ACM Template

UESTC_Jungle

Last build at October 25, 2018

Contents

1 Datastructure

1.1 Fenwick

```
/* Fenwick Tree (Binary Indexed Tree), by Abreto <m@abreto.net>. */
#include <cstring>

using namespace std;

template <class T = int, int MAXN = 100001>
struct fenwick {
  static inline int lowbit(int x) {
    return (x&(-x));
  }
  int N;
  T f[MAXN]; /* 1=based. */
  fenwick(void):N(MAXN) {
    init();
  }
  fenwick(int n):N(n) {
    init();
  }
  void init(void) {
    memset(f,0,sizeof(f));
  }
  void upd(int i, T dx) {
    while(i <= N) {
      f[i] += dx;
      i += lowbit(i);
    }
  }
  T sum(int i) {
    T ret = 0;
    while(i) {
      ret += f[i];
      i -= lowbit(i);
    }
    return ret;
  }
};
```

```
/* Fenwick Tree (Binary Indexed Tree), by Abreto <m@abreto.net>. */

#define MAXN 100001
#define LOWBIT(x)   ((x)&(-(x)))

int N;
int fen[MAXN];

void update(int i, int dx) {
  while(i <= N) {
    fen[i] += dx;
    i += LOWBIT(i);
  }
}

int sum(int i) {
  int s = 0;
  while(i > 0) {
    s += fen[i];
    i -= LOWBIT(i);
  }
  return s;
}
```

1.2 BST in pb_ds

```
/* Red-Black tree via pb_ds. */
#include<bits/stdc++.h>
#include<ext/pb_ds/assoc_container.hpp>
#include<ext/pb_ds/tree_policy.hpp>
using namespace __gnu_pbds;
using namespace std;
template <typename T>
using ordered_set = tree<T, null_type, less<T>, rb_tree_tag, tree_order_statistics_node_update>;

int main() {
  ordered_set<int>  s;
  s.insert(1);
  s.insert(3);
  cout << s.order_of_key(2) << endl; // the number of elements in the s less than 2
  cout << *s.find_by_order(0) << endl; // print the 0-th smallest number in s(0-based)
}
```

1.3 Segment Tree

```
/* Segment tree (Interval tree, range tree), by Abreto <m@abreto.net>. */

template <int STMAX = 1000000>
struct segment_tree {
  struct node_t {
    static inline node_t merge(node_t n1, node_t n2) {
      node_t ans;
      ans.l = n1.l;
      ans.r = n2.r;
      /* merge n1 and n2 to ans. */
      return ans;
    }

    /* Data field */
    int l,r;
  } nodes[(STMAX+1)<<2];

  struct lazy_t {
    int marked; /* Optional */
    /* lazy mark. */

    lazy_t(void) {
      clear();
    }
    void clear(void) {
      marked=0;
    }
  } marks[(STMAX+1)<<2];

  inline void maintain_leaf(int o, int idx) {
    nodes[o].l = nodes[o].r = idx;
    /* Operations to single elements ... */
  }
  inline void maintain(int o) {
    nodes[o] = node_t::merge(nodes[o<<1], nodes[o<<1|1]);
  }

  /* Usage: build(1,1,n); */
  void build(int o, int l, int r) { /* [l,r] */
    if( r <= l ) {
      maintain_leaf(o, l);
    } else {
      int mid = l+r>>1;
      build(o<<1, l, mid);
      build(o<<1|1, mid+1, r);
      maintain(o);
    }
  }

  /* Modify all elements in [l,r] */
  void mark(lazy_t act, int o) {
    /* do something .. */
    marks[o].marked = 1;
  }

  /* Pass cached updates. */
  void pushdown(int o) {
    if( marks[o].marked ) {
      mark(marks[o], o<<1);
      mark(marks[o], o<<1|1);
      marks[o].clear();
    }
  }

  /* Do act on all elements in [L,R] */
  void upd(int L, int R, lazy_t act, int o, int l, int r) {
    if( L <= l && r <= R ) {
      mark(act, o);
    } else if (L <= R) {
      int mid = (l+r)>>1;
      pushdown(o);
      if( L <= mid ) upd(L, R, act, o<<1, l, mid);
      if( R > mid ) upd(L, R, act, o<<1|1, mid+1, r);
      maintain(o);
    }
  }

  node_t qry(int L, int R, int o, int l, int r) {
    if(L <= l && r <= R)
      return nodes[o];
    else if (L <= R) {
      int mid = (l+r)>>1;
      pushdown(o);
      if(R <= mid) return qry(L,R,o<<1,l,mid);
      if(L > mid) return qry(L,R,o<<1|1,mid+1,r);
      return node_t::merge(qry(L,R,o<<1,l,mid),qry(L,R,o<<1|1,mid+1,r));
    }
  }

  int N;

  segment_tree(void):N(STMAX) {}
  segment_tree(int n):N(n) {}
  void build(int n) {
    N = n;
    build(1,1,N);
  }
  void update(int L, int R, lazy_t act) {
    upd(L,R,act,1,1,N);
  }
  node_t query(int L, int R) {
    return qry(L,R,1,1,N);
  }
```

```
};
```

```
/* Segment tree (Interval tree, range tree), by Abreto <m@abreto.net>. */

#define MAXN    1000001

typedef struct {
  int l,r;
  /* Data field */
} node_t;

node_t merge(node_t n1, node_t n2) {
  node_t ans;
  ans.l = n1.l;
  ans.r = n2.r;
  /* merge n1 and n2 to ans. */
  return ans;
}

typedef struct {
  int marked; /* Optional */
  /* lazy mark. */
} lazy_t;

int A[MAXN];
node_t nodes[MAXN<<2];
lazy_t marks[MAXN<<2];

void maintain_leaf(int o, int idx) {
  nodes[o].l = nodes[o].r = idx;
  /* Operations to single elements ... */
}
void maintain(int o) {
  nodes[o] = merge(nodes[o<<1], nodes[o<<1|1]);
}

/* Usage: build(1,1,n); */
void build(int o, int l, int r) { /* [l,r] */
  if( r <= l ) {
    maintain_leaf(o, l);
  } else {
    int mid = l+r>>1;
    build(o<<1, l, mid);
    build(o<<1|1, mid+1, r);
    maintain(o);
  }
  marks[o].marked = 0;
}

/* Modify all elements in [l,r] */
void mark(lazy_t act, int o) {
  /* do something .. */
  marks[o].marked = 1;
}

/* Pass cached updates. */
void pushdown(int o) {
  if( marks[o].marked ) {
    mark(marks[o], o<<1);
    mark(marks[o], o<<1|1);
    marks[o].marked = 0;
  }
}

```

```
/* **DISCARDED** */
/* Set A[p]=v. Usage: modify(p, v, 1, 1, n);
void modify(int p, int v, int o, int l, int r)
{
    if( r - l < 2 )
    {
        maintain_leaf(o, v);
    } else {
        int mid = (l+r)/2;
        pushdown(o);
        if( p <= mid ) modify(p, v, o*2, l, mid);
        else modify(p, v, o*2+1, mid, r);
        maintain(o);
    }
}*/

/* Do act on all elements in [L,R] */
void update(int L, int R, lazy_t act, int o, int l, int r) {
  if( L <= l && r <= R ) {
    mark(act, o);
  } else if (L <= R) {
    int mid = (l+r)>>1;
    pushdown(o);
    if( L <= mid ) update(L, R, act, o<<1, l, mid);
    if( R > mid ) update(L, R, act, o<<1|1, mid+1, r);
    maintain(o);
  }
}
```

1.4 Sparse Table

```
/* RMQ with Sparse Table, by Abreto <m@abreto.net>. */

int min(int a, int b) {
  return (a<b)?a:b;
}

#define MAXN    100001
#define MAXLOG  32

int N;
int A[MAXN];    /* indexed from 0. */
int st[MAXN][MAXLOG];

void st_init() {
  int i = 0, j = 0, t = 0;
  for(i = 0; i < N; ++i) st[i][0] = A[i];
  for(j = 1; (t=(1<<j)) <= N; ++j)
    for(i = 0; (i+t-1) < N; ++i)
      st[i][j] = min(st[i][j-1], st[i+(t>>1)][j-1]);
  /* st(i,j) = min(st(i,j-1), st(i+2^(j-1),j-1)). */
}

int st_query(int l, int r) {
  int k = 0;
  while((1<<(k+1)) <= (r-l+1)) k++;
  return min(st[l][k], st[r-(1<<k)+1][k]);
}
```

1.5 Treap

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

using namespace std;

#define MAXN  (2560000)

int __treap_mem[MAXN];
void init_treap_mem(void) {
  for( int i = 1 ; i < MAXN ; i++ )
    __treap_mem[i-1] = i;
}
int alloc_address(void) {
  int ret = __treap_mem[0];
  __treap_mem[0] = __treap_mem[ret];
  return ret;
}
void free_address(int p) {
  __treap_mem[p] = __treap_mem[0];
  __treap_mem[0] = p;
}

typedef int key_t;
typedef int val_t;
struct treap {
  key_t x;
  val_t v;
  int r;    /* random priority */
  int eq, s;  /* number of equal ones, size of subtree (include root itself) */
  treap *fa;  /* point to its father */
  treap *ch[2]; /* 0 for left child, 1 for right child. */

  treap(void);
  inline void maintain(void); /* update s */
  inline void set_child(int d, treap *child);
  inline int which(void); /* determine which child this is of its father */
  inline int cmp(key_t ox); /* determine which child to insert ox */
  treap *rotate(void);  /* rotate this to its father, return this */
} treap_nodes[MAXN];

treap *new_treap(key_t x, val_t v, treap *f) {
  treap *ret = treap_nodes + alloc_address();
  ret->x = x;
  ret->v = v;
  ret->eq = ret->s = 1;
  ret->fa = f;
  ret->ch[0] = ret->ch[1] = NULL;
}
void free_treap(treap *p) {
  free_address( p - treap_nodes );
}
void treap_clear(void) {
  init_treap_mem();
}

treap::treap(void) {
  r = rand();
  eq = s = 0;
  fa = ch[0] = ch[1] = NULL;
}
inline void treap::maintain(void) {
  s = eq;
  for( int i = 0 ; i < 2 ; i++ )
    if( ch[i] )
      s += ch[i]->s;
}
inline void treap::set_child(int d, treap *child) {
  ch[d] = child;
  maintain();
  if( child ) child->fa = this;
}
inline int treap::which(void) {
  if( NULL == fa ) return -1; /* this is not a child */
  else return ( fa->ch[1] == this );
}
inline int treap::cmp(key_t ox) {
  if( ox == x ) return -1;  /* equal */
  else return ( ox > x );   /* left less, right more */
}
treap *treap::rotate(void) {
  if ( NULL == fa ) return this;  /* no father, already global root. */
  int d = which();
  fa->set_child(d, ch[d^1]);
  set_child(d^1, fa);
  return this;
}

// --- deprecated |
typedef int T;

struct node {
  T v;  /* value of this node */
  int r;  /* random priority */
  int eq; /* the number of equal things */
  int s;  /* the size of subtree rooted at this */
  node *ch[2];  /* 0 for left child, 1 for right child. */
  node(void) {
    r = rand();
    ch[0] = ch[1] = NULL;
  }
  /* return where to insert x */
  int cmp(T x) {
    if(v == x) return -1;
    else return (x < v) ? 0 : 1;
  }
  /* return 1 if this node is prior to other */
  int pri(node *o) {
    return (r > (o->r));
  }
  /* maintain the s field */
  void maintain(void) {
    s = eq;
    if(NULL != ch[0]) s += ch[0]->s;
    if(NULL != ch[1]) s += ch[1]->s;
  }
};

/* move o to ch[d] of o->ch[d^1] */
void rotate(node *&o, int d) {
  node *k = o->ch[d^1];
  o->ch[d^1] = k->ch[d];
  o->maintain();
  k->ch[d] = o;
  k->maintain();
  o = k;
}
```

1.6 Leftist Heap

```
/* HDU 1512 Monkey King（左偏树模板题） */
#include<iostream>
#include<cstdio>
using namespace std;
const int maxn = 100000+5;

int n, m;

struct Heap {
  int l,r,fa,val,dis;
} t[maxn];


int finds(int x) {
  return t[x].fa == -1? x:t[x].fa = finds(t[x].fa);
}

int merge(int x, int y) {
  if(x == 0)  return y;   //如果为0的话，就说明是空子树，根节点当然就是另一节点了
  if(y == 0)  return x;
  if(t[y].val>t[x].val)  swap(x,y);  //始终往右子树进行插入
  t[x].r = merge(t[x].r,y);
  t[t[x].r].fa = x;
  if(t[t[x].l].dis < t[t[x].r].dis) swap(t[x].l,t[x].r);   //是否需要左右子树的对换，这样是为了右子树尽量短
  if(t[x].r == 0)  t[x].dis = 0;   //距离的重新分配
  else t[x].dis = t[t[x].r].dis + 1;
  return x;
}

int pop(int &root) {
  int l = t[root].l;
  int r = t[root].r;
  t[root].l = t[root].r = t[root].dis = 0;
  t[root].fa = -1;
  t[l].fa = t[r].fa = -1;  //删除root根节点
  return merge(l,r);       //这样一来相当于分裂成了两棵子树，重新进行合并，最后返回值为合并后的根节点
}

int push(int x, int y) {
  return merge(x,y);
}

int main() {
  //freopen(”in.txt”,”r”,stdin);
  while(~scanf(”%d”,&n)) {
    for(int i=1; i<=n; i++) {
      t[i].l=t[i].r=t[i].dis=0;
      t[i].fa=-1;
      scanf(”%d”,&t[i].val);
    }
    scanf(”%d”,&m);
    while(m--) {
      int a,b;
      scanf(”%d%d”,&a,&b);
      int x=finds(a);
      int y=finds(b);
      if(x!=y) {
        t[x].val/=2;
        int xx = push( pop(x),x);
        t[y].val/=2;
        int yy = push( pop(y),y);
        printf(”%d\n”,t[merge(xx,yy)].val);
      } else puts(”-1”);
    }
  }
  return 0;
}
```

1.7 Splay

```
/* splay, by Abreto<m@abreto.net>. */

#ifndef NULL
#define NULL 0
#endif

struct node {
  node *f, *ch[2];
  int sz;
  node(node *fa = NULL, node *lc = NULL, node *rc = NULL) {
    f = fa;
    ch[0] = lc;
    ch[1] = rc;
    maintain();
  }
  inline int szof(const int d) const {
    return ch[d] ? ch[d]->sz : 0;
  }
  inline void maintain(void) {
    sz = szof(0) + szof(1) + 1;
  }
  inline int which(void) {
    if (NULL == f) return 0;
    return (f->ch[1] == this);  /* f[which()] == this */
  }
  inline node *setf(node *fa, int d = 0) {
    f = fa;
    if (f) {
      f->ch[d] = this;
      f->maintain();
    }
    return f;
  }
  inline node *setc(node *son, int d = 0) {
    ch[d] = son;
    if (son) son->f = this;
    maintain();
    return this;
  }
  /* rotate this to its fater, return this. */
  inline node *rotate(void) {
    if (f != NULL) {
      node *ff = f->f;
      int d = which(), fd = f->which();
      setc(f->setc(ch[d ^ 1], d), d ^ 1);
      setf(ff, fd);
    }
    return this;
  }
  /* splay this to child of target */
  inline node *splay(node * const target = NULL) {
    while (f != target) {
      if (target != f->f) {
        ( (which() == f->which()) ? f : this )->rotate();
      }
      rotate();
    }
    return this;
  }
  /* 0-based rank */
  inline node *get_k_th(unsigned k) {
    node *p = this;
    int rank;
    while (k != (rank = (p->szof(0)))) {
      if (k < rank) {
        p = p->ch[0];
      } else {
        k -= (rank + 1);
        p = p->ch[1];
      }
    }
    return p->splay(f);
  }
};
```

```
/* HDU 3487 - Play with Chain, by Abreto<m@abreto.net>. */
#include <bits/stdc++.h>

using namespace std;

#define MAXN    300300

int n, m;

#define LC(p)   ch[p][0]
#define RC(p)   ch[p][1]
#define TARGET(p) LC(RC(p))

int nodes;
int val[MAXN], ch[MAXN][2], fa[MAXN], sz[MAXN];
int rev[MAXN];

inline int new_node(int v, int f) {
  int p = (++nodes);
  val[p] = v;
  fa[p] = f;
  ch[p][0] = ch[p][1] = rev[p] = 0;
  sz[p] = 1;
  return p;
}
inline void maintain(int p) {
  if (p) {
    sz[p] = sz[LC(p)] + sz[RC(p)] + 1;
  }
}
inline void make_child(int f, int d, int p) { /* make p the d-th ch of f */
  ch[f][d] = p;
  if(p) fa[p] = f;
}
inline void myrev(int p) {
  if (p) {
    rev[p] ^= 1;
    swap(LC(p), RC(p));
  }
}
```

```
inline void pushdown(int p) {
  if(p && rev[p]) {
    if(LC(p)) myrev(LC(p));
    if(RC(p)) myrev(RC(p));
    rev[p] = 0;
  }
}
int build(int f = 0, int l = 0, int r = n+1) {
  if(r < l) return 0;
  if(l == r) return new_node(l, f);
  int mid = l+r>>1;
  int p = new_node(mid, f);
  LC(p) = build(p, l, mid-1);
  RC(p) = build(p, mid+1, r);
  maintain(p);
  return p;
}
inline int which(int p) {  /* return 1 if p is a right child or 0 if p is a left child. */
  return (RC(fa[p]) == p);
}
inline int rotate(int p) { /* rotate p to its father. [!] make sure p is not global root. */
  int f = fa[p], ff = fa[f];
  if(0 == f) return p;  /* p is global root */
  pushdown(f);
  pushdown(p);
  int d = which(p), df = which(f);
  make_child(f, d, ch[p][d^1]);
  make_child(p, d^1, f);
  maintain(f);
  maintain(p);
  fa[p] = ff;
  if(ff) ch[ff][df] = p;
  return p;
}
inline int splay(int p, int fr) { /* splay p to the son of fr, return p. */
  pushdown(p);
  while(fa[p] != fr) {
    int f = fa[p], dp = which(p);
    if(fa[f] == fr) {
      return rotate(p);
    } else {
      int df = which(f);
      if(dp == df) {
        rotate(f);
      } else {
        rotate(p);
      }
      rotate(p);
    }
  }
  return p;
}
inline int get_k_th(int root, int k) {
  int p = root;
  int rank;
  while(k != (rank = (sz[LC(p)] + 1))) {
    pushdown(p);
    if(k < rank) p = LC(p);
    else {
      k -= rank;
      p = RC(p);
    }
  }
  return splay(p, fa[root]);
}
inline int merge(int left, int right) {
  pushdown(left);
  if(RC(left)) left = get_k_th(left, sz[left]);
  RC(left) = right;
  maintain(left);
  fa[right] = left;
  return left;
}
inline int split(int root, int d) { /* split ch[root][d], return the root of splited out. */
  pushdown(root);
  int child = ch[root][d];
  ch[root][d] = 0;
  maintain(root);
  fa[child] = 0;
  return child;
}
inline int concat(int root, int d, int p) { /* make p be ch[root][d], return root */
  pushdown(root);
  ch[root][d] = p;
  fa[p] = root;
  maintain(root);
  return root;
}

void myclear(void) {
  nodes = 0;
}

int ans[MAXN];
void inorder(int p, int &pos) {
  if(0 == p) return;
  pushdown(p);
  inorder(LC(p), pos);
  if( (0 < val[p]) && (val[p] < n+1) ) ans[pos++] = val[p];
  inorder(RC(p), pos);
}

void handle() {
  int i;
  int root;
  myclear();
  root = build(0);
  while(m--) {
    char command[8];
    int a, b, c;
    int tar;
    scanf(”%s%d%d”, command, &a, &b);
    if(’C’ == command[0]) {
      scanf(”%d”, &c);
      root = get_k_th(root, a);
      RC(root) = get_k_th(RC(root), b-a+2);
      tar = split(RC(root), 0);
      maintain(root);
      root = get_k_th(root, c+1);
      RC(root) = get_k_th(RC(root), 1);
      RC(root) = concat(RC(root), 0, tar);
      maintain(root);
    } else {
      root = get_k_th(root, a);
      RC(root) = get_k_th(RC(root), b-a+2);
      myrev(TARGET(root));
    }
  }
  int pos = 0;
  inorder(root, pos);
  for(i = 0; i < n; i++) printf(”%s%d”, i ? ”␣”:””, ans[i]);
  puts(””);
}

int main(void) {
  while( scanf(”%d%d”, &n, &m) && (n > 0) && (m > 0) )
    handle();
  return 0;
}
```

1.8 Persistent Segment Tree

1. 首先, 给你一颗值为横坐标的线段树, 每个节点上存着该值出现了多少次, 这样的一颗线段树你会求区间 k 大值吧. 二分即可.
2. 然后, 假设区间是数组 $arr[n]$, 区间长度是 n, 那么给你 n 颗线段树, 第 i 颗线段树是第 $i-1$ 颗线段树插入 $arr[i]$ 得到.
3. 如果你有了这 n 颗线段树, 想求区间 $[l,r]$ 中的第 k 大值, 那么你需要在第 r 颗和第 $l-1$ 颗线段树的差线段树上作二分, 就可以求得区间第 k 大值.
4. 差线段树很好理解, 比如你有一个部分和数组 sum, $sum[r]-sum[l-1]$ 就是部分和的差, 代表区间 $[l,r]$ 的和, 差线段树同理.
5. 现在, 可持久化线段树出现为你解决最后一个问题, 空间问题. 内存很小, 不能够存下 n 颗线段树. 但是, 第 2 条中提到, 由于第 i 颗线段是是第 $i-1$ 颗线段是插入仅一个值得到的, 两颗线段树的区别不大, 仅有 $\log(n)$ 个节点发生了改变, 我们仅仅需要记录这 $\log(n)$ 的数据就可以记录这个增量, 这就是可持久化线段树.

2 Dynamic Programming

2.1 LIS $O(n\log n)$

