  #include<cstring>
6  #include<cstdlib>
7  #include<cmath>
8  #include<algorithm>
9  #include<iostream>
10 #include<vector>
11 #include<string>
12 using namespace std;
13 typedef long long LL;
14 const int V=1500;
15 const int En=900000;
16 const int inf=0x3f3f3f3f;
17 struct Edge {
18   int num,ne;
19   int c;
20 } e[En];
21 int p[V],K;
22 void add(int x,int y,int c) {
23   e[K].num=y;
24   e[K].c=c;
25   e[K].ne=p[x];
26   p[x]=K++;
27   e[K].num=x;
```

```
28    e[K].c=0;
29    e[K].ne=p[y];
30    p[y]=K++;
31  }
32  int d[V],pre[V],pree[V],gap[V],cur[V];
33  int N,st,ed;
34  int low[V];
35  int sap() {
36    int ret=0;
37    bool fail;
38    for(int i=0; i<=N; i++) {
39      d[i]=0;
40      gap[i]=0;
41      cur[i]=p[i];
42      low[i]=0;
43    }
44    low[st]=inf;
45    gap[0]=N;
46    int u=st;
47    while(d[st]<N) {
48      fail=true;
49      for(int i=cur[u]; i!=-1; i=e[i].ne) {
50        int v=e[i].num;
51        cur[u]=i;
52        if(e[i].c&&d[u]==d[v]+1) {
53          pre[v]=u;
54          pree[v]=i;
55          low[v]=min(low[u],e[i].c);
56          u=v;
57          if(u==ed) {
58            do {
59              e[pree[u]].c-=low[ed];
60              e[pree[u]^1].c+=low[ed];
61              u=pre[u];
62            } while(u!=st);
63            ret+=low[ed];
64          }
65          fail=false;
66          break;
67        }
68      }
69      if(fail) {
70        gap[d[u]]--;
71        if(!gap[d[u]])return ret;
72        d[u]=N;
73        for(int i=p[u]; i!=-1; i=e[i].ne)
74          if(e[i].c)d[u]=min(d[u],d[e[i].num]+1);
75        gap[d[u]]++;
76        cur[u]=p[u];
77        if(u!=st)u=pre[u];
78      }
79    }
80    return ret;
81  }
82  int n,m,s,t;
83  struct Elf {
84    int u,v,lo,up;
85  } b[12000];
86  int lb[12000];
87  int doit() {
88    int i;
89    N=n+2;
90    st=n;
91    ed=n+1;
```

```
92    for(i=0; i<N; i++)p[i]=-1;
93    K=0;
94    for(i=0; i<n; i++)lb[i]=0;
95    for(i=0; i<m; i++) {
96      lb[b[i].u]-=b[i].lo;
97      lb[b[i].v]+=b[i].lo;
98      add(b[i].u,b[i].v,b[i].up-b[i].lo);
99    }
100   for(i=0; i<n; i++) {
101     if(lb[i]>0)add(st,i,lb[i]);
102     else add(i,ed,-lb[i]);
103   }
104   add(t,s,inf);
105   int te=sap();
106   for(i=p[st]; i!=-1; i=e[i].ne)
107     if(e[i].c!=0)return -1;
108   st=s;
109   ed=t;
110   te=sap();
111   return te;
112 }
```

9.1.9 上下界最小流

```
1  #include<cstdio>
2  #include<cstdlib>
3  #include<cstring>
4  #include<cmath>
5  #include<algorithm>
6  using namespace std;
7  const int V=600;
8  const int En=50000;
9  const int oo=0x3f3f3f3f;
10 struct Edge {
11   int num,ne,c;
12 } e[En];
13 int p[V],K;
14 void add(int x,int y,int c) {
15   e[K].num=y;
16   e[K].c=c;
17   e[K].ne=p[x];
18   p[x]=K++;
19   e[K].num=x;
20   e[K].c=0;
21   e[K].ne=p[y];
22   p[y]=K++;
23 }
24 int d[V],cur[V],low[V],pre[V],gap[V],pree[V];
25 int st,ed,N;
26 int sap() {
27   int ret=0;
28   bool fail;
29   memset(gap,0,sizeof(gap));
30   memset(low,0,sizeof(low));
31   memset(d,0,sizeof(d));
32   for(int i=0; i<N; i++)cur[i]=p[i];
33   gap[0]=N;
34   low[st]=oo;
35   int u=st;
36   while(d[st]<N) {
37     fail=true;
38     for(int i=cur[u]; i!=-1; i=e[i].ne) {
39       int v=e[i].num;
```

```
40        cur[u]=i;
41        if(e[i].c&&d[u]==d[v]+1) {
42          pre[v]=u;
43          pree[v]=i;
44          low[v]=min(low[u],e[i].c);
45          u=v;
46          if(u==ed) {
47            do {
48              e[pree[u]].c-=low[ed];
49              e[pree[u]^1].c+=low[ed];
50              u=pre[u];
51            } while(u!=st);
52            ret+=low[ed];
53          }
54          fail=false;
55          break;
56        }
57      }
58      if(fail) {
59        gap[d[u]]--;
60        if(!gap[d[u]])return ret;
61        d[u]=N;
62        for(int i=p[u]; i!=-1; i=e[i].ne)
63          if(e[i].c)d[u]=min(d[u],d[e[i].num]+1);
64        gap[d[u]]++;
65        cur[u]=p[u];
66        if(u!=st)u=pre[u];
67      }
68    }
69    return ret;
70  }
71  struct ELF {
72    int u,v,lo;
73  } b[En];
74  int n,m,lb[V],ts,tt;
75  void solve() {
76    N=n+4;
77    ts=0;
78    tt=n+1;
79    st=n+2;
80    ed=n+3;
81    memset(lb,0,sizeof(lb));
82    int i,u,v;
83    for(i=0; i<N; i++)p[i]=-1;
84    K=0;
85    for(i=0; i<m; i++) {
86      u=b[i].u;
87      v=b[i].v;
88      lb[v]+=b[i].lo;
89      lb[u]-=b[i].lo;
90      add(u,v,oo-b[i].lo);
91    }
92    for(i=1; i<=n; i++) {
93      add(ts,i,oo);
94      add(i,tt,oo);
95    }
96    for(i=0; i<n+2; i++) {
97      if(lb[i]>0)add(st,i,lb[i]);
98      else add(i,ed,-lb[i]);
99    }
100   int ans=sap();
101   add(tt,ts,oo);
102   printf("%d\n",sap());
```

```
}
int _,ca,i;
int main() {
  scanf("%d",&_);
  ca=0;
  while(_--) {
    ca++;
    scanf("%d%d",&n,&m);
    for(i=0; i<m; i++) {
      scanf("%d%d%d",&b[i].u,&b[i].v,&b[i].lo);
    }
    printf("Case #%d: ",ca);
    solve();
  }
}
```

9.1.10 全局最小割

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

9.1.11 最小树型图

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

```
63      N=cnt;
64      Root=id[Root];
65    }
66    return true;
67  }
```