```

int top = 0;
for( int i=1; i<=n; i++ ) {
  if( ap[i] > dp[top] ) { // 如果大于 ”模拟栈” 的栈顶元素直接 入栈 长度加 1
    top++;
    dp[top] = ap[i];
    continue;
  }
  int m = ap[i];
  // lower_bound 前闭后开 返回不小于 m 的最小值的位置
  pos = lower_bound(dp,dp+top,m)-dp; // 注意减去dp
  if( dp[pos] > ap[i])
    dp[pos] = ap[i];
}
```

2.2 LCS $O(n\log n)$

总的来说，就是把 LCS 转化成 LIS，然后用 LIS 的 $O(N\log N)$ 算法来求解。
实现如下：（引用）
假设有两个序列 $s_1[1\ldots6]=abcadc$, $s_2[1\ldots7]=cabedab$.

记录 s_1 中每个元素在 s_2 中出现的位置, 再将位置按降序排列, 则上面的例子可表示为:
$loc(a) = \{6, 2\}, loc(b) = \{7, 3\}, loc(c) = \{1\}, loc(d) = \{5\}$. (倒着扫一遍 s_2 即可把位置扔进 `vector`).
将 s_1 中每个元素的位置按 s_1 中元素的顺序排列成一个序列 $s_3 = \{6, 2, 7, 3, 1, 6, 2, 5, 1\}$.
在对 s_3 求 LIS 得到的值即为求 LCS 的答案。

2.3 Improved by quadrilateral inequality

```
/*
 * 四边形不等式
 *
 * 如果 dp(i,j) 满足 dp(i,j)<=dp(i,j+1)<=dp(i+1,j+1)
 * 那么决策 s(i,j) 满足 s(i,j)<=s(i,j+1)<=s(i+1,j+1)
 * 可以变形为:
 *      s(i-1,j) <= s(i,j) <= s(i,j+1)  // dp方向: i增j减
 *  或
 *      s(i,j-1) <= s(i,j) <= s(i+1,j)  // dp方向: 区间长度L增
 */
#include <bits/stdc++.h>

using namespace std;

#define MAXN    1024
#define inf     (0x3fffffff)

int n, m;
int v[MAXN];
int s[MAXN];
int w[MAXN][MAXN];
int dp[MAXN][MAXN];
int c[MAXN][MAXN];

int wa(void) {
  int i, j, k;
  for(i = 1; i <= n; ++i) {
    scanf("%d", v+i);
    s[i] = v[i] + s[i-1];
  }
  for(i = 1; i <= n; ++i) {
    w[i][i] = 0;
    for(j = i+1; j <= n; ++j)
      w[i][j] = w[i][j-1] + v[j] * (s[j-1] - s[i-1]);
  }
  /* doing dp */
  for(i = 1; i <= n; ++i) {
    dp[i][0] = w[1][i];
    c[i][0] = 1;
    c[i][i] = i-1;
    for(j = i-1; j > 0; j--) {
      dp[i][j] = inf;
      for(k = c[i-1][j]; k <= c[i][j+1]; ++k)
        if(dp[k][j-1]+w[k+1][i] <= dp[i][j]) {
          dp[i][j] = dp[k][j-1] + w[k+1][i];
          c[i][j] = k;
        }
    }
  }
  /* dp done */
  return dp[n][m];
}

int main(void) {
  while(EOF != scanf("%d%d", &n, &m) && n && m) {
    printf(”%d\n”, wa());
  }
  return 0;
}
```

2.4 Improved by Slope

```
/* type 1: */
/* bzoj 1010 */
#include <bits/stdc++.h>

using namespace std;
typedef long double ll;
#define MAXN    50050
#define eps     (1e-8)

int N;
ll L;
ll S[MAXN];
ll f[MAXN];
ll dp[MAXN];

inline ll k(int j) {
  return (-2.0) * (f[j] + L);
}
inline ll b(int j) {
  return dp[j] + f[j]*f[j] + 2ll*f[j]*L;
}
inline ll g(int j, int i) {
  return k(j) * f[i] + b(j);
}

/* check if l1 & l3 <= l2 */
inline int check(int l1, int l2, int l3) {
  /*ll left = b(l3)*k(l1)+b(l1)*k(l2)+b(l2)*k(l3);
  ll right = b(l1)*k(l3)+b(l3)*k(l2)+b(l2)*k(l1);*/
  ll left = b(l3)*k(l1)-b(l1)*k(l3);
  ll right = k(l2)*(b(l3)-b(l1))+b(l2)*(k(l1)-k(l3));
  return (left <= right);
}

int Q[MAXN], ql, qr;

int main(void) {
  int i;
  scanf(”%d%Lf”, &N, &L);
  L += 1.0;
  for(i = 1; i <= N; ++i) {
    scanf(”%Lf”, S+i);
    S[i] += S[i-1];
    f[i] = S[i] + (double)i;
  }
  Q[qr++] = 0;
  for(i = 1; i <= N; ++i) {
    /* <!-- STARED */
    for(; ql+1 < qr && g(Q[ql],i) >= g(Q[ql+1],i); ql++);
    dp[i] = g(Q[ql], i) + f[i]*f[i] + L*L; //printf(”%d: %lld,%lld\n”, i, dp[i], dp[i]-f[i]*f[i]);
    for(; ql+1 < qr && check(Q[qr-2], Q[qr-1], i); qr--);
    Q[qr++] = i;
    /* --> */
  }
  printf(”%lld\n”, (long long int)round(dp[N]));
  return 0;
}
```

2.5 Steiner Tree

令 $f[i][sta]$ 表示 i 号节点，与其他节点的连通性为 sta 时的最小代价，这里 sta 是一个二进制数，在它二进制下的每一位中，0 表示不连通，1 表示联通
状态转移：

- 由子集转移而来
$$f[i][sta] = \min_{s\subseteq sta}\{f[i][s] + f[i][sta \setminus s] - val[i]\}$$
- 由不含该节点的状态转移而来
$$f[i][j] = \min\{f[k][j] + val[i]\}$$

流程：

```
枚举状态集S
{
     枚举S的子集s
     {
         更新f[S][1~n]
     }
     将 f[S][x]<inf 的x入队
     spfa(S)
}
```

代码：

```
int f[1<<M][N];
queue<int> q;
bool in[N];

void spfa(int S) {
  while (!q.empty()) {
    int now=q.front();
    q.pop();
    in[now]=0;
    for (int i=st[now]; i; i=way[i].nxt) {
      int y=way[i].y;
      if (f[S][y]>f[S][now]+val[y]) {
        f[S][y]=f[S][now]+val[y];
        if (!in[y]) q.push(y),in[y]=1;
      }
    }
  }
}

void work() {
  int cnt=0;
  memset(f,0x7f,sizeof(f));

  for (int i=1; i<=n; i++)
    if (!val[i]) f[1<<cnt][i]=0,cnt++;
  for (int S=1; S<(1<<cnt); S++) {
    for (s=(S-1)&S; s; s=(s-1)&S)
      for (int i=1; i<=n; i++)
        f[S][i]=min(f[S][i],f[s][i]+f[S^s][i]-val[i]);
    for (int i=1; i<=n; i++)
      if (f[S][i]<INF&&!in[i])
```

```
        q.push(i),in[i]=1;
    spfa(S);
  }

  int ans=INF;
  for (int i=1; i<=n; i++) ans=min(ans,f[(1<<cnt)-1][i]);
  printf(”%d\n”,ans);
}
// ———————————————————
// 作者：Coco_T_
// 来源：CSDN
// 原文：https://blog.csdn.net/wu_tongtong/article/details/78992913
// 版权声明：本文为博主原创文章，转载请附上博文链接！
```

3 Geometry

3.1 2D

3.1.1 Point

```
/* 2D Point Class, by Abreto<m@abreto.net> */
#include <cmath>

/**
 * Define ABG2d_USE_LL if you want to use long long int for cordnates.
 */

namespace ab_geometry_2d {

using namespace std;

typedef double ab_float;

const ab_float pi = acos(-1.);

#ifdef ABG2d_USE_LL
typedef long long int T;
#else
typedef ab_float T;
const ab_float eps = 1e-8;
#endif

inline T myabs(T x) {
  if(x < 0) return (-x);
  return x;
}

inline int sgn(T x) {
  /* no difference’’ in fact */
#ifdef ABG2d_USE_LL
  if (0 == x) return 0;
#else
  if (myabs(x) < eps) return 0;
#endif
  return (x > 0) ? 1 : -1;
}

inline T sqr(T x) {
  return (x * x);
}

struct point {
  T x,y;
  point(void):x(T()),y(T()) {}
  point(T xx, T yy):x(xx),y(yy) {}
  inline T norm2(void) {
    return sqr(x) + sqr(y);
  }
  inline ab_float norm(void) {
    return sqrt((ab_float)(norm2()));
  }
  inline point rotate(const ab_float &cost, const ab_float &sint) {} // TODO:
  inline point operator-(void) const {
    return point(-x,-y);
  }
  inline point operator+(const point& b) const {
    return point(x+b.x,y+b.y);
  }
  inline point operator-(const point& b) const {
    return point(x-b.x,y-b.y);
  }
  inline point operator->*(const point &b) const {
    return (b-(*this));
  }
  inline T operator*(const point& b) const {
    return ((x)*(b.x))+((y)*(b.y)); /* inner product */
  }
  inline T operator^(const point& b) const {
    return ((x)*(b.y))-((b.x)*(y)); /* outter product */
  }
  inline point& operator+=(const point& b) {
    point tmp=(*this)+b;
    (*this)=tmp;
    return (*this);
  }
  inline point& operator-=(const point& b) {
    point tmp=(*this)-b;
    (*this)=tmp;
    return (*this);
  }
  inline bool operator==(const point& b) const {
    return (0==sgn(x-b.x))&&(0==sgn(y-b.y));
  }
  inline bool operator!=(const point& b) const {
    return !((*this)==b);
  }
  inline point operator<<(const ab_float& theta) const {
    ab_float ct = cos(theta), st = sin(theta);  /* rotate counter-clockwise in radian */
    return point(ct*x - st*y, st*x + ct*y);
  }
};

typedef point vec;


}   // namespace ab_geometry_2d
```

3.1.2 Circle

Base

```
/* 2D Circle Base Class, by Abreto<m@abreto.net>. */

/* requirement: point.cc */
```

```
#include "point.cc"

#include <utility>

namespace ab_geometry_2d {

using namespace std;

struct circle {
  point o;
  T r;
  circle(void) : r(T()) {}
  circle(point center, T radius) : o(center), r(radius) {}

  inline ab_float arclen(ab_float theta) {
    return theta * r;
  }
  inline ab_float circumference(void) {
    return 2. * pi * r;
  }
  inline ab_float area(void) {
    return pi * r * r;
  }

  /* bool contain(const circle &C, const bool including_touch = false) const
  {
      T dis2 = (o->*(C.o)).norm2();
      T raw_diff = r - C.r;
      if ( -1 == sgn(raw_diff) ) return false;
      T dr2 = sqr(raw_diff);
      return (dis2 < dr2) || (including_touch && (dis2 == dr2));
  }
  inline bool in(const circle &C, const bool including_touch = false) const
  {
    return C.contain(*this, including_touch);
  } */
  enum relation_t {
    same = 0x00000,
    contain = 0x00001,
    intouch = 0x00010,
    intersect = 0x00100,
    outtouch = 0x01000,
    separate = 0x10000,
    unknow_relation = 0xfffff
  };
  relation_t with(const circle &C) const {
    T dis2 = (o->*(C.o)).norm2();
    T dr2 = sqr(r - C.r), rs2 = sqr(r + C.r);
    if ( 0 == sgn(dis2) && 0 == sgn(dr2) ) return same;
    if ( -1 == sgn(dis2 - dr2) ) return contain;
    if ( 0 == sgn(dis2 - dr2) ) return intouch;
    if ( -1 == sgn(dr2 - dis2) && -1 == sgn(dis2 - rs2) ) return intersect;
    if ( 0 == sgn(dis2 - rs2) ) return outtouch;
    if ( -1 == sgn(rs2 - dis2) ) return separate;
    return unknow_relation;
  }

  enum point_relation_t {
    in = 0x001,
    on = 0x010,
    out = 0x100,
    unknow_point_relation = 0xfff
  };
  point_relation_t with(const point &P) const {
    T dis2 = (o->*P).norm2();
    T r2 = sqr(r);
    int type = sgn(dis2 - r2);
    if (-1 == type) return in;
    if ( 0 == type) return on;
    if (+1 == type) return out;
    return unknow_point_relation;
  }

  ab_float central_angle(const point &A, const point &B, const bool reflex = false) const {
    T dot = (A * B);
    if (0 == sgn(dot)) return 1. * (A != B) * pi;
    ab_float angle = ((ab_float)(dot)) / r / r;
    if ( reflex ) angle = 2. * pi - angle;
    return angle;
  }

  /* be sure (*this) intersect with C */
  pair<point,point> crosspoint(const circle &C) const {
    ab_float d = (o ->* (C.o)).norm();
    // TODO:
  }
};

}
```

k 次圆交

```
//china no.1
#pragma comment(linker, ”/STACK:1024000000,1024000000”)
#include <vector>
#include <iostream>
#include <string>
#include <map>
#include <stack>
#include <cstring>
#include <queue>
#include <list>
#include <stdio.h>
#include <set>
#include <algorithm>
#include <cstdlib>
#include <cmath>
#include <iomanip>
#include <cctype>
#include <sstream>
#include <functional>
#include <stdlib.h>
#include <time.h>
#include <bitset>
using namespace std;

#define pi acos(-1)
#define PI acos(-1)
#define endl ’\n’
#define srand() srand(time(0));
#define me(x,y) memset(x,y,sizeof(x));
#define foreach(it,a) for(__typeof((a).begin()) it=(a).begin();it!=(a).end();it++)
#define close() ios::sync_with_stdio(0); cin.tie(0);
#define FOR(x,n,i) for(int i=x;i<=n;i++)
#define FOr(x,n,i) for(int i=x;i<n;i++)
#define W while
#define sgn(x) ((x) < 0 ? -1 : (x) > 0)
#define bug printf(”***********\n”);
```

```
#define db double
typedef long long LL;
const int INF=0x3f3f3f3f;
const LL LINF=0x3f3f3f3f3f3f3f3fLL;
const int dx[]= {-1,0,1,0,1,-1,-1,1};
const int dy[]= {0,1,0,-1,-1,1,-1,1};
const int maxn=1e3+10;
const int maxx=1e6+100;
const double EPS=1e-8;
const double eps=1e-8;
const int mod=10000007;
template<class T>inline T min(T a,T b,T c) {
  return min(min(a,b),c);
}
template<class T>inline T max(T a,T b,T c) {
  return max(max(a,b),c);
}
template<class T>inline T min(T a,T b,T c,T d) {
  return min(min(a,b),min(c,d));
}
template<class T>inline T max(T a,T b,T c,T d) {
  return max(max(a,b),max(c,d));
}
template <class T>
inline bool scan_d(T &ret) {
  char c;
  int sgn;
  if (c = getchar(), c == EOF) {
    return 0;
  }
  while (c != ’-’ && (c < ’0’ || c > ’9’)) {
    c = getchar();
  }
  sgn = (c == ’-’) ? -1 : 1;
  ret = (c == ’-’) ? 0 : (c - ’0’);
  while (c = getchar(), c >= ’0’ && c <= ’9’) {
    ret = ret * 10 + (c - ’0’);
  }
  ret *= sgn;
  return 1;
}

inline bool scan_lf(double &num) {
  char in;
  double Dec=0.1;
  bool IsN=false,IsD=false;
  in=getchar();
  if(in==EOF) return false;
  while(in!=’-’&&in!=’.’&&(in<’0’||in>’9’))in=getchar();
  if(in==’-’) {
    IsN=true;
    num=0;
  } else if(in==’.’) {
    IsD=true;
    num=0;
  } else num=in-’0’;
  if(!IsD) {
    while(in=getchar(),in>=’0’&&in<=’9’) {
      num*=10;
      num+=in-’0’;
    }
  }
  if(in!=’.’) {
    if(IsN) num=-num;
    return true;
  } else {
    while(in=getchar(),in>=’0’&&in<=’9’) {
      num+=Dec*(in-’0’);
      Dec*=0.1;
    }
  }
  if(IsN) num=-num;
  return true;
}

void Out(LL a) {
  if(a < 0) {
    putchar(’-’);
    a = -a;
  }
  if(a >= 10) Out(a / 10);
  putchar(a % 10 + ’0’);
}
void print(LL a) {
  Out(a),puts(””);
}
//freopen( ”in.txt” , ”r” , stdin );
//freopen( ”data.txt” , ”w” , stdout );
//cerr << ”run time is ” << clock() << endl;
/*struct Point
{
    double x, y;
    Point(const Point& rhs): x(rhs.x), y(rhs.y) { } //拷贝构造函数
    Point(double x = 0, double y = 0) : x(x), y(y) { }
    inline void input()
    {
        scanf(”%lf%lf”,&x,&y);
    }
    inline void print()
    {
        printf(”%.6lf %.6lf\n”,x,y);
    }
};*/
db sqr(db x) {
  return x*x;
}
int dcmp(double x) {
  if(fabs(x) < EPS) return 0;
  else return x < 0 ? -1 : 1;
}
struct Circle {
  double x, y, r, angle;
  int d;
  Circle() {}
  Circle(double xx, double yy, double ang = 0, int t = 0) {
    x = xx;
    y = yy;
    angle = ang;
    d = t;
  }
  void get() {
    scanf(”%lf%lf%lf”, &x, &y, &r);
    d = 1;
  }
};
Circle cir[maxn],tp[maxn*2];
double area[maxn];
double dis(Circle a,Circle b) {
  return sqrt(sqr(a.x - b.x) + sqr(a.y - b.y));
}
double cross(Circle p0,Circle p1,Circle p2) {
  return (p1.x - p0.x) * (p2.y - p0.y) - (p1.y - p0.y) * (p2.x - p0.x);
}
//圆相交
int CirCrossCir(Circle p1, double r1,Circle p2, double r2,Circle &cp1,Circle &cp2) {
  double mx = p2.x - p1.x, sx = p2.x + p1.x, mx2 = mx * mx;
  double my = p2.y - p1.y, sy = p2.y + p1.y, my2 = my * my;
  double sq = mx2 + my2, d = -(sq - sqr(r1 - r2)) * (sq - sqr(r1 + r2));
  if (d + eps < 0) return 0;
  if (d < eps) d = 0;
  else d = sqrt(d);
  double x = mx * ((r1 + r2) * (r1 - r2) + mx * sx) + sx * my2;
  double y = my * ((r1 + r2) * (r1 - r2) + my * sy) + sy * mx2;
  double dx = mx * d, dy = my * d;
  sq *= 2;
  cp1.x = (x - dy) / sq;
  cp1.y = (y + dx) / sq;
  cp2.x = (x + dy) / sq;
  cp2.y = (y - dx) / sq;
  if (d > eps) return 2;
  else return 1;
}
bool circmp(const Circle& u, const Circle& v) {
  return dcmp(u.r - v.r) < 0;
}
bool cmp(const Circle& u, const Circle& v) {
  if (dcmp(u.angle - v.angle)) return u.angle < v.angle;
  return u.d > v.d;
}
//0.5*r*r*(K-sin(K))
double calc(Circle cir,Circle cp1,Circle cp2) {
  double ans = (cp2.angle - cp1.angle) * sqr(cir.r)
               - cross(cir, cp1, cp2) + cross(Circle(0, 0), cp1, cp2);
  return ans / 2;
}

void CirUnion(Circle cir[], int n) {
  Circle cp1, cp2;
  sort(cir, cir + n, circmp);
  for (int i = 0; i < n; ++i)
    for (int j = i + 1; j < n; ++j)
      if (dcmp(dis(cir[i], cir[j]) + cir[i].r - cir[j].r) <= 0)
        cir[i].d++;
  for (int i = 0; i < n; ++i) {
    int tn = 0, cnt = 0;
    for (int j = 0; j < n; ++j) {
      if (i == j) continue;
      if (CirCrossCir(cir[i], cir[i].r, cir[j], cir[j].r,
                      cp2, cp1) < 2) continue;
      cp1.angle = atan2(cp1.y - cir[i].y, cp1.x - cir[i].x);
      cp2.angle = atan2(cp2.y - cir[i].y, cp2.x - cir[i].x);
      cp1.d = 1;
      tp[tn++] = cp1;
      cp2.d = -1;
      tp[tn++] = cp2;
      if (dcmp(cp1.angle - cp2.angle) > 0) cnt++;
    }
    tp[tn++] = Circle(cir[i].x - cir[i].r, cir[i].y, pi, -cnt);
    tp[tn++] = Circle(cir[i].x - cir[i].r, cir[i].y, -pi, cnt);
    sort(tp, tp + tn, cmp);
    int p, s = cir[i].d + tp[0].d;
    for (int j = 1; j < tn; ++j) {
      p = s;
      s += tp[j].d;
      area[p] += calc(cir[i], tp[j - 1], tp[j]);
    }
  }
}
int n;
void solve() {
  for(int i=0; i<n; i++)
    cir[i].get();
  me(area,0);
  CirUnion(cir,n);
  for(int i=1; i<=n; i++) {
    area[i]-=area[i+1];
    printf(”[%d]␣=␣%.3f\n”, i, area[i]);
  }
}
int main() {
  while(scanf(”%d”,&n)!=EOF)
    solve();
}
```

universe

```

Point CircumCenter(Point a,Point b,Point c) { //三角形的外心
  Point cp;
  double a1 = b.x-a.x,b1 = b.y-a.y,c1 = (a1*a1 + b1*b1)/2;
  double a2 = c.x-a.x,b2 = c.y-a.y,c2 = (a2*a2 + b2*b2)/2;
  double d = a1*b2 - a2*b1;
  cp.x = a.x + (c1*b2-c2*b1)/d;
  cp.y = a.y + (a1*c2-a2*c1)/d;
  return cp;
}
```

3.1.3 Convex hull

```
/* 2D Convex Hull, by Abreto <m@abreto.net>. */
#include ”2d_base.hh”
#include <cmath>
#include <algorithm>

using namespace std;

point O;

bool comp_angle(point_t a, point_t b) {
  double t = (a-O).X(b-O);
  if(fe(t,0.0)) return fl((b-O).mag2(),(a-O).mag2());
  else return fl(0.0,t);
}

void convex_hull_graham(vp& convex, vp src) {
  int i = 0, top = 0;
  O = src[0];
  for(auto pt : src)
    if( pt.x < O.x || (pt.x == O.x && pt.y < O.y))
      O = pt;
  sort(src.begin(), src.end(), comp_angle);
  convex.push_back(src[0]);
  convex.push_back(src[1]);
  top = 1;
  for(i = 2; i < src.size(); ++i) {
    while(top>1 && fle((convex[top]-convex[top-1]).X(src[i]-convex[top]),0.0)) {
      convex.pop_back();
      --top;
    }
    convex.push_back(src[i]);
    ++top;
  }
}
```

3.1.4 Intersect Area

```
#include <cstdio>
#include <cmath>
#include <algorithm>

using namespace std;

//#define inf 1000000000000
#define M 8
#define LL long long
#define eps 1e-12
#define PI acos(-1.0)
using namespace std;
struct node {
  double x,y;
  node() {}
  node(double xx,double yy) {
    x=xx;
    y=yy;
  }
  node operator -(node s) {
    return node(x-s.x,y-s.y);
  }
  node operator +(node s) {
    return node(x+s.x,y+s.y);
  }
  double operator *(node s) {
    return x*s.x+y*s.y;
  }
  double operator ^(node s) {
    return x*s.y-y*s.x;
  }
};
double max(double a,double b) {
  return a>b?a:b;
}
double min(double a,double b) {
  return a<b?a:b;
}
double len(node a) {
  return sqrt(a*a);
}
double dis(node a,node b) { //两点之间的距离
  return len(b-a);
}
double cross(node a,node b,node c) { //叉乘
  return (b-a)^(c-a);
}
double dot(node a,node b,node c) { //点成
  return (b-a)*(c-a);
}
int judge(node a,node b,node c) { //判断c是否在ab线段上（前提是c在直线ab上）
  if(c.x>=min(a.x,b.x)
      &&c.x<=max(a.x,b.x)
      &&c.y>=min(a.y,b.y)
      &&c.y<=max(a.y,b.y))
    return 1;
  return 0;
}
double area(node b,node c,double r) {
  node a(0.0,0.0);
  if(dis(b,c)<eps)
    return 0.0;
  double h=fabs(cross(a,b,c))/dis(b,c);
  if(dis(a,b)>r-eps&&dis(a,c)>r-eps) { //两个端点都在圆的外面则分为两种情况
    double angle=acos(dot(a,b,c)/dis(a,b)/dis(a,c));
    if(h>r-eps) {
      return 0.5*r*r*angle;
    } else if(dot(b,a,c)>0&&dot(c,a,b)>0) {
      double angle1=2*acos(h/r);
      return 0.5*r*r*fabs(angle-angle1)+0.5*r*r*sin(angle1);
    } else {
      return 0.5*r*r*angle;
    }
  } else if(dis(a,b)<r+eps&&dis(a,c)<r+eps) { //两个端点都在圆内的情况
    return 0.5*fabs(cross(a,b,c));
  } else { //一个端点在圆上一个端点在圆内的情况
    if(dis(a,b)>dis(a,c)) { //默认b在圆内
      swap(b,c);
    }
    if(fabs(dis(a,b))<eps) { //ab距离为0直接返回0
      return 0.0;
    }
    if(dot(b,a,c)<eps) {
      double angle1=acos(h/dis(a,b));
      double angle2=acos(h/r)-angle1;
      double angle3=acos(h/dis(a,c))-acos(h/r);
      return 0.5*dis(a,b)*r*sin(angle2)+0.5*r*r*angle3;

    } else {
      double angle1=acos(h/dis(a,b));
      double angle2=acos(h/r);
      double angle3=acos(h/dis(a,c))-angle2;
      return 0.5*r*dis(a,b)*sin(angle1+angle2)+0.5*r*r*angle3;
    }
  }
}

node A, B, C;
int R;

bool compar(node &p1, node &p2) {
  return (p1^p2)>eps;
}

double f(double x, double y) {
  node O(x,y);
  node p[8];
  p[0] = A-O;
  p[1] = B-O;
  p[2] = C-O;
  sort(p, p+3, compar);
  p[3] = p[0];
  O=node(0,0);
  double sum=0;
  /* <!-- 求面积交部分 */
  for(int i=0; i<3; i++) { /* 按顺或逆时针顺序最后取绝对值就好 */
```

```
    int j=i+1;
    double s=area(p[i],p[j],(double)R);
    if(cross(O,p[i],p[j])>0)
      sum+=s;
    else
      sum-=s;
  }
  if(sum < -eps) sum = -sum;
  /* --> */
  return sum;
}

double trifind(double x, double y1, double y2) {
  double l = y1, r = y2;
  while(r-l>eps) {
    double mid = (l+r)/2.0;
    double mmid = (mid+r)/2.0;
    if( f(x,mmid) > f(x,mid)+eps )
      l = mid;
    else
      r = mmid;
  }
  return f(x,l);
}

double findmin(double x1, double x2, double y1, double y2) {
  double l = x1, r = x2;
  while(r-l>eps) {
    double mid = (l+r)/2.0;
    double mmid = (mid+r)/2.0;
    if( trifind(mmid,y1,y2) > trifind(mid,y1,y2)+eps )
      l = mid;
    else
      r = mmid;
  }
  return trifind(l,y1,y2);
}

double ans(int a, int b, int c, int r) {
  A = node(0,0);
  B = node((double)c,0);
  R = r;
  double da = a, db = b, dc = c;
  double cosa = (db*db+dc*dc-da*da)/(2.0*db*dc);
  double alpha = acos(cosa);
  C = node(db*cosa, db*sin(alpha));
  return findmin(0.0, c, 0.0, db*sin(alpha));
}

int main(void) {
  int a = 0, b = 0, c = 0, r = 0;
  while(EOF != scanf(”%d%d%d%d”,&a,&b,&c,&r) && (a||b||c||r))
    printf(”%.8lf\n”, ans(a,b,c,r));
  return 0;
}
```

3.1.5 Universe

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

struct Point {
  double x, y;
  Point(double x = 0, double y = 0) : x(x), y(y) {}
};

typedef Point Vector;

Vector operator + (Vector A, Vector B) {
  return Vector(A.x + B.x, A.y + B.y);
}
Vector operator - (Vector A, Vector B) {
  return Vector(A.x - B.x, A.y - B.y);
}
Vector operator * (Vector A, double p) {
  return Vector(A.x*p, A.x*p);
}
Vector operator / (Vector A, double p) {
  return Vector(A.x/p, A.x/p);
}

bool operator < (const Point& a, const Point b) {
  return a.x < b.x || (a.x == b.x && a.y < b.y);
}

const double EPS = 1e-10;

int dcmp(double x) {
  if(fabs(x) < EPS) return 0;
  else return x < 0 ? -1 : 1;
}

bool operator == (const Point& a, const Point& b) {
  return dcmp(a.x-b.x) == 0 && dcmp(a.y-b.y);
}

//向量a的极角
double Angle(const Vector& v) {
  return atan2(v.y, v.x);//\share\CodeBlocks\templates\wizard\console\cpp
}

//向量点积
double Dot(Vector A, Vector B) {
  return A.x*B.x + A.y*B.y;
}

//向量长度\share\CodeBlocks\templates\wizard\console\cpp
double Length(Vector A) {
  return sqrt(Dot(A, A));
}

//向量夹角
double Angle(Vector A, Vector B) {
  return acos(Dot(A, B) / Length(A) / Length(B));
}

//向量叉积
double Cross(Vector A, Vector B) {
  return A.x*B.y - A.y*B.x;
}

//三角形有向面积的二倍
double Area2(Point A, Point B, Point C) {
  return Cross(B-A, C-A);
}

//向量逆时针旋转rad度(弧度)
```

```
Vector Rotate(Vector A, double rad) {
  return Vector(A.x*cos(rad)-A.y*sin(rad), A.x*sin(rad)+A.y*cos(rad));
}

//计算向量A的单位法向量。左转90°，把长度归一。调用前确保A不是零向量。
Vector Normal(Vector A) {
  double L = Length(A);
  return Vector(-A.y/L, A.x/L);
}

/************************************************************************
使用复数类实现点及向量的简单操作

#include <complex>
typedef complex<double> Point;
typedef Point Vector;

double Dot(Vector A, Vector B) { return real(conj(A)*B)}
double Cross(Vector A, Vector B) { return imag(conj(A)*B);}
Vector Rotate(Vector A, double rad) { return A*exp(Point(0, rad)); }

*************************************************************************/

/****************************************************************************
* 用直线上的一点p0和方向向量v表示一条指向。直线上的所有点P满足P = P0+t*v;
* 如果知道直线上的两个点则方向向量为B-A，所以参数方程为A+(B-A)*t;
* 当t 无限制时， 该参数方程表示直线。
* 当t > 0时， 该参数方程表示射线。
* 当 0 < t < 1时， 该参数方程表示线段。
*****************************************************************************/

//直线交点,须确保两直线有唯一交点。
Point GetLineIntersection(Point P, Vector v, Point Q, Vector w) {
  Vector u = P - Q;
  double t = Cross(w, u)/Cross(v, w);
  return P+v*t;
}

//点到直线距离
double DistanceToLine(Point P, Point A, Point B) {
  Vector v1 = B - A, v2 = P - A;
  return fabs(Cross(v1, v2) / Length(v1)); //不取绝对值，得到的是有向距离
}

//点到线段的距离
double DistanceToSegmentS(Point P, Point A, Point B) {
  if(A == B) return Length(P-A);
  Vector v1 = B-A, v2 = P-A, v3 = P-B;
  if(dcmp(Dot(v1, v2)) < 0) return Length(v2);
  else if(dcmp(Dot(v1, v3)) > 0) return Length(v3);
  else return fabs(Cross(v1, v2)) / Length(v1);
}

//点在直线上的投影
Point GetLineProjection(Point P, Point A, Point B) {
  Vector v = B - A;
  return A+v*(Dot(v, P-A)/Dot(v, v));
}

//线段相交判定，交点不在一条线段的端点
bool SegmentProperIntersection(Point a1, Point a2, Point b1, Point b2) {
  double c1 = Cross(a2-a1, b1-a1), c2 = Cross(a2-a1, b2-a1);
  double c3 = Cross(b2-b1, a1-b1), c4 = Cross(b2-b1, a2-b1);
  return dcmp(c1)*dcmp(c2) < 0 && dcmp(c3)*dcmp(c4) < 0;
```

```
}

//判断点是否在点段上，不包含端点
bool OnSegment(Point P, Point a1, Point a2) {
  return dcmp(Cross(a1-P, a2-P) == 0 && dcmp((Dot(a1-P, a2-P)) < 0));
}

//计算凸多边形面积
double ConvexPolygonArea(Point *p, int n) {
  double area = 0;
  for(int i = 1; i < n-1; i++)
    area += Cross(p[i] - p[0], p[i+1] - p[0]);
  return area/2;
}

//计算多边形的有向面积
double PolygonArea(Point *p, int n) {
  double area = 0;
  for(int i = 1; i < n-1; i++)
    area += Cross(p[i] - p[0], p[i+1] - p[0]);
  return area/2;
}

/***********************************************************************
* Morley定理：三角形每个内角的三等分线，相交成的三角形是等边三角形。
* 欧拉定理：设平面图的定点数，边数和面数分别为V,E,F。则V+F-E = 2;
***********************************************************************/

struct Circle {
  Point c;
  double r;

  Circle(Point c, double r) : c(c), r(r) {}
  //通过圆心角确定圆上坐标
  Point point(double a) {
    return Point(c.x + cos(a)*r, c.y + sin(a)*r);
  }
};

struct Line {
  Point p;
  Vector v;
  double ang;
  Line() {}
  Line(Point p, Vector v) : p(p), v(v) {}
  bool operator < (const Line& L) const {
    return ang < L.ang;
  }
};

//直线和圆的交点，返回交点个数，结果存在sol中。
//该代码没有清空sol。
int getLineCircleIntersecion(Line L, Circle C, double& t1, double& t2, vector<Point>& sol) {
  double a = L.v.x, b = L.p.x - C.c.x, c = L.v.y, d = L.p.y - C.c.y;
  double e = a*a + c*c, f = 2*(a*b + c*d), g = b*b + d*d - C.r*C.r;
  double delta = f*f - 4*e*g;
  if(dcmp(delta) < 0) return 0; //相离
  if(dcmp(delta) == 0) {        //相切
    t1 = t2 = -f / (2*e);
    sol.push_back(C.point(t1));
    return 1;
  }
  //相交
  t1 = (-f - sqrt(delta)) / (2*e);
  sol.push_back(C.point(t1));
  t2 = (-f + sqrt(delta)) / (2*e);
  sol.push_back(C.point(t2));
  return 2;
}

//两圆相交
int getCircleCircleIntersection(Circle C1, Circle C2, vector<Point>& sol) {
  double d = Length(C1.c - C2.c);
  if(dcmp(d) == 0) {
    if(dcmp(C1.r - C2.r == 0)) return -1;    //两圆完全重合
    return 0;                                //同心圆，半径不一样
  }
  if(dcmp(C1.r + C2.r - d) < 0) return 0;
  if(dcmp(fabs(C1.r - C2.r) == 0)) return -1;

  double a = Angle(C2.c - C1.c);                //向量C1C2的极角
  double da = acos((C1.r*C1.r + d*d - C2.r*C2.r) / (2*C1.r*d));
  //C1C2到C1P1的角
  Point p1 = C1.point(a-da), p2 = C1.point(a+da);
  sol.push_back(p1);
  if(p1 == p2) return 1;
  sol.push_back(p2);
  return 2;
}

const double PI = acos(-1);
//过定点做圆的切线
//过点p做圆C的切线，返回切线个数。v[i]表示第i条切线
int getTangents(Point p, Circle C, Vector* v) {
  Vector u = C.c - p;
  double dist = Length(u);
  if(dist < C.r) return 0;
  else if(dcmp(dist - C.r) == 0) {
    v[0] = Rotate(u, PI/2);
    return 1;
  } else {
    double ang = asin(C.r / dist);
    v[0] = Rotate(u, -ang);
    v[1] = Rotate(u, +ang);
    return 2;
  }
}

//两圆的公切线
//返回切线的个数，-1表示有无数条公切线。
//a[i], b[i] 表示第i条切线在圆A，圆B上的切点
int getTangents(Circle A, Circle B, Point *a, Point *b) {
  int cnt = 0;
  if(A.r < B.r) {
    swap(A, B);
    swap(a, b);
  }
  int d2 = (A.c.x - B.c.x)*(A.c.x - B.c.x) + (A.c.y - B.c.y)*(A.c.y - B.c.y);
  int rdiff = A.r - B.r;
  int rsum = A.r + B.r;
  if(d2 < rdiff*rdiff) return 0;   //内含
  double base = atan2(B.c.y - A.c.y, B.c.x - A.c.x);
  if(d2 == 0 && A.r == B.r) return -1;   //无限多条切线
  if(d2 == rdiff*rdiff) {         //内切一条切线
    a[cnt] = A.point(base);
    b[cnt] = B.point(base);
    cnt++;
    return 1;
  }
  //有外共切线
  double ang = acos((A.r-B.r) / sqrt(d2));
  a[cnt] = A.point(base+ang);
  b[cnt] = B.point(base+ang);
  cnt++;
  a[cnt] = A.point(base-ang);
  b[cnt] = B.point(base-ang);
  cnt++;
  if(d2 == rsum*rsum) {  //一条公切线
    a[cnt] = A.point(base);
    b[cnt] = B.point(PI+base);
    cnt++;
  } else if(d2 > rsum*rsum) {   //两条公切线
    double ang = acos((A.r + B.r) / sqrt(d2));
    a[cnt] = A.point(base+ang);
    b[cnt] = B.point(PI+base+ang);
    cnt++;
    a[cnt] = A.point(base-ang);
    b[cnt] = B.point(PI+base-ang);
    cnt++;
  }
  return cnt;
}

typedef vector<Point> Polygon;

//点在多边形内的判定
int isPointInPolygon(Point p, Polygon poly) {
  int wn = 0;
  int n = poly.size();
  for(int i = 0; i < n; i++) {
    if(OnSegment(p, poly[i], poly[(i+1)%n])) return -1; //在边界上
    int k = dcmp(Cross(poly[(i+1)%n]-poly[i], p-poly[i]));
    int d1 = dcmp(poly[i].y - p.y);
    int d2 = dcmp(poly[(i+1)%n].y - p.y);
    if(k > 0 && d1 <= 0 && d2 > 0) wn++;
    if(k < 0 && d2 <= 0 && d1 > 0) wn++;
  }
  if(wn != 0) return 1;       //内部
  return 0;                   //外部
}

//凸包
/***************************************************************
* 输入点数组p， 个数为p， 输出点数组ch。 返回凸包顶点数
* 不希望凸包的边上有输入点，把两个<= 改成 <
* 高精度要求时建议用dcmp比较
* 输入点不能有重复点。函数执行完以后输入点的顺序被破坏
****************************************************************/
int ConvexHull(Point *p, int n, Point* ch) {
  sort(p, p+n);      //先比较x坐标，再比较y坐标
  int m = 0;
  for(int i = 0; i < n; i++) {
    while(m > 1 && Cross(ch[m-1] - ch[m-2], p[i]-ch[m-2]) <= 0) m--;
    ch[m++] = p[i];
  }
  int k = m;
  for(int i = n-2; i >= 0; i++) {
    while(m > k && Cross(ch[m-1] - ch[m-2], p[i]-ch[m-2]) <= 0) m--;
    ch[m++] = p[i];
  }
  if(n > 1) m--;
  return m;
}

//用有向直线A->B切割多边形poly， 返回“左侧”。 如果退化，可能会返回一个单点或者线段
//复杂度O(n2);
Polygon CutPolygon(Polygon poly, Point A, Point B) {
  Polygon newpoly;
  int n = poly.size();
  for(int i = 0; i < n; i++) {
    Point C = poly[i];
    Point D = poly[(i+1)%n];
    if(dcmp(Cross(B-A, C-A)) >= 0) newpoly.push_back(C);
    if(dcmp(Cross(B-A, C-D)) != 0) {
      Point ip = GetLineIntersection(A, B-A, C, D-C);
      if(OnSegment(ip, C, D)) newpoly.push_back(ip);
    }
  }
  return newpoly;
}

//半平面交

//点p再有向直线L的左边。（线上不算）
bool Onleft(Line L, Point p) {
  return Cross(L.v, p-L.p) > 0;
}

//两直线交点，假定交点唯一存在
Point GetIntersection(Line a, Line b) {
  Vector u = a.p - b.p;
  double t = Cross(b.v, u) / Cross(a.v, b.v);
  return a.p+a.v*t;
}

int HalfplaneIntersection(Line* L, int n, Point* poly) {
  sort(L, L+n);               //按极角排序

  int first, last;            //双端队列的第一个元素和最后一个元素
  Point *p = new Point[n];    //p[i]为q[i]和q[i+1]的交点
  Line *q = new Line[n];      //双端队列
  q[first = last = 0] = L[0]; //队列初始化为只有一个半平面L[0]
  for(int i = 0; i < n; i++) {
    while(first < last && !Onleft(L[i], p[last-1])) last--;
    while(first < last && !Onleft(L[i], p[first])) first++;
    q[++last] = L[i];
    if(fabs(Cross(q[last].v, q[last-1].v)) < EPS) {
      last--;
      if(Onleft(q[last], L[i].p)) q[last] = L[i];
    }
    if(first < last) p[last-1] = GetIntersection(q[last-1], q[last]);
  }
  while(first < last && !Onleft(q[first], p[last-1])) last--;
  //删除无用平面
  if(last-first <= 1) return 0;   //空集
  p[last] = GetIntersection(q[last], q[first]);

  //从deque复制到输出中
  int m = 0;
  for(int i = first; i <= last; i++) poly[m++] = p[i];
  return m;
}
```

4 Graph

4.1 Tree

4.1.1 Universe

```

/* find root(重心) */

void findroot(int u, int fa) {
  int i;
  size[u] = 1;
  f[u] = 0;
  for (i = last[u]; i; i = e[i][2]) {
    if (!vis[e[i][0]] && e[i][0] != fa) {
      findroot(e[i][0], u);
      size[u] += size[e[i][0]];
      if (f[u] < size[e[i][0]])
        f[u] = size[e[i][0]];
    }
  }
  if (f[u] < ALL - size[u])
    f[u] = ALL - size[u];
  if (f[u] < f[root]) root = u;
}

/* --- da --- */

int dep[MAXN+1];
int ancestor[MAXN+1][MAXLGN];
int minw[MAXN+1][MAXLGN];

void dfs(int u, int fa) {
  ancestor[u][0] = fa;
  dep[u] = dep[fa] + 1;
  for(int e = u[front]; e; e = E[e].n) {
    int v = E[e].v, w = E[e].w;
    if(v != fa) {
      minw[v][0] = w;
      dfs(v, u);
    }
  }
}

void init_system(void) {
  int i = 0, w = 0;
  int t = 0;
  dep[0] = -1;
  dfs(1,0);
  for(w = 1; (t=(1<<w)) < N; ++w)
    for(i = 1; i <= N; ++i) if( dep[i] >= t ) {
        ancestor[i][w] = ancestor[ancestor[i][w-1]][w-1];
        minw[i][w] = min(minw[i][w-1], minw[ancestor[i][w-1]][w-1]);
      }
}

int query(int a, int b) {
  if(dep[a] < dep[b]) return query(b,a);
  else {  /* now dep[s] > dep[t] */
    int i = 0;
    int maxbit = MAXLGN-1;
    int ret = INF;
    //while((1<<maxbit) <= dep[a]) maxbit++;
    /* first up a to same dep with b. */
    for(i = maxbit; i >= 0; i--)
      if(dep[a] - (1<<i) >= dep[b]) {
        ret = min(ret, minw[a][i]);
        a = ancestor[a][i];
      }
    if(a == b) return ret;
    for(i = maxbit; i >= 0; i--)
      if(dep[a] - (1<<i) >= 0 && ancestor[a][i] != ancestor[b][i]) {
        ret = min(ret, min(minw[a][i], minw[b][i]));
        a = ancestor[a][i];
        b = ancestor[b][i];
      }
    ret = min(ret, min(minw[a][0], minw[b][0]));
    return ret;
  }
}
```

4.1.2 Point Divide and Conquer

Version 1

```
/* Tree::Point divide and conquer, by Abreto<m@abreto.net>. */
#include <bits/stdc++.h>

using namespace std;
typedef long long int ll;

#define MAXN    (100001)
#define MAXV    (MAXN+1)
#define MAXE    (MAXN<<1)
struct edge {
  int v;
  edge *n;
  edge(void):v(0),n(NULL) {}
  edge(int vv,edge *nn):v(vv),n(nn) {}
};
int nE;
edge E[MAXE];
edge *front[MAXV];
int label[MAXV];    /* 0 for '(', 1 for ')' */
void add_edge(int u, int v) {
  int ne = ++nE;
  E[ne] = edge(v, u[front]);
  u[front] = &(E[ne]);
}

int n;
ll ans;

char del[MAXV];
namespace findroot {
int ALL;
int nfind;
int vis[MAXV];
int size[MAXV];
int f[MAXV];
int root;
void __find(int u, int fa) {
  vis[u] = nfind;
  size[u] = 1;
  f[u] = 0;
  for(edge *e=u[front]; e; e = e->n) {
    int v = e->v;
    if((!del[v]) && (vis[v] != nfind) && (v != fa)) {
      __find(v, u);
      size[u] += size[v];
      if(f[u] < size[v])  f[u] = size[v];
    }
  }
  if(f[u] < ALL - size[u])  f[u] = ALL - size[u];
  if(f[u] < f[root])  root = u;
}
int find(int u, int all) {
  ++nfind;
  ALL = all;
  f[root = 0] = MAXV;
  __find(u,0);
  return root;
}
}

namespace workspaces {
int maxdep;
int dep[MAXV];
ll cntin[MAXV], cntout[MAXV];
int in[2][MAXV];    /* 0 for '(', 1 for ')' */
int out[2][MAXV];
void getdeep(int u, int fa) {
  dep[u] = dep[fa] + 1;
  if(dep[u] > maxdep) maxdep = dep[u];
  for(edge *e = u[front]; e; e = e->n)
    if((!del[e->v]) && (fa != e->v))
      getdeep(e->v, u);
}
void dfs(int u, int fa) {
  {
    /* out from root */
    out[0][u] = out[0][fa];
    out[1][u] = out[1][fa];
    if(0 == label[u]) { /* meet '(' */
      out[0][u]++;
    } else {    /* meet ')' */
      if(out[0][u]) out[0][u]--;
      else out[1][u]++;
    }
    if(out[0][u] == 0)
      cntout[out[1][u]]++;
  }
  {
    /* in to root */
    in[0][u] = in[0][fa];
    in[1][u] = in[1][fa];
    if(0 == label[u]) { /* meet '(' */
      if(in[1][u]) in[1][u]--;
      else in[0][u]++;
    } else {    /* meet ')' */
      in[1][u]++;
    }
    if(0 == in[1][u])
      cntin[in[0][u]]++;
  }
  /* do something */
  for(edge *e = u[front]; e; e = e->n) {
    int v = e->v;
    if((!del[v]) && (v != fa)) {
      dfs(v, u);
    }
  }
}
inline void init_maxdep(void) {
  maxdep = 0;
}
inline void update_maxdep(int u) {
  dep[u] = 1;
  if(dep[u] > maxdep) maxdep = dep[u];
  for(edge *e = u[front]; e; e = e->n)
    if((!del[e->v]))
      getdeep(e->v, u);
}
inline void clear(void) {
  for(int i = 0; i <= maxdep+1; ++i)
    cntin[i] = cntout[i] = 0;
}
inline void work(int u) {
  in[0][u] = in[1][u] = out[0][u] = out[1][u] = 0;
  in[label[u]][u] = out[label[u]][u] = 1;
  if(out[0][u] == 0) cntout[out[1][u]]++;
  if(0 == in[1][u]) cntin[in[0][u]]++;
  /* update in and out if neccessary */
  for(edge *e = u[front]; e; e = e->n)
    if(!(del[e->v]))
      dfs(e->v, u);
}
};

ll count(int u, int p) {
  ll ret = 0;
  workspace::init_maxdep();
  workspace::update_maxdep(u);
  workspace::clear();
  if(-1 == p) {
    for(edge *e = u[front]; e; e = e->n)
      if((!(del[e->v])))
        workspace::work(e->v);
    p = label[u];
    /* single end */
    if(0 == p) ret = workspace::cntout[1];
    else ret = workspace::cntin[1];
  } else {
    workspace::work(u);
  }
  if(0 == p) { /* p is ’(’ */
    for(int i = 0; i < workspace::maxdep; ++i)  /* concatenation */
      ret += workspace::cntin[i] * workspace::cntout[i+1];
  } else {    /* p is ’)’ */
    for(int i = 0; i < workspace::maxdep; ++i)  /* concatenation */
      ret += workspace::cntin[i+1] * workspace::cntout[i];
  }
  return ret;
}

void handle(int u) {
  del[u] = 1; /* delete current root. */
  ans += count(u, -1);
  /* do something */
  for(edge *e = u[front]; e; e = e->n) {
    int v = e->v;
    if(!del[v]) {
      ans -= count(v, label[u]);
      /* do something */
      int r = findroot::find(v, findroot::size[v]);
      handle(r);
    }
  }
}

void proc(void) {
  int r = findroot::find(1,n);
  handle(r);
}

char ls[MAXV+1];
int main(void) {
  int i = 0;
  scanf(”%d”, &n);
  scanf(”%s”, ls);
  for(i = 0; i < n; ++i)
    label[i+1] = ls[i] – ’(’;
  for(i = 1; i < n; ++i) {
    int ai, bi;
    scanf(”%d␣%d”, &ai, &bi);
    add_edge(ai, bi);
    add_edge(bi, ai);
  }
  proc();
  printf(”%lld\n”, ans);
  return 0;
}
```

Version 2

```
/* 2016 ACM/ICPC Asia Regional Dalian. Problem , by Abreto<m@abreto.net>. */
#include <bits/stdc++.h>

using namespace std;
typedef long long int ll;

/* offset in [1,k] */
#define GET(i,offset)   (((i)>>((offset)–1))&1)
#define SET(i,offset)   ((i)|(1<<((offset)–1)))
#define REV(i,offset)   ((i)^(1<<((offset)–1)))

#define MAXN    (50005)
#define MAXV    (MAXN+1)
#define MAXE    (MAXN<<1)
struct edge {
  int v;
  edge *n;
  edge(void):v(0),n(NULL) {}
  edge(int vv,edge *nn):v(vv),n(nn) {}
};
int nE;
edge E[MAXE];
edge *front[MAXV];
int label[MAXV];    /* each kind */
void add_edge(int u, int v) {
  int ne = ++nE;
  E[ne] = edge(v, u[front]);
  u[front] = &(E[ne]);
}

int n, k;
ll ans;
int all_kind;

int ndel;
int del[MAXV];
namespace findroot {
int ALL;
ll nfind;
ll vis[MAXV];
int size[MAXV];
int f[MAXV];
int root;
void __find(int u, int fa) {
  vis[u] = nfind;
  size[u] = 1;
  f[u] = 0;
  for(edge *e=u[front]; e; e = e->n) {
    int v = e->v;
    if((del[v] != ndel) && (vis[v] != nfind) && (v != fa)) {
      __find(v, u);
      size[u] += size[v];
      if(f[u] < size[v])  f[u] = size[v];
    }
  }
  if(f[u] < ALL - size[u])  f[u] = ALL - size[u];
  if(f[u] < f[root])  root = u;
}
int find(int u, int all) {
  ++nfind;
  ALL = all;
  f[root = 0] = MAXV;
  __find(u,0);
  return root;
}
}

namespace workspace {
ll cnt[1024];
int dp[MAXV];
void dfs(int u, int fa) {
  dp[u] = dp[fa] | label[u];
  cnt[dp[u]] ++;
  /* dig into children */
  for(edge *e = u[front]; e; e = e->n) {
    int v = e->v;
    if((del[v] != ndel) && (v != fa)) {
      dfs(v, u);
    }
  }
}
inline void clear(void) {
  for(int i = 1; i <= all_kind; ++i)
    cnt[i] = 0;
}
inline void work(int u) {
  dp[u] = label[u];
  cnt[dp[u]] ++;
  for(edge *e = u[front]; e; e = e->n)
    if((del[e->v] != ndel))
      dfs(e->v, u);
}
inline void show(void) {
  for(int i = 0; i <= all_kind; ++i)
    printf(”cnt[%d]␣=␣%lld\n”, i, cnt[i]);
  for(int i = 1; i <= n; ++i)
    printf(”dp[%d]␣=␣%d\n”, i, dp[i]);
}
```

```
};


ll count(int u, int p) {
  ll ret = 0;
  workspace::clear();
  //printf(”%d,%d  :\n”, u, p);
  if(-1 == p) {
    for(edge *e = u[front]; e; e = e->n)
      if(((del[e->v]) != ndel))
        workspace::work(e->v);
    p = label[u];
    /* single end */
    for(int i = 1; i <= all_kind; i++)
      if(all_kind == (i|p))
        ret += (workspace::cnt[i]<<1);
  } else {
    workspace::work(u);
  }
  //workspace::show();
  for(int i = 1; i <= all_kind; ++i)
    if( workspace::cnt[i] > 0 )
      for(int j = 1; j <= all_kind; ++j)
        if(all_kind == (i|p|j))
          ret += workspace::cnt[i] * workspace::cnt[j];
  //printf(”%lld\n”, ret);
  return ret;
}

void handle(int u) {
  //printf(”proccessing %d\n”, u);
  del[u] = ndel; /* delete current root. */
  ans += count(u, -1);
  /* do something */
  for(edge *e = u[front]; e; e = e->n) {
    int v = e->v;
    if(del[v] != ndel) {
      ans -= count(v, label[u]);
      /* do something */
      int r = findroot::find(v, findroot::size[v]);
      handle(r);
    }
  }
}

void proc(void) {
  int r = findroot::find(1,n);
  handle(r);
}

void clear(void) {
  int i;
  ans = 0;
  nE = 0;
  for(i = 0; i <= n; ++i) {
    front[i] = NULL;
  }
  //findroot::nfind = 0;
  ndel++;
}

void mozhu(void) {
  int i = 0;
  int li;
  for(i = 1; i <= n; ++i) {
    scanf(”%d”, &li);
    label[i] = 1<<(li-1);
  }
  for(i = 1; i < n; ++i) {
    int ai, bi;
    scanf(”%d␣%d”, &ai, &bi);
    add_edge(ai, bi);
    add_edge(bi, ai);
  }
  all_kind = (1<<k)-1;
  proc();
  if(1 == k) ans += n;
  printf(”%lld\n”, ans);
}

int main(void) {
  while( EOF != scanf(”%d%d”, &n, &k) ) {
    clear();
    mozhu();
  }
  return 0;
}
```

4.1.3 Hevay chain decompostion

```
/* bzoj 1036 */
/* 树链剖分 */
#include <bits/stdc++.h>

using namespace std;

#define MAXN    30030
#define MAXM    (MAXN<<1)
struct edge {
  int v;
  edge *n;
  edge(void) {}
  edge(int vv, edge *nn):v(vv),n(nn) {}
};
typedef edge *ep;
int nE;
edge E[MAXM];
ep front[MAXN];
void add_edge(int u, int v) {
  int ne = ++nE;
  E[ne] = edge(v, u[front]);
  u[front] = &(E[ne]);
}

int n;
int fa[MAXN], son[MAXN], sz[MAXN], dep[MAXN];
int top[MAXN];
int id[MAXN];
int tot;

void calc(int u, int uf) {
  dep[u] = dep[uf] + 1;
  fa[u] = uf;
  sz[u] = 1;
  son[u] = -1;
  for(ep e = u[front]; e; e = e->n) {
    if(e->v != uf) {
      calc(e->v, u);
      sz[u] += sz[e->v];
      if( -1 == son[u] || sz[son[u]] < sz[e->v] )
        son[u] = e->v;
    }
  }
}
void link(int u, int f) {
  id[u] = (++tot);
  top[u] = f;
  if(son[u] > 0) {
    link(son[u], f);
  }
  for(ep e = u[front]; e; e = e->n) {
    if(e->v != fa[u] && e->v != son[u]) {
      link(e->v, e->v);
    }
  }
}

/* 其实是树链剖分 */
void make_link_cut_tree(void) {
  calc(1, 0);
  link(1, 1);
}

int w[MAXN];
int sum[MAXN<<2], mx[MAXN<<2];

void maintain(int o, int l, int r) {
  sum[o] = sum[o<<1] + sum[o<<1|1];
  mx[o] = max(mx[o<<1], mx[o<<1|1]);
}
void build(int o = 1, int l = 1, int r = n) {
  if(r == l) {
    sum[o] = w[l];
    mx[o] = w[l];
  } else {
    int mid = l+r>>1;
    build(o<<1, l, mid);
    build(o<<1|1, mid+1, r);
    maintain(o, l, r);
  }
}
void update(int p, int x, int o = 1, int l = 1, int r = n) {
  if(p <= l && r <= p) {
    sum[o] = x;
    mx[o] = x;
  } else {
    int mid = l+r>>1;
    if(p <= mid) update(p,x,o<<1,l,mid);
    else update(p,x,o<<1|1,mid+1,r);
    maintain(o,l,r);
  }
}
int qs(int L, int R, int o = 1, int l = 1, int r = n) {
  if(R < l || r < L) return 0;
  else if (L <= l && r <= R) {
    return sum[o];
  } else {
    int mid = l+r>>1;
    return qs(L,R,o<<1,l,mid)+qs(L,R,o<<1|1,mid+1,r);
  }
}
```

```
int qm(int L, int R, int o = 1, int l = 1, int r = n) {
  if(L <= l && r <= R) {
    return mx[o];
  } else {
    int mid = l+r>>1;
    if(R <= mid) return qm(L, R, o<<1, l, mid);
    else if ( L > mid ) return qm(L, R, o<<1|1, mid+1, r);
    else return max(qm(L, R, o<<1, l, mid),qm(L, R, o<<1|1, mid+1, r));
  }
}

void change(int u, int t) {
  update(id[u], t);
}
int qmax(int u, int v) {
  int ret = -1000000000;
  while(top[u] != top[v]) {
    if( dep[top[u]] > dep[top[v]] ) {
      /* jump u */
      ret = max(ret, qm(id[top[u]], id[u]));
      u = fa[top[u]];
    } else {
      ret = max(ret, qm(id[top[v]], id[v]));
      v = fa[top[v]];
    }
  }
  ret = max(ret, qm(min(id[u],id[v]),max(id[u],id[v])));
  return ret;
}
int qsum(int u, int v) {
  int ret = 0;
  while(top[u] != top[v]) {
    if( dep[top[u]] > dep[top[v]] ) {
      /* jump u */
      ret += qs(id[top[u]], id[u]);
      u = fa[top[u]];
    } else {
      ret += qs(id[top[v]], id[v]);
      v = fa[top[v]];
    }
  }
  ret += qs(min(id[u],id[v]),max(id[u],id[v]));
  return ret;
}

int main(void) {
  int i;
  scanf(”%d”, &n);
  for(i = 1; i < n; ++i) {
    int a, b;
    scanf(”%d%d”, &a, &b);
    add_edge(a, b);
    add_edge(b, a);
  }
  make_link_cut_tree();
  for(i = 1; i <= n; ++i) {
    scanf(”%d”, &(w[id[i]]));
  }
  build();
  scanf(”%d”, &i);
  while(i--) {
    char command[8];
    int a, b;
    scanf(”%s␣%d␣%d”, command, &a, &b);
    if('C' == command[0]) change(a, b);
    else if ('M' == command[1]) printf("%d\n", qmax(a, b));
    else if ('S' == command[1]) printf("%d\n", qsum(a, b));
  }
  return 0;
}
```

4.2 2-SAT

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

using namespace std;

namespace two_sat {
const int maxn = 100000;
const int maxm = 1000000;
struct edge {
  int v;
  edge *n;
  edge(void):v(0),n(NULL) {}
  edge(int vv, edge *nn):v(vv),n(nn) {}
};
typedef edge *ep;
int n;
int nE;
edge E[maxm];
ep front[maxn];
void add_edge(int u, int v) {
  int ne = ++nE;
  E[ne] = edge(v, u[front]);
  u[front] = &(E[ne]);
}
/* (x = xval or y = yval), indexed from 0 */
void add_clause(int x, int xv, int y, int yv) {
  x = x*2 + xv;
  y = y*2 + yv;
  add_edge(x^1, y);
  add_edge(y^1, x);
}

char mark[maxn<<1];
int S[maxn<<1], c;
void init(int N) {
  n = N;
  for(int i = 0; i < n*2; ++i) {
    i[front] = NULL;
    i[mark] = 0;
  }
  nE = 0;
}

int dfs(int x) {
  if(mark[x^1]) return 0;
  if(mark[x]) return 1;
  mark[x] = 1;
  S[c++] = x;
  for(ep e = x[front]; e; e = e->n)
    if(!dfs(e->v)) return 0;
  return 1;
}

int solve(void) {
  for(int i = 0; i < n*2; i += 2)
    if(!mark[i] && !mark[i+1]) {
      c = 0;
      if(!dfs(i)) {
        while(c > 0) mark[S[--c]] = 0;
        if(!dfs(i+1)) return 0;
      }
    }
  return 1;
}
}
```

4.3 Cut Edge and Point

```
Finding cut edges
The code below works properly because the lemma above (first lemma):
  h[root] = 0
            par[v] = -1
                     dfs (v):
                     d[v] = h[v]
                            color[v] = gray
                                     for u in adj[v]:
                                         if color[u] == white
                                           then par[u] = v and dfs(u) and d[v] = min(d[v], d[u])
                                               if d[u] > h[v]
                                                 then the edge v-u is a cut edge
                                                 else if u != par[v])
          then d[v] = min(d[v], h[u])
                      color[v] = black
                                 In this code, h[v]▯ =▯ height of vertex v in the DFS tree and d[v]▯ =▯ min(h[w] where there is at least vertex u in subtree of v in the DFS tree where there is an edge between u and w).

                                     Finding cut vertices
                                     The code below works properly because the lemma above (first lemma):
                                     h[root] = 0
                                               par[v] = -1
                                                   dfs (v):
                                                   d[v] = h[v]
                                                       color[v] = gray
                                                 for u in adj[v]:
                                                           if color[u] == white
                                                             then par[u] = v and dfs(u) and d[v] = min(d[v], d[u])
                                                                   if d[u] >= h[v] and (v != root or number_of_children(v) > 1)
                                                                     then the edge v is a cut vertex
                                                                     else if u != par[v])
                didth d[v] = min(d[v], h[u])
                            color[v] = black
                                       In this code, h[v]▯ =▯ height of vertex v in the DFS tree and d[v]▯ =▯ min(h[w] where
```

there is at least vertex u in subtree of v in the DFS tree where there is an edge between u and w).

4.4 Euler Path

```
/* Euler path, by Abreto<m@abreto.net>. */
#define MAXV    (1024)
#define MAXE    (MAXV*MAXV)

typedef struct {
  int id;
  int nxt;
  int del;
} egde_t;
int front[MAXV];
egde_t edg[MAXE];
int d[MAXV];
int ind[MAXV], outd[MAXV];
int nedges;
void add_edge(int u, int v) {
  int newedge = ++nedges;
  edg[newedge].id = v;
  edg[newedge].nxt = u[front];
  edg[newedge].del = 0;
  u[front] = newedge;
  outd[u]++;
  ind[v]++;
  d[u]++;
  d[v]++;
}
void del_edge(int u, int v) {
  int e = 0;
  for(e=u[front]; e; e=edg[e].nxt)
    if(edg[e].id==v) {
      edg[e].del = 1;
      outd[u]--;
      ind[v]--;
      d[u]--;
      d[v]--;
      return;
    }
}

int path[MAXV];
int l;

void add2path(int u) {
  path[l++] = u;
}

/* Directed graph */
void euler(int x) {
  if(outd[x]) {
    int e = 0;
    for(e=x[front]; e; e=edg[e].nxt)
      if(!edg[e].del) {
        int v = edg[e].id;
        del_edge(x,v);
        euler(v);
      }
  }
  add2path(x);
}

/* Undirected graph */
void euler(int x) {
  if(d[x]) {
    int e = 0;
    for(e=x[front]; e; e=edg[e].nxt)
      if(!edg[e].del) {
        int v = edg[e].id;
        del_edge(x,v);
        del_edge(v,x);
        euler(v);
      }
  }
  add2path(x);
}
```

4.5 Shortest Path

4.5.1 Dijkstra

```
/* Shortest Path Dijstra, by Abreto<m@abreto.net>. */
#include <cstdio>
#include <set>
#include <utility>

using namespace std;
typedef set< pair<int,int> > spii;

#define MAXN    512
#define MAXV    (MAXN*MAXN)

struct egde_t {
  int id;
  int nxt;
};
int front[MAXV];
egde_t edg[MAXV<<3];
int nedges;
void add_edge(int u, int v) {
  int newedge = ++nedges;
  edg[newedge].id = v;
  edg[newedge].nxt = u[front];
  u[front] = newedge;
}

int d[MAXV];
int vis[MAXN];
int solid[MAXV];

int dijstra(int s, int t) {
  int v = s[front];
  spii q;
  q.insert(make_pair(0, s));
  while(!q.empty()) {
    auto it = q.begin();
    int u = it->second;
    int v = u[front];
    q.erase(it);
    solid[u] = 1;
    if(u == t) break;
    while(v) {
      int w = edg[v].id;
      if(!solid[w]) {
        if( (0==d[w]) || (d[u] + 1 < d[w]) ) {
          q.erase(make_pair(d[w],w));
          d[w] = d[u] + 1;
          q.insert(make_pair(d[w],w));
        }
      }
      v = edg[v].nxt;
    }
  }
  return d[t];
}
```

4.5.2 Shortest Path Fast Algorithm

```
/* Shortest Path Fast Algorithm, by Abreto<m@abreto.net>. */
#include <cstdio>
#include <cstring>
#include <queue>
#include <utility>

using namespace std;

#define MAXN    128

struct edge {
  int v;
  int w;
  int n;
};
edge edg[MAXN<<1];
int nedg;
int indegree[MAXN];
int front[MAXN];
int find_edge(int u, int v) {
  int e = u[front];
  while(e) {
    if(edg[e].v == v) return e;
    e = edg[e].n;
  }
  return 0;
}
void add_edge(int u, int v, int w) {
  int e = find_edge(u,v);
  if(0==e) {
    int newnode = ++nedg;
    edg[newnode].v = v;
    edg[newnode].w = w;
    edg[newnode].n = u[front];
    u[front] = newnode;
    indegree[v]++;
  } else {
    edg[e].w = (w < edg[e].w)?w:(edg[e].w);
  }
}

int n;

char inq[MAXN];
int vis[MAXN];
int d[MAXN];
int spfa(int s) { /* return 1 if fuhuan exists. */
```

```
  queue<int> q;
  memset(inq, 0, sizeof(inq));
  memset(d, -1, sizeof(d));
  memset(vis, 0, sizeof(vis));
  d[s] = 0;
  inq[s] = 1;
  q.push(s);
  while(!q.empty()) {
    int u = q.front();
    q.pop();
    printf(”proc␣%d..\n”, u);
    inq[u] = 0;
    if(vis[u]++ > n)
      return 1;
    for(int e = front[u]; e; e = edg[e].n) {
      int v = edg[e].v, w = edg[e].w;
      if( -1==d[v] || d[u] + w < d[v] ) {
        d[v] = d[u] + w;
        if(!inq[v]) {
          inq[v] = 1;
          q.push(v);
        }
      }
    }
  }
  return 0;
}
```

4.5.3 K-th shortest path

```
/**
 * poj
 * Problem#2449
 * Accepted
 * Time: 250ms
 * Memory: 9252k
 */
#include <iostream>
#include <fstream>
#include <sstream>
#include <algorithm>
#include <cstdio>
#include <cstdlib>
#include <cstring>
#include <ctime>
#include <cctype>
#include <cmath>
#include <vector>
#include <queue>
#include <stack>
#include <map>
#include <set>
#include <bitset>
using namespace std;
typedef bool boolean;

typedef class Edge {
public:
  int end;
  int next;
  int w;

  Edge(int end = 0, int next = -1, int w = 0):end(end), next(next), w(w) {        }
```

```
} Edge;

const int N = 1e3, M = 1e5;

typedef class MapManager {
public:
  int cnt;
  int h[N + 5];
  Edge edge[M + 5];

  MapManager() {        }
  MapManager(int n):cnt(-1) {
//            h = new int[(n + 1)];
//            edge = new Edge[(m + 1)];
    memset(h, -1, sizeof(int) * (n + 1));
  }

  inline void addEdge(int u, int v, int w) {
    edge[++cnt] = (Edge(v, h[u], w));
//            h[u] = (signed)edge.size() - 1;
    h[u] = cnt;
  }

  inline int start(int node) {
    return h[node];
  }

  Edge& operator [] (int pos) {
    return edge[pos];
  }
} MapManager;
#define m_endpos -1

int n, m;
MapManager g;
MapManager rg;
int s, t, k;
int ds[N + 5];

inline void init() {
  scanf(”%d%d”, &n, &m);
  memset(g.h, -1, sizeof(int) * (n + 1));
  memset(rg.h, -1, sizeof(int) * (n + 1));
  for(int i = 1, u, v, w; i <= m; i++) {
    scanf(”%d%d%d”, &u, &v, &w);
    g.addEdge(u, v, w);
    rg.addEdge(v, u, w);
  }
  scanf(”%d%d%d”, &s, &t, &k);
//    ds = new int[(n + 1)];
}

#define g rg
#define f ds
#define que que1
boolean vis[N + 5];
queue<int> que;
boolean spfa(int s, int t) {
  memset(f, 0x7f, sizeof(int) * (n + 1));
  memset(vis, false, sizeof(boolean) * (n + 1));
  que.push(s);
  f[s] = 0;
  while(!que.empty()) {
    int e = que.front();
    que.pop();
    vis[e] = false;
    for(int i = g.start(e); i != m_endpos; i = g[i].next) {
      int& eu = g[i].end;
//            cout << e << ” ” << eu << ” ” << i <<endl;
      if(f[e] + g[i].w < f[eu]) {
        f[eu] = f[e] + g[i].w;
        if(!vis[eu]) {
          que.push(eu);
          vis[eu] = true;
        }
      }
    }
  }
  return (f[t] != 0x7f7f7f7f);
}
#undef g
#undef f
#undef que

typedef class Status {
public:
  int node;
  int dis;
  int priority;

  Status(int node = 0, int dis = 0):node(node), dis(dis), priority(h()) {        }

  int h() {
    return dis + ds[node];
  }

  boolean operator < (Status b) const {
    return priority > b.priority;
  }
} Status;

int label[N + 5];
priority_queue<Status> que;
int bfs(int s, int t) {
  if(s == t)    k++;
//    label = new int[(n + 1)];
  memset(label, 0, sizeof(int) * (n + 1));
  que.push(Status(s, 0));
  while(!que.empty()) {
    Status e = que.top();
    que.pop();
    label[e.node]++;
    if(e.node == t && label[e.node] == k)
      return e.dis;
    for(int i = g.start(e.node); i != m_endpos; i = g[i].next) {
      if(label[g[i].end] < k)
        que.push(Status(g[i].end, e.dis + g[i].w));
    }
  }
  return −1;
}

inline void solve() {
  if(!spfa(t, s)) {
    puts(”−1”);
    return;
  }
  printf(”%d”, bfs(s, t));
```

```
}

int main() {
  init();
  solve();
  return 0;
}
```

4.6 Maxflow

```
/* Max Flow Problem, by Abreto<m@abreto.net> */

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

#define MAXV    (100000)
#define MAXE    (1000000)
struct edge {
  static int N;
  int v, w;
  edge *n;
  edge(void):v(0),w(0),n(NULL) {}
  edge(int vv, int ww, edge *nn):v(vv),w(ww),n(nn) {}
};
int nE;
edge E[MAXE];
edge *front[MAXV];
void add_edge(int u, int v, int w) {
  int ne = ++nE;
  E[ne] = edge(v, w, u[front]);
  u[front] = &(E[ne]);
}
edge *find_edge(int u, int v) {
  for(edge *e = u[front]; e != NULL; e = e->n)
    if(e->v == v)
      return e;
  return NULL;
}
void grant_e(int u, int v, int w) {
  edge *e = find_edge(u, v);
  if(NULL == e) add_edge(u,v,w);
  else e->w += w;
}

int vis[MAXV];
int path[MAXV];
int dfs(int u, int t) {
  vis[u] = 1;
  if(u == t) return 1;
  for(edge *e = u[front]; e != NULL; e = e->n) {
    int v = e->v;
    if(!vis[v] && e->w && dfs(v,t)) {
      path[u] = v;
      return 1;
    }
  }
  return 0;
}
int find_path(int s, int t) {
  memset(vis, 0, sizeof(vis));
  return dfs(s,t);
}
```

```
int max_flow(int s, int t) {
  int flow = 0;
  while(find_path(s,t)) {
    int i = 0;
    int minf = find_edge(s,path[s])->w;
    for(i = path[s]; i != t; i = path[i])
      minf = min(minf, find_edge(i,path[i])->w);
    for(i = s; i != t; i = path[i]) {
      grant_e(i, path[i], -minf);
      grant_e(path[i], i, minf);
    }
    flow += minf;
  }
  return flow;
}

/* Dinic */
#define N 1000
#define INF 100000000

struct Edge {
  int from,to,cap,flow;
  Edge(int u,int v,int c,int f):from(u),to(v),cap(c),flow(f) {}
};

struct Dinic {
  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;//边表，dges[e]和dges[e^1]互为反向弧
  vector<int>G[N];//邻接表，G[i][j]表示结点i的第j条边在e数组中的编号
  bool vis[N]; //BFS的使用
  int d[N]; //从起点到i的距离
  int cur[N]; //当前弧下标

  void addedge(int from,int to,int cap) {
    edges.push_back(Edge(from,to,cap,0));
    edges.push_back(Edge(to,from,0,0));
    int  m=edges.size();
    G[from].push_back(m-2);
    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<G[x].size(); i++) {
        Edge&e=edges[G[x][i]];
        if(!vis[e.to]&&e.cap>e.flow) { //只考虑残量网络中的弧
          vis[e.to]=1;
          d[e.to]=d[x]+1;
          Q.push(e.to);
        }
      }

    }
    return vis[t];
  }

  int dfs(int x,int a) { //x表示当前结点，a表示目前为止的最小残量
```

```
    if(x==t||a==0)return a;//a等于0时及时退出，此时相当于断路了
    int flow=0,f;
    for(int&i=cur[x]; i<G[x].size(); i++) { //从上次考虑的弧开始，注意要使用引用，同时修改cur[x]
      Edge&e=edges[G[x][i]];//e是一条边
      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)break;//a等于0及时退出，当a!=0,说明当前节点还存在另一个曾广路分支。

      }
    }
    return flow;
  }

  int Maxflow(int s,int t) { //主过程
    this->s=s,this->t=t;
    int flow=0;
    while(bfs()) { //不停地用bfs构造分层网络，然后用dfs沿着阻塞流增广
      memset(cur,0,sizeof(cur));
      flow+=dfs(s,INF);
    }
    return flow;
  }
};

/* ISAP */
struct Edge {
  int from,to,cap,flow;
};
const int maxn=650;
const int INF=0x3f3f3f3f;
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];
  void AddEdge(int from,int to,int cap) {
    edges.push_back((Edge) {
      from,to,cap,0
    });
    edges.push_back((Edge) {
      to,from,0,0
    });
    m=edges.size();
    G[from].push_back(m-2);
    G[to].push_back(m-1);
  }
  bool RevBFS() {
    memset(vis,0,sizeof(vis));
    queue<int>Q;
    Q.push(t);
    d[t]=0;
    vis[t]=1;
    while(!Q.empty()) {
      int x=Q.front();
      Q.pop();
      for(int i=0; i<G[x].size(); i++) {
        Edge &e =edges[G[x][i]^1];
        if(!vis[e.from]&&e.cap>e.flow) {
          vis[e.from]=1;
          d[e.from]=d[x]+1;
          Q.push(e.from);
        }
      }
    }
    return vis[s];
  }
  int Augment() {
    int x=t, a=INF;
    while(x!=s) {
      Edge &e = edges[p[x]];
      a= min(a,e.cap-e.flow);
      x=edges[p[x]].from;
    }
    x=t;
    while(x!=s) {
      edges[p[x]].flow+=a;
      edges[p[x]^1].flow-=a;
      x=edges[p[x]].from;
    }
    return a;
  }
  int Maxflow(int s,int t,int n) {
    this->s=s,this->t=t,this->n=n;
    int flow=0;
    RevBFS();
    memset(num,0,sizeof(num));
    for(int i=0; i<n; i++) {
      num[d[i]]++;
    }
    int x=s;
    memset(cur,0,sizeof(cur));
    while(d[s]<n) {
      if(x==t) {
        flow+=Augment();
        x=s;
      }
      int ok=0;
      for(int i=cur[x]; i<G[x].size(); i++) {
        Edge &e =edges[G[x][i]];
        if(e.cap>e.flow&&d[x]==d[e.to]+1) {
          ok=1;
          p[e.to]=G[x][i];
          cur[x]=i;
          x=e.to;
          break;
        }
      }
      if(!ok) {
        int m=n-1;
        for(int i=0; i<G[x].size(); i++) {
          Edge &e =edges[G[x][i]];
          if(e.cap>e.flow)
            m=min(m,d[e.to]);
        }
        if(--num[d[x]]==0)
          break;
        num[d[x]=m+1]++;
        cur[x]=0;
        if(x!=s)
          x=edges[p[x]].from;
      }
    }
    return flow;
  }
};
int main() {
  int n,m,a,b,c,res;
  while(scanf(”%d%d”,&m,&n)!=EOF) {
    ISAP tmp;
    for(int i=0; i<m; i++) {
      scanf(”%d%d%d”,&a,&b,&c);
      tmp.AddEdge(a,b,c);
    }
    res=tmp.Maxflow(1,n,n);
    printf(”%d\n”,res);
  }
  return 0;
}
```

4.7 Strongly Connected Component

```
/* Kosaraju */
#define MAXN    10010
#define MAXM    100010
struct edge {
  int v;
  edge *n;
  edge(void):v(0),n(NULL) {}
  edge(int vv, edge *nn):v(vv),n(nn) {}
};
int nE;
edge E[MAXM<<1];
edge *ori[MAXN];
edge *inv[MAXN];
void add_edge(edge *front[], int u, int v) {
  int ne = ++nE;
  E[ne] = edge(v, u[front]);
  u[front] = &(E[ne]);
}
void connect(int u, int v) {
  add_edge(ori, u, v);
  add_edge(inv, v, u);
}

int vis[MAXN];
int vst[MAXN];
void first_dfs(int u, int &sig) {
  vis[u] = 1;
  for(edge *e = u[ori]; e; e = e->n)
    if(!vis[e->v])
      first_dfs(e->v, sig);
  vst[++sig] = u;
}
int mark[MAXN];
void second_dfs(int u, int sig) {
  vis[u] = 1;
  mark[u] = sig;
  for(edge *e = u[inv]; e; e = e->n)
    if(!vis[e->v])
      second_dfs(e->v, sig);
}

```

```
int N, M;

int kosaraju(void) {
  int i;
  int sig = 0;
  for(i = 0; i <= N; ++i) vis[i] = 0;
  for(i = 1; i <= N; ++i) {
    if(!vis[i])
      first_dfs(i, sig);
  }
  sig = 1;
  for(i = 0; i <= N; ++i) vis[i] = 0;
  for(i = N; i > 0; --i) {
    if(!vis[vst[i]])
      second_dfs(vst[i], sig++);
  }
  for(i = 1; i <= N; ++i)
    if(mark[i] != 1)
      return 0;
  return 1;
}


void clear(void) {
  nE = 0;
  for(int i = 0; i <= N; ++i) {
    ori[i] = inv[i] = NULL;
  }
}

/* Tarjan */
#define MAXN    10010
#define MAXM    100010
struct edge {
  int v;
  edge *n;
  edge(void):v(0),n(NULL) {}
  edge(int vv, edge *nn):v(vv),n(nn) {}
};
typedef edge *ep;
int nE;
edge E[MAXM];
edge *front[MAXN];
void add_edge(int u, int v) {
  int ne = ++nE;
  E[ne] = edge(v, u[front]);
  u[front] = &(E[ne]);
}

int mark[MAXN];
int dfn[MAXN], low[MAXN];
int stk[MAXN];
int stk_top;

void tardfs(int u, int stamp, int &scc) {
  mark[u] = 1;
  dfn[u] = low[u] = stamp;
  stk[stk_top++] = u;
  for(ep e = u[front]; e; e = e->n) {
    if(0 == mark[e->v]) tardfs(e->v, ++stamp, scc);
    if(1 == mark[e->v]) low[u] = min(low[u], low[e->v]);
  }
  if(dfn[u] == low[u]) {
    ++scc;
    do {
      low[stk[stk_top-1]] = scc;
      mark[stk[stk_top-1]] = 2;
    } while(stk[(stk_top--)-1] != u);
  }
}

int tarjan(int n) {
  int scc = 0, lay = 1;
  for(int i = 1; i <= n; ++i)
    if(0 == mark[i])
      tardfs(i, lay, scc);
  return scc;
}

int N, M;

void clear(void) {
  nE = 0;
  for(int i = 0; i <= N; ++i) {
    i[front] = NULL;
    mark[i] = low[i] = 0;
  }
  stk_top = 0;
}

/* Garbow */
#define MAXN    10010
#define MAXM    100010

struct edge {
  int v;
  edge *n;
  edge(void):v(0),n(NULL) {}
  edge(int vv, edge *nn):v(vv),n(nn) {}
};
typedef edge *ep;

int nE;
edge E[MAXM];
edge *front[MAXN];
void add_edge(int u, int v) {
  int ne = ++nE;
  E[ne] = edge(v, u[front]);
  u[front] = &(E[ne]);
}

int stk1[MAXN], stk1t;
int stk2[MAXN], stk2t;
int low[MAXN], belg[MAXN];

void garbowdfs(int u, int lay, int &scc) {
  stk1[++stk1t] = u;
  stk2[++stk2t] = u;
  low[u] = ++lay;
  for(ep e=u[front]; e; e = e->n) {
    if(!low[e->v]) garbowdfs(e->v, lay, scc);
    else if (0 == belg[e->v])
      while(low[stk2[stk2t]] > low[e->v])
        --stk2t;
  }
  if(stk2[stk2t] == u) {
    stk2t--;
    scc++;
    do {
      belg[stk1[stk1t]] = scc;
    } while(stk1[stk1t--] != u);
  }
}

int grabow(int n) {
  int i;
  int scc = 0, lay = 0;
  for(i = 0; i <= n; ++i) {
    belg[i] = low[i] = 0;
  }
  for(i = 1; i <= n; ++i)
    if(0 == low[i])
      garbowdfs(i, lay, scc);
  return scc;
}

int N, M;

void clear(void) {
  nE = 0;
  for(int i = 0; i <= N; ++i) {
    front[i] = NULL;
  }
}
```

4.8 Perfect elimination ordering

求弦图的最大团数/最小色数的时候，只要在完美消除序列上从后往前贪心染色即可。而求最大独立集/最小团覆盖的时候，只要在完美消除序列上从前往后贪心取点即可。

```
/**
 * BZOJ 1006
 * [HNOI2008] 神奇的国度
 * 最大势法求完美消除序列
 * by Abreto<m@abreto.net>.
 **/
#include <cassert>
#include <cstdio>
#include <vector>
#include <bitset>
#include <algorithm>

using namespace std;
typedef vector<int> vi;
typedef vi::iterator vii;
#define pb push_back
#define MAXN 10100
#define MAXM 1000100

int n;
vi g[MAXN];
int ans;

struct node_t {
  int v;
  node_t *nxt;
} node[MAXM << 2];
int used;
node_t *new_node(void) {
  return node + (used ++);
}
```

```

node_t *f[MAXN];  /* head */
void lkto(int pos, int item) {
  node_t *t = new_node();
  t->v = item;
  t->nxt = f[pos];
  f[pos] = t;
}

int usedby[MAXN];
int color[MAXN];
bitset<MAXN> added;
int label[MAXN], max_label;
void mcs(void) {
  for (int i = 1; i <= n; i++) lkto(0, i);
  for (int i = n; i > 0; i--) {
    node_t *cur = f[max_label];
    assert(cur != NULL);
    while (added.test(cur->v)) {  /* already added */
      cur = cur->nxt;
      while (NULL == cur)
        cur = f[ --max_label ];
    }
    f[ max_label ] = cur->nxt;
    while (max_label && NULL == f[max_label]) max_label--;
    int u = cur->v;
    added.set(u);
    /* the i-th is u */
    for (vii it = g[u].begin(); it != g[u].end(); it++) {
      int v = *it;
      if (!added.test(v)) {
        label[v] ++;
        max_label = max(max_label, label[v]);
        lkto(label[v], v);
      }

      usedby[color[v]] = i;
    }
    for (int j = 1; j <= n; j++)
      if (usedby[j] != i) {
        color[u] = j;
        break;
      }
    ans = max(ans, color[u]);
  }
}

int main(void) {
  int m;
  scanf(”%d%d”, &n, &m);
  while (m--) {
    int ai, bi;
    scanf(”%d%d”, &ai, &bi);
    g[ai].pb(bi);
    g[bi].pb(ai);
  }
  mcs();
  printf(”%d\n”, ans);
  return 0;
}
```

5 Math

5.1 Euler Function

```
/* Euler function phi(x), by Abreto<m@abreto.net>. */

#define MAXX    3000000

int phi[MAXX];
void get_euler(void) {
  int i = 0, j = 0;
  phi[1] = 1;
  for(i = 2; i < MAXX; ++i)
    if(!phi[i])
      for(j = i; j < MAXX; j += i) {
        if(!phi[j]) phi[j] = j;
        phi[j] = phi[j]/i * (i-1);
      }
}
```

5.2 Möbius Function

```
void sieve() {
  fill(isPrime, isPrime + maxn, 1);
  mu[1] = 1, num = 0;
  for (int i = 2; i < maxn; ++i) {
    if (isPrime[i]) primes[num++] = i, mu[i] = -1;
    static int d;
    for (int j = 0; j < num && (d = i * primes[j]) < maxn; ++j) {
      isPrime[d] = false;
      if (i % primes[j] == 0) {
        mu[d] = 0;
        break;
      } else mu[d] = -mu[i];
    }
  }
}
```

5.3 Number Theory Inverse

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

const int n=10000000;      /*  */
const long long mod=1e9+7;  /*  prime required. */

long long fact[n],fiv[n],inv[n];

int main() {
  fact[0]=fact[1]=1;
  fiv[0]=fiv[1]=1;
  inv[1]=1;
  for (int i=2; i<n; i++) {
    fact[i]=fact[i-1]*i%mod;
    inv[i]=(mod-mod/i)*inv[mod%i]%mod;
    fiv[i]=inv[i]*fiv[i-1]%mod;
  }
  for (int i=1; i<n; i++) {
    if (fact[i]*fiv[i]%mod!=1)  printf(”fact␣wrong:␣%d\n”,i);
    if (inv[i]*i%mod!=1)        printf(”intv␣wrong:␣%d\n”,i);
  }
  cout<<”complete”<<endl;
  return 0;
}
```

5.4 Chinese Remainder Theorem

$$x \equiv a_i \pmod{m_i}$$

```
/* Chinese Remainder Theorem, by Abreto<m@abreto.net>. */
#include ”euler.c”

#define MAXN    64

typedef long long int ll;

ll quickpow(ll a, ll b, ll mod) {
  ll ret = 1, base = a;
  while(b > 0) {
    if(b & 1) ret = (ret * base) % mod;
    base = (base * base) % mod;
    b >>= 1;
  }
  return ret;
}

ll N;
ll a[MAXN], m[MAXN]; /* a and m is indexed from 0. */
ll x, M;

void naive_crt(void) {
  int i = 0;
  ll Mi[MAXN], nMi[MAXN];
  ll t[MAXN];

  M = 1;
  for(i = 0; i < N; ++i)
    M *= a[i];
  for(i = 0; i < N; ++i)
    Mi[i] = M / a[i];
  get_euler();
  for(i = 0; i < N; ++i)
    nMi[i] = quickpow(Mi[i], phi[a[i]]−1, a[i]);
  for(i = 0; i < N; ++i)
    t[i] = ((a[i] * Mi[i]) % M) * nMi[i] % M;
  for(i = 0; i < N; ++i)
    x = (x + t[i]) % M;
}
```

5.5 Linear congruences

```
#include <cstdio>
#include <cassert>
#include <cstdlib>

using namespace std;

class mod_equ_resolver {
  typedef long long int ll;
  ll a, m;
  inline void gurantee(void) {
    if ( a < 0 ) {
      ll k = ( -a ) / m;
      a += ( k + 1ll ) * m;
      a = ( a + m ) % m;
    } else {
      a %= m;
    }
    // printf("x = %lld (mod %lld)\n",a, m);
  }
public:
  mod_equ_resolver(void) {
    a = 0ll;
    m = 1ll;
  }
  ll exgcd(ll m, ll n, ll &x, ll &y) {
    if ( 0 == n ) {
      x = 1;
      y = 0;
      return m;
    }
    ll g = exgcd( n, m % n, x, y );
    ll t = x;
    x = y;
    y = t - m / n * y;
    return g;
  }
  int onemore(ll a2, ll m2) {
    ll x, y;
    ll g = exgcd( m, m2, x, y );
    assert(x*m+y*m2==g);
    a2 = (a2 + m2) % m2;
    if ( abs( a2 - a ) % g ) return -1;
    ll newm = m / g * m2;
    ll newa = a + ( a2 - a ) / g * x * m;
    a = newa;
    m = newm;
    gurantee();
    return 0;
  }
  ll resolve(void) {
    return a;
  }
};
```

Usage: For

$$\begin{cases} x \equiv a_1 \mod m_1 \\ x \equiv a_2 \mod m_2 \\ \vdots \\ x \equiv a_k \mod m_k \end{cases}$$

run

```
mod_equ_resolver solver;
for (int i = 1; i <= k; i++)
    solver.onemore(a[i], m[i]);
```

then the solution is

$$x \equiv solver.a \mod solver.m$$

5.6 FFT

```
#include <cmath>
using namespace std;
```

```
namespace fft {
#define eps (1e-9)
template < typename T = double >
struct dbl {
  T x;
  dbl(void):x(0.0) {}
  template <typename U>
  dbl(U a):x((T)a) {}
  inline char sgn(void) {
    return ((x>=-eps)&&(x<=eps))?(0):((x>eps)?(1):(-1));
  }
  inline T tabs(void) {
    return ((x>=-eps)&&(x<=eps))?(0.0):((x>eps)?(x):(-x));
  }
  inline dbl abs(void) {
    return dbl(tabs());
  }
  template <typename U> inline dbl &operator=(const U b) {
    x=(T)b;
    return (*this);
  }
  inline T *operator&(void) {
    return &x;
  }
  inline dbl operator-(void) const {
    return dbl(-x);
  }
  inline dbl operator+(const dbl &b) const {
    return dbl(x+b.x);
  }
  inline dbl operator-(const dbl &b) const {
    return dbl(x-b.x);
  }
  inline dbl operator*(const dbl &b) const {
    return dbl(x*b.x);
  }
  inline dbl operator/(const dbl &b) const {
    return dbl(x/b.x);
  }
  template <typename U> inline dbl operator^(const U &b) const {
    T ret=1.0,base=x;
    while(b) {
      if(b&1)ret*=base;
      base*=base;
      b>>=1;
    }
    return dbl(ret);
  }
  inline dbl operator+=(const dbl &b) {
    return dbl(x+=b.x);
  }
  inline dbl operator-=(const dbl &b) {
    return dbl(x-=b.x);
  }
  inline dbl operator*=(const dbl &b) {
    return dbl(x*=b.x);
  }
  inline dbl operator/=(const dbl &b) {
    return dbl(x/=b.x);
  }
  template <typename U> inline dbl operator^=(const U &b) {
    dbl tmp=(*this)^b;
    *this=tmp;
    return tmp;
  }
  inline bool operator==(const dbl &b) const {
    return (0 == ((*this)-b).sgn());
  }
  inline bool operator!=(const dbl &b) const {
    return (0 != ((*this)-b).sgn());
  }
  inline bool operator<(const dbl &b) const {
    return (-1 == ((*this)-b).sgn());
  }
  inline bool operator<=(const dbl &b) const {
    return (((*this)==b) || ((*this)<b));
  }
  inline bool operator>(const dbl &b) const {
    return (b < (*this));
  }
  inline bool operator>=(const dbl &b) const {
    return (((*this)==b) || ((*this)>b));
  }
  template <typename U> inline operator U() const {
    return (U)x;
  }
  inline char operator[](unsigned n) {
    if(n >= 0) {
      long long int ret=x;
      while(n--) {
        ret/=10;
      }
      return (ret%10);
    } else {
      T ret=x;
      n=-n;
      while(n--)ret*=10.0;
      return ((long long int)ret)%10;
    }
  }
};
template <typename T>
struct Complex {
  T x,y;  /* x + iy */
  Complex(void):x(T()),y(T()) {}
  Complex(T xx):x(xx) {}
  Complex(T xx,T yy):x(xx),y(yy) {}
  inline Complex operator-(void) const {
    return Complex(-x,-y);
  }
  inline Complex operator+(const Complex& b) const {
    return Complex(x+b.x,y+b.y);
  }
  inline Complex operator-(const Complex& b) const {
    return Complex(x-b.x,y-b.y);
  }
  inline Complex operator*(const Complex& b) const {
    return Complex(x*b.x-y*b.y,x*b.y+y*b.x);
  }
  inline Complex operator/(const Complex& b) const {
    T bo=b.x*b.x+b.y*b.y;
    return Complex((x*b.x+y*b.y)/bo,(y*b.x-x*b.y)/bo);
  }
  inline Complex& operator+=(const Complex& b) {
    Complex tmp=(*this)+b;
    (*this)=tmp;
    return (*this);
  }
  inline Complex& operator-=(const Complex& b) {
    Complex tmp=(*this)-b;
    (*this)=tmp;
    return (*this);
  }
  inline Complex& operator*=(const Complex& b) {
    Complex tmp=(*this)*b;
    (*this)=tmp;
    return (*this);
  }
  inline Complex& operator/=(const Complex& b) {
    Complex tmp=(*this)/b;
    (*this)=tmp;
    return (*this);
  }
  inline friend Complex operator+(const T& a, const Complex& b) {
    return Complex(a)+b;
  }
  inline friend Complex operator-(const T& a, const Complex& b) {
    return Complex(a)-b;
  }
  inline friend Complex operator*(const T& a, const Complex& b) {
    return Complex(a)*b;
  }
  inline friend Complex operator/(const T& a, const Complex& b) {
    return Complex(a)/b;
  }
};
typedef dbl<> Double;
typedef Complex<Double> ComplexD;
typedef long long int ll;
const int maxn = 2000000; /* !! */
const Double pi(acos(-1.0));

void build(ComplexD _P[], ComplexD P[], int n, int m, int curr, int &cnt) {
  if(m == n) {
    _P[curr] = P[cnt++];
  } else {
    build(_P, P, n, m*2, curr, cnt);
    build(_P, P, n, m*2, curr+m, cnt);
  }
}

void FFT(ComplexD P[], int n, int oper) { /* n should be 2^k. */
  static ComplexD _P[maxn];
  int cnt = 0;
  build(_P, P, n, 1, 0, cnt);
  copy(_P, _P+n, P);
  for(int d = 0; (1<<d)<n; ++d) {
    int m = 1<<d;
    int m2 = m*2;
    Double p0 = pi / m * oper;
    ComplexD unit_p0(cos(p0.x), sin(p0.x));
    for(int i = 0; i < n; i += m2) {
      ComplexD unit(1,0);
      for(int j = 0; j < m; ++j) {
        ComplexD &P1 = P[i+j+m], &P2 = P[i+j];
        ComplexD t = unit * P1;
        P1 = P2 - t;
        P2 = P2 + t;
        unit *= unit_p0;
      }
    }
  }
  if(-1 == oper) {
    for(int i = 0; i < n; ++i)
      P[i] /= Double(n);
  }
}
}
```

5.7 NTT

```
#include<bits/stdc++.h>
#define ll long long
const int N=262144;
const ll MOD=50000000001507329LL;//998244353 1004535809
using namespace std;
int n,m;
ll a[N],b[N],x[N],y[N];
ll wn[25];
ll Mul(ll x,ll y) { //乘法超ll用快速乘，主函数也需要用
  ll ans=(x*y-(ll)((long double)x/MOD*y+1e-8)*MOD);
  return ans<0?ans+MOD:ans;
}
ll Qpow(ll a,ll b,ll M) {
  ll ans=1;
  a%=M;
  while(b) {
    if(b&1) ans=Mul(ans,a);
    a=Mul(a,a);
    b>>=1;
  }
  return ans;
}
void Getwn() { //主函数预处理getwn()
  for(int i=0; i<25; i++) {
    wn[i]=Qpow(3,(MOD-1)/(1<<i),MOD);
  }
}
void NTT(ll *x,int n,int rev) {
  int i,j,k,ds;
  ll w,u,v;
  for(i=1,j=n>>1,k=n>>1; i<n-1; i++,k=n>>1) {
    if(i<j) swap(x[i],x[j]);
    while(j>=k) j-=k,k>>=1;
    if(j<k) j+=k;
  }
  for(i=2,ds=1; i<=n; i<<=1,ds++) {
    for(j=0; j<n; j+=i) {
      w=1;
      for(k=j; k<j+i/2; k++) {
        u=x[k];
        v=Mul(w,x[k+i/2]);
        x[k]=(u+v)%MOD;
        x[k+i/2]=(u-v+MOD)%MOD;
        w=Mul(w,wn[ds]);
      }
    }
  }
  if(rev==-1) {
    for(i=1; i<n/2; i++) swap(x[i],x[n-i]);
    w=Qpow(n,MOD-2,MOD);
    for(i=0; i<n; i++) x[i]=Mul(x[i],w);
  }
}
```

```
int main() {
  Getwn();
  while(~scanf(”%d%d”,&n,&m)) {
    for(int i=0; i<n; i++)scanf(”%lld”,&a[i]);
    for(int i=0; i<m; i++)scanf(”%lld”,&b[i]);
    int len=1,s=n+m;
    while(len<s)len<<=1;
    for(int i=n; i<len; i++)a[i]=0;
    for(int i=m; i<len; i++)b[i]=0;
    NTT(a,len,1);
    NTT(b,len,1);
    for(int i=0; i<len; i++)a[i]=Mul(a[i],b[i]);
    NTT(a,len,-1);
    for(int i=0; i<=s; i++)printf(”%lld␣”,a[i]);
    puts(””);
  }
}

// ------------------------------------------------------------------------------
#include<cstdio>
#include<iostream>
#include<cstring>
#include<cmath>
#include<complex>
using namespace std;
typedef long long LL;
const LL MOD=998244353,g=3,gi=332748118;
const LL N=1000005;
LL n,m;
LL a[N],b[N];
LL pow (LL x,LL y) {
  if (y==1) return x;
  LL lalal=pow(x,y>>1);
  lalal=lalal*lalal%MOD;
  if (y&1) lalal=lalal*x%MOD;
  return lalal;
}
void ntt (LL *a,LL n,LL o) {
  if (n==1) return ;
  LL k=(n>>1);
  LL w=1,wn=pow(o==1?g:gi,(MOD-1)/n),a0[k],a1[k];
  for (LL u=0; u<k; u++) {
    LL i=u*2;
    a0[u]=a[i];
    a1[u]=a[i+1];
  }
  ntt(a0,k,o);
  ntt(a1,k,o);
  for (LL u=0; u<k; u++) {
    a[u]=a0[u]+w*a1[u]%MOD;
    a[u]=(a[u]%MOD+MOD)%MOD;
    a[u+k]=a0[u]-w*a1[u];
    a[u+k]=(a[u+k]%MOD+MOD)%MOD;
    w=w*wn%MOD;
  }
}
void ntt(LL *a,LL n,LL op) {
  for (LL u=0; u<n; u++) bin[u]=(bin[u>>1]>>1)|((u&1)*(n>>1));
  for (LL u=0; u<n; u++) if (u<bin[u]) swap(a[u],a[bin[u]]);
  for (LL u=1; u<n; u<<=1) {
    LL wn=pow(op==1?g:gi,(MOD-1)/(u<<1)),w,t;
    for (LL i=0; i<n; i=i+(u<<1)) {
      w=1;
      for (LL k=0; k<u; k++) {
        t=w*a[u+i+k]%MOD;
        a[u+i+k]=(a[i+k]-t+MOD)%MOD;
        a[i+k]=(a[i+k]+t)%MOD;
        w=w*wn%MOD;
      }
    }
  }
  if(op==-1) {
    LL Inv=pow(n,MOD-2);
    for(LL i=0; i<n; i++) a[i]=a[i]*Inv%MOD;
  }
}

int main() {
  scanf(”%I64d%I64d”,&n,&m);
  for (LL u=0; u<=n; u++) scanf(”%I64d”,&a[u]);
  for (LL u=0; u<=m; u++) scanf(”%I64d”,&b[u]);
  m=m+n;
  n=1;
  while (n<=m) n<<=1;
  ntt(a,n,1);
  ntt(b,n,1);
  for (LL u=0; u<=n; u++) a[u]*=b[u];
  ntt(a,n,-1);
  LL inv=pow(n,MOD-2);
  for (LL u=0; u<=m; u++)   printf(”%I64d␣”,a[u]*inv%MOD);
  return 0;
}
```

5.8 Fast Walsh–Hadamard transform

- 异或

$$\mathcal{F}\{A\} = [\mathcal{F}\{A_0\} + \mathcal{F}\{A_1\}, \mathcal{F}\{A_0\} - \mathcal{F}\{A_1\}]$$

$$\mathcal{F}^{-1}\{A\} = \left[\mathcal{F}^{-1}\{\frac{A_0 + A_1}{2}\}, \mathcal{F}^{-1}\{\frac{A_0 - A_1}{2}\}\right]$$

- 按位与

$$\mathcal{F}\{A\} = [\mathcal{F}\{A_0\} + \mathcal{F}\{A_1\}, \mathcal{F}\{A_1\}]$$

$$\mathcal{F}^{-1}\{A\} = \left[\mathcal{F}^{-1}\{A_0\} - \mathcal{F}^{-1}\{A_1\}, \mathcal{F}^{-1}\{A_1\}\right]$$

- 按位或

$$\mathcal{F}\{A\} = [\mathcal{F}\{A_0\}, \mathcal{F}\{A_1\} + \mathcal{F}\{A_0\}]$$

$$\mathcal{F}^{-1}\{A\} = \left[\mathcal{F}^{-1}\{A_0\}, \mathcal{F}^{-1}\{A_1\} - \mathcal{F}^{-1}\{A_0\}\right]$$

```
void FWT(int a[],int n) {
  for(int d=1; d<n; d<<=1)
    for(int m=d<<1,i=0; i<n; i+=m)
      for(int j=0; j<d; j++) {
        int x=a[i+j],y=a[i+j+d];
        a[i+j]=(x+y)%mod,a[i+j+d]=(x-y+mod)%mod;
        //xor:a[i+j]=x+y,a[i+j+d]=(x-y+mod)%mod;
        //and:a[i+j]=x+y;
        //or:a[i+j+d]=x+y;
      }
}

void UFWT(int a[],int n) {
  for(int d=1; d<n; d<<=1)
    for(int m=d<<1,i=0; i<n; i+=m)
```

```
      for(int j=0; j<d; j++) {
        int x=a[i+j],y=a[i+j+d];
        a[i+j]=1LL*(x+y)*rev%mod,a[i+j+d]=(1LL*(x-y)*rev%mod+mod)%mod;
        //xor:a[i+j]=(x+y)/2,a[i+j+d]=(x-y)/2;
        //and:a[i+j]=x-y;
        //or:a[i+j+d]=y-x;
      }
}
void solve(int a[],int b[],int n) {
  FWT(a,n);
  FWT(b,n);
  for(int i=0; i<n; i++) a[i]=1LL*a[i]*b[i]%mod;
  UFWT(a,n);
}
```

5.9 Lucas

```
/* Lucas, by Abreto<m@abreto.net>. */

struct __lucas {
  static const int maxp = 100000;
  typedef long long int ll;
  int p;
  int f[maxp]; // fiv[maxp], inv[maxp];
  inline int mul(const int a, const int b) {
    ll z = 1ll * a * b;
    z -= z / p * p;
    return z;
  }
  int qow(int a, int x) {
    int ret = 1;
    while (x) {
      if (1 & x) ret = mul(ret, a);
      a = mul(a, a);
      x >>= 1;
    }
    return ret;
  }
  void init(int np) {
    p = np;
    // return; // uncomment this line if use binom()
    f[0] = f[1] = 1;
    // fiv[0] = fiv[1] = 1;
    // inv[1] = 1;
    for (int i = 2; i < p; i++) {
      f[i] = mul(f[i - 1], i);
      // inv[i] = mul(p - p / i, inv[p % i]);
      // fiv[i] = mul(fiv[i - 1], inv[i]);
    }
  }
  int C(int n, int k) {
    if (n < k) return 0;
    return mul(f[n], qow(mul(f[k], f[n - k]), p - 2));
  }
  /** use following if get TLE { */
  int binom(int n, int k) {
    if (n < k) return 0;
    if (k > n - k) k = n - k;
    int a = 1, b = 1;
    while (k) {
      a = mul(a, n);
      b = mul(b, k);
      n--;
      k--;
    }
    return mul(a, qow(b, p - 2));
  }
  /** } --- */
  int operator()(int n, int k) {
    if (0 == k) return 1;
    if (n < p && k < p) return C(n, k);
    return mul(C(n % p, k % p), (*this)(n / p, k / p));
  }
} lucas;
```

5.10 Linear Programming

```
/* 线性规划 */
#include<bits/stdc++.h>

using namespace std;
const int Maxn=110,Maxm=59;
class Simplex {
  /*
     功能:
     接受有n个约束, m个基本变量的方程组a[0~n][0~m]
     a[0][]存放需要最大化的目标函数, a[][0]存放常数
     Base[]存放基本变量的id,初始为1~m
     Rest[]存放松弛变量的id,初始为m+1~m+n
     返回此线性规划的最小值ans
     要求方案的话, Base[]中的变量值为0,Rest[]中的变量值为相应行的[0]
     如果solve
     返回1,说明运行正常ans是它的最大值
     返回0,说明无可行解
     返回-1,说明解没有最大值
     测试:
     m=2,n=3
     double a[4][3]={
     {0,1,3},
     {8,-1,1},
     {-3,1,1},
     {2,1,-4}
     };
     solve=1,ans=64/3;
     注意ac不了可能是eps的问题
   */
public:
  static const double Inf;
  static const double eps;
  int n,m;
  double a[Maxn][Maxm];
  int Base[Maxm],Rest[Maxn];
  double val[Maxm];
  double ans;
  void pt() {
    for(int i=0; i<=n; i++) {
      for(int j=0; j<=m; j++)printf(”%.2f␣”,a[i][j]);
      puts(””);
    }
  }
  void pivot(int x,int y) { //将第x个非基本变量和第y个基本变量调换
    swap(Rest[x],Base[y]);
    double tmp=-1./a[x][y];
    a[x][y]=-1.;
    for(int j=0; j<=m; j++)a[x][j]*=tmp;
    for(int i=0; i<=n; i++) {
      if(i==x||fabs(a[i][y])<eps)continue;
      tmp=a[i][y];
      a[i][y]=0;
      for(int j=0; j<=m; j++)a[i][j]+=tmp*a[x][j];
    }
  }
  bool opt() {
    while(1) {
      int csi=0;
      for(int i=1; i<=m; i++)if(a[0][i]>eps&&(!csi||Base[i]<Base[csi]))csi=i;
      if(!csi)break;
      int csj=0;
      double cur;
      for(int j=1; j<=n; j++) {
        if(a[j][csi]>-eps)continue;
        double tmp=-a[j][0]/a[j][csi];
        if(!csj||tmp+eps<cur||(fabs(tmp-cur)<eps&&Rest[j]<Rest[csj]))csj=j,cur=tmp;
      }
      if(!csj)return 0;
      pivot(csj,csi);
    }
    ans=a[0][0];
    return 1;
  }
  bool init() {
    ans=0;
    for(int i=1; i<=m; i++)Base[i]=i;
    for(int i=1; i<=n; i++)Rest[i]=m+i;
    int cs=1;
    for(int i=2; i<=n; i++)if(a[i][0]<a[cs][0])cs=i;
    if(a[cs][0]>=-eps)return 1;
    static double tmp[Maxm];
    for(int i=0; i<=m; i++)tmp[i]=a[0][i],a[0][i]=0;
    for(int i=1; i<=n; i++)a[i][m+1]=1.;
    a[0][m+1]=-1.;
    Base[m+1]=m+n+1;
    pivot(cs,++m);
    opt();
    m--;
    if(a[0][0]<-eps)return 0;
    cs=-1;
    for(int i=1; i<=n; i++) {
      if(Rest[i]>m+n) {
        cs=i;
        break;
      }
    }
    if(cs>=1) {
      int nxt=-1;
      m++;
      for(int i=1; i<=m; i++)if(a[cs][i]>eps||a[cs][i]<-eps) {
          nxt=i;
          break;
        }
      pivot(cs,nxt);
      m--;
    }
    for(int i=1; i<=m; i++) {
      if(Base[i]>m+n) {
        swap(Base[i],Base[m+1]);
        for(int j=0; j<=n; j++)a[j][i]=a[j][m+1];
        break;
      }
    }
    for(int i=1; i<=m; i++)a[0][i]=0;
    a[0][0]=tmp[0];
    for(int i=1; i<=m; i++)if(Base[i]<=m)a[0][i]=tmp[Base[i]];
    for(int i=1; i<=n; i++) {
      if(Rest[i]<=m) {
        for(int j=0; j<=m; j++)a[0][j]+=tmp[Rest[i]]*a[i][j];
      }
    }
    return 1;
  }
  void getval() {
    for(int i=1; i<=m; i++)val[i]=0;
    for(int i=1; i<=n; i++)if(Rest[i]<=m)val[Rest[i]]=a[i][0];
    //for(int i=1;i<=m;i++)printf(”%.2f ”,val[i]);puts(””);
  }
  int solve() {
    if(!init())return 0;
    if(!opt())return -1;
    getval();
    return 1;
  }
} solver;
const double Simplex:: Inf=1e80;
const double Simplex:: eps=1e-8;
int main() {
  int m,n,type;
  scanf(”%d%d%d”,&m,&n,&type);
  solver.a[0][0]=0;
  for(int i=1; i<=m; i++)scanf(”%lf”,&solver.a[0][i]);
  for(int i=1; i<=n; i++) {
    for(int j=1; j<=m+1; j++) {
      if(j==m+1)scanf(”%lf”,&solver.a[i][0]);
      else {
        scanf(”%lf”,&solver.a[i][j]);
        solver.a[i][j]=-solver.a[i][j];
      }
    }
  }
  solver.m=m,solver.n=n;
  int rep=solver.solve();
  if(rep==0)puts(”Infeasible”);
  else if(rep==-1)puts(”Unbounded”);
  else {
    printf(”%.12f\n”,solver.ans);
    if(type==1) {
      for(int i=1; i<=m; i++)printf(”%.12f%c”,solver.val[i],i==m?’\n’:’␣’);
    }
  }
}
```

5.11 Big Prime Test

```
#include <iostream>
#include <cstdlib>
using namespace std;
typedef long long LL;
LL minfactor, p[11] = {2, 3, 5, 7, 11, 13, 17, 19, 23, 29};
LL gcd(LL a, LL b) {
  return b ? gcd(b, a % b) : a;
}
```

```
LL qmult(LL a, LL b, LL mod) {  // 快速乘模
  LL sum = 0;
  while (b) {
    if (b & 1) {
      sum += a;
      if (sum >= mod) sum -= mod;  // 此处无需用%，%运算比减法慢很多
    }
    b >>= 1, a <<= 1;
    if (a >= mod) a -= mod;
  }
  return sum;
}
LL qpow(LL a, LL b, LL mod) {  // 快速幂模
  LL res = 1;
  while (b) {
    if (b & 1) res = qmult(res, a, mod);
    b >>= 1;
    a = qmult(a, a, mod);
  }
  return res;
}
bool prime_test(LL n, LL a) {  // 对整数n，底数a进行测试，返回true表示通过测试
  LL p = qpow(a, n - 1, n);
  if (p != 1) return false;
  else {  // 二次探测
    LL s = n - 1;
    while (!(s & 1) && p == 1) {
      s >>= 1;
      p = qpow(a, s, n);
    }
    if (p == 1 || p == n - 1) return true;
    else return false;
  }
}
bool Miller_Rabin(LL n) {  // 对整数n进行Miller_Rabin素数测试，返回true表示通过测试
  if (n <= 29) {    // if这一块其实可以不用
    for (int i = 0; i < 10; i++) {
      if (n == p[i]) return true;
    }
    return false;
  }
  for (int i = 0; i < 10; i++) {  // 利用前10个素数作为底数测试的正确率已经非常高
    if (gcd(n, p[i]) == 1 && !prime_test(n, p[i])) return false;
  }
  return true;
}
LL randf(LL x, LL n, LL c) {  // 满足要求的产生伪随机数函数
  return (qmult(x, x, n) + c) % n;
}
LL pollard_rho(LL n, LL c) {  // 查找n的因数，c为上面函数要用的随机数，c也可自己指定（但要有变化）
  LL x = rand() % n, y = x, i = 1, k = 2, p;  // 随机生成随机数的初始值，也可自己指定
  while (true) {
    i++;
    x = randf(x, n, c);
    p = gcd(y - x + n, n);
    if (p > 1 && p < n) return p;
    if (y == x) return n;   // 判圈，返回n表示查找失败，要更新随机种子重新查找
    if (i == k) {
      y = x;  // 更新范围和记录的数
      k <<= 1;
    }
  }
}
```

```
void find_factor(LL n) {  // 查找所有因数
  if (Miller_Rabin(n)) {
    minfactor = min(minfactor, n);
    return ;
  }
  LL p = n;
  while (p == n) p = pollard_rho(n, rand() % (n - 1) + 1);  // 查找失败则更新随机种子重新查找，直到找到因子
  find_factor(p);     // 递归查找更小因子
  find_factor(n / p);
}

int main() {
  int t;
  cin >> t;
  while (t--) {
    LL N;
    cin >> N;
    if (Miller_Rabin(N)) cout << ”Prime” << endl;
    else {
      minfactor = N;
      find_factor(N);
      cout << minfactor << endl;
    }
  }
  return 0;
}
```

5.11.1 Miller Rabin

```
/* Miller-Rabin Prime Test, by Abreto<m@abreto.net>. */

namespace miller_rabin {

typedef long long int ll;

inline ll add(const ll a, const ll b, const ll mod) {
  ll z = a + b;
  if (z >= mod) z -= mod;
  return z;
}
inline ll mul(ll a, ll b, const ll mod) {
  ll z = 0;
  if (a >= mod) a %= mod;
  if (b >= mod) b %= mod;
  while (b) {
    if (1 & b) z = add(z, a, mod);
    a = add(a, a, mod);
    b >>= 1;
  }
  return z;
}

ll qow(ll a, ll x, ll mod) {
  ll ret = 1ll;
  while (x) {
    if (1 & x) ret = mul(ret, a, mod);
    a = mul(a, a, mod);
    x >>= 1;
  }
  return ret;
}

const int K = 5;
const int p[] = {
  2, 3, 7, 61, 24251
};
const ll strong = 46856248255981ll;
/* 46 856 248 255 981 in (0, 1e16) */

bool mr(ll n, int k) {
  ll d = n - 1;
  int s = 0;
  while (d > 1 && 0 == (d & 1)) {
    s++;
    d >>= 1;
  }
  for (int i = 0; i < k; i++) {
    ll a = (i < K) ? p[i] : (1 + rand() % (n - 1));
    ll x = qow(a, d, n);
    for (int j = 0; j < s; j++) {
      ll xp = mul(x, x, n);
      if (1 == xp && x != 1 && x != n-1) return false;
      x = xp;
    }
    if (x != 1) return false;
  }
  return true;
}

/* 2,3,5,7,11,13 */
const int pre[] = {3, 5, 7, 11, 13};
bool test(ll n, int k = 5) {
  if (2 == n) return true;
  if (0 == (n & 1)) return false;
  if (strong == n) return false;
  for (int i = 0; i < 5; i++) {
    if (n == pre[i]) return true;
    if (n == n / pre[i] * pre[i])
      return false;
  }
  return mr(n, k);
}

}
```

5.11.2 Pollard's rho

```
/* Pollard's rho, by Abreto<m@abreto.net>. */

namespace pollards_rho {

typedef long long int ll;

inline ll add(const ll a, const ll b, ll mod) {
  ll z = a + b;
  if (z >= mod) z -= mod;
  return z;
}
inline ll mul(ll a, ll b, ll mod) {
  ll z = 0ll;
  if (a >= mod) a -= a / mod * mod;
  if (b >= mod) b -= b / mod * mod;
  while (b) {
    if (1 & b) z = add(z, a, mod);
    a = add(a, a, mod);
    b >>= 1;
  }
  return z;
}

ll gcd(ll m, ll n) {
  return (0 == n) ? m : gcd(n, m % n);
}

ll find(ll n, int c = -1) {
  ll x = rand() % n;
  ll y = x, k = 2;
  for (int i = 2; ; i++) {
    x = add(mul(x, x, n), (n + c) % n, n);
    ll d = gcd( y - x + n, n );  // change to abs(y - x) if get WA
    if (1 != d && n != d) return d;
    if (y == x) return n;
    if (i == k) {
      y = x;
      k <<= 1;
    }
  }
}

/** usage:
 * void find(ll n, int c = 107)
 * {
 *   if (1 == n) return;
 *   if ( miller-rabin(n) )
 *   {
 *     n is a prime;
 *     return;
 *   }
 *   ll p = n, k = c;
 *   while (p >= n) p = pollards_rho(p, k--);
 *   find(p, c);
 *   find(n/p, c);
 * }
 **/

}
```

5.12 Montgomery modular multiplication

```

/* -- Montgomery modular algorithm { -- */
struct Mod64 {
  typedef long long ll;
  typedef unsigned long long u64;
  typedef __int128_t i128;
  typedef __uint128_t u128;
  Mod64() :n_(0) {}
  Mod64(u64 n) :n_(init(n)) {}
  static u64 init(u64 w) {
    return reduce(u128(w) * r2);
  }
  static void set_mod(u64 m) {
    mod = m;
    assert(mod & 1);
    inv = m;
    for (int i = 0; i < 5; ++i) inv *= 2 - inv * m;
    r2 = -u128(m) % m;
  }
  static u64 reduce(u128 x) {
    u64 y = u64(x >> 64) - u64((u128(u64(x)*inv)*mod) >> 64);
    return ll(y)<0 ? y + mod : y;
  }
  Mod64& operator += (Mod64 rhs) {
    n_ += rhs.n_ - mod;
    if (ll(n_)<0) n_ += mod;
    return *this;
  }
  Mod64 operator + (Mod64 rhs) const {
    return Mod64(*this) += rhs;
  }
  Mod64& operator -= (Mod64 rhs) {
    n_ -= rhs.n_;
    if (ll(n_)<0) n_ += mod;
    return *this;
  }
  Mod64 operator - (Mod64 rhs) const {
    return Mod64(*this) -= rhs;
  }
  Mod64& operator *= (Mod64 rhs) {
    n_ = reduce(u128(n_)*rhs.n_);
    return *this;
  }
  Mod64 operator * (Mod64 rhs) const {
    return Mod64(*this) *= rhs;
  }
  u64 get() const {
    return reduce(n_);
  }
  static u64 mod, inv, r2;
  u64 n_;
};

Mod64::u64 Mod64::mod, Mod64::inv, Mod64::r2;
/* -- } Montgomery modular algorithm -- */

/**
 * usage:
 * First, Mod64::set_mod();
 * Mod64 a, b, c(init_val);
 * a = b * c;
 * printf(”%llu\n”, a.get());
 **/
```

5.13 Berlekamp Massey

```
/* Berlekamp Massey by HoldZhu. */
#include <cstdio>
#include <vector>

using namespace std;

namespace BerlekampMassey {
const int mod = 1e9 + 7;
int L, m, b, n;
vector<int> s, C, B;
void init() {
  s.clear();
  C.clear();
  B.clear();
  C.push_back(1);
  B.push_back(1);
  L = n = 0;
  m = b = 1;
}
int pow_mod(int a, int k) {
  int s = 1;
  while (k) {
    if (k & 1)
      s = 1ll * s * a % mod;
    a = 1ll * a * a % mod;
    k >>= 1;
  }
  return s;
}
void update(int d) {
  s.push_back(d);
  for (int i = 1; i <= L; ++i)
    d = (d + 1ll * C[i] * s[n - i] % mod) % mod;
  if (d == 0)
    ++m;
  else if (2 * L <= n) {
    vector<int> T = C;
    C.resize(n + 1 - L + 1);
    for (int i = L + 1; i <= n + 1 - L; ++i)
      C[i] = 0;
    for (int i = 0; i < B.size(); ++i)
      C[i + m] = (C[i + m] + mod - 1ll * d * pow_mod(b, mod - 2) % mod * B[i] % mod)
          % mod;
    L = n + 1 - L;
    B = T;
    b = d;
    m = 1;
  } else {
    for (int i = 0; i < B.size(); ++i)
      C[i + m] = (C[i + m] + mod - 1ll * d * pow_mod(b, mod - 2) % mod * B[i] % mod)
          % mod;
    ++m;
  }
  ++n;
}
void output() {
  printf(”F(n)=”);
  for (int i = 1; i < C.size(); ++i) {
    int output = (mod - C[i]) % mod;
    if (output > mod / 2)
      output -= mod;
    printf(”%s%d*F(n-%d)”, (output < 0 || i == 1) ? ”” : ”+”, output, i);
  }
  puts(””);
}
void output_code_for() {
  static const char *name = ”dp”;
  static const char *index = ”i”;
  static const char *upperbound = ”maxn”;
  puts(”//␣Generated␣by␣Berlekamp-Massey␣algorithm”);
  for (int i = 1; i < C.size(); ++i) {
    printf(”%s[%d]=%d;\n”, name, i - 1, s[i - 1]);
  }
  printf(”for(int␣i=%d;i<%s;++i)\n”, (int)C.size() - 1, upperbound);
  printf(”␣␣%s[%s]=((”, name, index);
  for (int i = 1; i < C.size(); ++i) {
    int output = (mod - C[i]) % mod;
    if (output > mod / 2)
```

```
      output -= mod;
    printf(”%s%d*%s[%s-%d]%%mod”, (output < 0 || i == 1) ? ”” : ”+”, output, name,
       index, i);
  }
  puts(”)%mod+mod)%mod;”);
}
void output_code_matrix() {
  // TODO
}
};

/** usage */
int usage() {
  // int arr[12] = {2, 24, 96, 416, 1536, 5504, 18944, 64000, 212992, 702464,
     2301952, 7512064};
  int arr[] = {3, 20, 119, 696, 4059, 23660, 137903, 803760, 4684659};
  BerlekampMassey::init();
  for (auto ai : arr) {
    BerlekampMassey::update(ai);
  }
  printf(”Formule:␣”);
  BerlekampMassey::output();
  printf(”Code:␣\n”);
  BerlekampMassey::output_code_for();
  return 0;
}
```

5.14 Inclusion—exclusion principle

5.14.1 General form

若 $A_1, A_2, \ldots, A_n$ 为有限集，则

$$\left|\bigcup_{i=1}^{n} A_i\right| = \sum_{\phi \neq J \subseteq \{1,2,\ldots,n\}} (-1)^{|J|-1} \left|\bigcap_{j \in J} A_j\right|$$

5.14.2 A generalization

若

$$g(A) = \sum_{S \subseteq A} f(S)$$

则

$$f(A) = \sum_{S \subseteq A} (-1)^{|A|-|S|} g(S)$$

更一般的，如果 S 是多重集合 (multiset)，那么

$$f(A) = \sum_{S \subseteq A} \mu(A - S) g(S)$$

其中

- 当 S 是含有偶数个元素的集合（没有重复元素）时，$\mu(S) = 1$
- 当 S 是含有奇数个元素的集合（没有重复元素）时，$\mu(S) = -1$
- 当 S 含有重复元素时，$\mu(S) = 0$.

5.14.3 Applications

1. 乱序排列

   如果集合 A 含有 n 个元素，则乱序排列的数目为 $[n!/e]$，$[x]$ 表示最接近 x 的整数.

5.15 Lindström–Gessel–Viennot lemma

对于一张无边权的 DAG 图，给定 n 个起点和对应的 n 个终点，这 n 条不相交路径的方案数为

$$\begin{vmatrix} e(a_1,b_1) & e(a_1,b_2) & \cdots & e(a_1,b_n) \\ e(a_2,b_1) & e(a_2,b_2) & \cdots & e(a_2,b_n) \\ \vdots & \vdots & \ddots & \vdots \\ e(a_n,b_1) & e(a_n,b_2) & \cdots & e(a_n,b_n) \end{vmatrix} \quad \text{(该矩阵的行列式)}$$

其中 $e(a,b)$ 为图上 a 到 b 的方案数.

6 String

6.1 Hash

```
/* Common hash for any substrings. */

typedef unsigned long long int llu;
#define MAXN 1000000
int n;
char s[MAXN];
llu H[MAXN], xP[MAXN], P = 99991ll;
void init(void) {
  int i = 0;
  xP[0] = 1ll;
  for(i = 1; i < MAXN; ++i) xP[i] = xP[i-1] * P;
  H[n] = 0;
  for(i = n-1; i >= 0; --i) H[i] = H[i+1]*P + s[i];
}
#define HASH(i,l)   (H[i] - H[i+l]*xP[l])
```

6.2 KMP

```
/* KMP, by Abreto<m@abreto.net>. */
#include <string.h>

/* !!NEED IMPROVING!! */

#define MAXL  (1000010)

char W[MAXL], T[MAXL];
int f[MAXL];
int lW, lT;

int count(void) {
  int cnt = 0;
  int i, j;
  lW = strlen(W);
  lT = strlen(T);
  // -- self-matching
  f[0] = j = -1;
  for(i = 1; i < lW; i++) {
    while( j >= 0 && W[j+1] != W[i] )
```

```
      j = f[j];
    if( W[j+1] == W[i] ) j++;
    f[i] = j;
  }
  // ---
  j = -1;
  for(i = 0; i < lT; i++) {
    while( j >= 0 && W[j+1] != T[i] )
      j = f[j];
    if( W[j+1] == T[i] ) j++;
    if( j == lW-1 ) {
      cnt++;
      j = f[j];
    }
  }
  return cnt;
}
```

6.3 exKMP

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

namespace exkmp {

const int maxn = 1000100, maxm = 1000100;

int n, m;
char S[maxn], T[maxm];

/* the length of longest prefix between T[i..m-1] and T[0..m-1] */
int nxt[maxm];
/* the length of longest prefix between S[i..n-1] and T[0..m-1] */
int ex[maxn];

void getsize() {
  n = strlen(S);
  m = strlen(T);
}

void self(void) {
  int q = 1, p = 0;
  nxt[0] = m;
  while (1 + p < m && T[1 + p] == T[p]) p++;
  nxt[1] = p;
  for (int i = 2; i < m; i++) {
    int l = nxt[i - q];
    if (i + l - 1 < p) {
      nxt[i] = l;
    } else {
      int j = max(0, p - i + 1);
      while (i + j < m && T[i + j] == T[j]) j++;
      nxt[i] = j;
      p = i + j - 1;
      q = i;
    }
  }
}

void run(void) {
  int q = 0, p = 0;
  self();
  while (p < n && p < m && S[p] == T[p]) p++;
  ex[0] = p;
  p--;
  for (int i = 1; i < n; i++) {
    int l = nxt[i - q];
    if (i + l - 1 < p) {
      ex[i] = l;
    } else {
      int j = max(0, p - i + 1);
      while (i + j < n && S[i + j] == T[j]) j++;
      ex[i] = j;
      p = i + j - 1;
      q = i;
    }
  }
}

void inspect(void) {
  printf(”S:␣”);
  for (int i = 0; i < n; i++) putchar(S[i]);
  puts(””);
  printf(”T:␣”);
  for (int i = 0; i < m; i++) putchar(T[i]);
  puts(””);
  printf(”next:”);
  for (int i = 0; i < m; i++) printf(”␣%d”, nxt[i]);
  puts(””);
  printf(”extend:”);
  for (int i = 0; i < n; i++) printf(”␣%d”, ex[i]);
  puts(””);
}

} // exkmp
```

6.4 Suffix Array

```
/* Suffix Array, copied. */

#define MAXN    (200010)
namespace mzry_sa {
int wx[MAXN],wy[MAXN],*x,*y,wss[MAXN],wv[MAXN];

bool dacmp(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]]=dacmp(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++);
}
}

/*
Suffix array O(n lg^2 n)
LCP table O(n)
*/
#include <cstdio>
#include <algorithm>
#include <cstring>

using namespace std;

#define REP(i, n) for (int i = 0; i < (int)(n); ++i)

namespace SuffixArray {
const int MAXN = 1 << 21;
char * S;
int N, gap;
int sa[MAXN], pos[MAXN], tmp[MAXN], lcp[MAXN];

bool sufCmp(int i, int j) {
  if (pos[i] != pos[j])
    return pos[i] < pos[j];
  i += gap;
  j += gap;
  return (i < N && j < N) ? pos[i] < pos[j] : i > j;
}

void buildSA() {
  N = strlen(S);
  REP(i, N) sa[i] = i, pos[i] = S[i];
  for (gap = 1;; gap <<= 1) {
    sort(sa, sa + N, sufCmp);
    REP(i, N - 1) tmp[i + 1] = tmp[i] + sufCmp(sa[i], sa[i + 1]);
    REP(i, N) pos[sa[i]] = tmp[i];
    if (tmp[N - 1] == N - 1) break;
  }
}

void buildLCP() {
  for (int i = 0, k = 0; i < N; ++i) if (pos[i] != N - 1) {
      for (int j = sa[pos[i] + 1]; S[i + k] == S[j + k];)
        ++k;
      lcp[pos[i]] = k;
      if (k)--k;
    }
}
} // end namespace SuffixArray

namespace HashSuffixArray {
const int
MAXN = 1 << 21;

typedef unsigned long long hash;

```

```
const hash BASE = 137;

int N;
char * S;
int sa[MAXN];
hash h[MAXN], hPow[MAXN];

#define getHash(lo, size) (h[lo] - h[(lo) + (size)] * hPow[size])

inline bool sufCmp(int i, int j) {
  int lo = 1, hi = min(N - i, N - j);
  while (lo <= hi) {
    int mid = (lo + hi) >> 1;
    if (getHash(i, mid) == getHash(j, mid))
      lo = mid + 1;
    else
      hi = mid - 1;
  }
  return S[i + hi] < S[j + hi];
}

void buildSA() {
  N = strlen(S);
  hPow[0] = 1;
  for (int i = 1; i <= N; ++i)
    hPow[i] = hPow[i - 1] * BASE;
  h[N] = 0;
  for (int i = N - 1; i >= 0; --i)
    h[i] = h[i + 1] * BASE + S[i], sa[i] = i;

  stable_sort(sa, sa + N, sufCmp);
}

} // end namespace HashSuffixArray

namespace lrj_sa {
const int MAXN = 1000;
char s[MAXN]; /* 原始字符数组（最后一个字符应必须是0，而前面的字符必须非0） */
int sa[MAXN], t[MAXN], t2[MAXN], c[MAXN], n;  /* n seems to be the length of s. */
/* every charactor is in [0,m-1] */
void build_sa(int m) {
  int i, *x = t, *y = t2;
  for(i = 0; i < m; ++i) c[i] = 0;
  for(i = 0; i < n; i++) c[x[i]=s[i]]++;
  for(i = 1; i < m; ++i) c[i] += c[i-1];
  for(i = n-1; i >= 0; --i) sa[--c[x[i]]] = i;
  for(int k = 1; k <= n; k <<= 1) {
    int p = 0;
    for(i = n-k; i < n; ++i) y[p++] = i;
    for(i = 0; i < n; ++i) if(sa[i] >= k) y[p++] = sa[i]-k;
    for(i = 0; i < m; i++) c[i] = 0;
    for(i = 0; i < n; i++) c[x[y[i]]]++;
    for(i = 0; i < m; ++i) c[i]+=c[i-1];
    for(i = n-1; i >= 0; --i) sa[--c[x[y[i]]]] = y[i];
    swap(x,y);
    p = 1;
    x[sa[0]] = 0;
    for(i = 1; i < n; ++i)
      x[sa[i]] = y[sa[i-1]]==y[sa[i]] && y[sa[i-1]+k]==y[sa[i]+k] ? p-1:p++;
    if(p >= n) break;
    m = p;
  }
}
int rank[MAXN], height[MAXN];
```

```
void get_height(void) {
  int i,j,k = 0;
  for(i = 0; i < n; ++i) rank[sa[i]] = i;
  for(i = 0; i < n; ++i) {
    if(k) k--;
    j = sa[rank[i]-1];
    while(s[i+k]==s[j+k]) k++;
    height[rank[i]] = k;
  }
}
} // end namespace lrj_sa
```

6.5 Aho-Corasick Automata

```
/* Aho-Corasick automaton algorithm, by Abreto<m@abreto.net>. */

#define MAXN  500500
#define NALPHA  26
#define FIRSTA  ’a’

/* pointer version => { */
struct vtx {
  vtx *nxt[NALPHA];
  vtx *fail;
  int end;
} vtxs[MAXN];
int nvtxs;
void myclr(void) {
  nvtxs = 0;
}
vtx *new_vtx(void) {
  vtx *ret = vtxs+(nvtxs++);
  for(int i = 0; i < NALPHA; i++)
    ret->nxt[i] = NULL;
  ret->fail = NULL;
  ret->end = 0;
  return ret;
}
void myins(vtx *root, char const *s) {
  for( ; *s ; s++ ) {
    int of = (*s) - FIRSTA;
    if ( NULL == root->nxt[of] ) {
      root->nxt[of] = new_vtx();
    }
    root = root->nxt[of];
  }
  root->end++;
}
void build_ac(vtx *root) {
  queue<vtx *> q;
  q.push(root);
  while(!q.empty()) {
    vtx *p = q.front();
    q.pop();
    for(int i = 0; i < NALPHA; i++) {
      if( NULL == p->nxt[i] ) continue;
      if( root == p ) p->nxt[i]->fail = root;
      else {
        vtx *t = p->fail;
        while ( t && NULL == t->nxt[i] ) {
          t = t->fail;
        }
        if (t) p->nxt[i]->fail = t->nxt[i];
        else p->nxt[i]->fail = root;
      }
      /* version[1] { */
      p->nxt[i]->end += p->nxt[i]->fail->end; /* update this sum, add its existing prefix to this. */
      /* } */
      q.push(p->nxt[i]);
    }
  }
}
int qry(vtx *root, char const *s) {
  vtx *p = root;
  int cnt = 0;
  for( ; *s ; s++ ) {
    int of = (*s) - FIRSTA;
    while( p != root && NULL == p->nxt[of] ) {
      p = p->fail;
    }
    if (p->nxt[of]) p = p->nxt[of];
    cnt += p->end;  // correct when version[1] exists.
    // if version[1] not exists, you need to add all ends from this vertex up.
    //for( vtx *t = p ; t ; t = t->fail )
    //  cnt += t->end;
  }
  return cnt;
}
/* } */

/* -- usage (of pointer version) -- */
#include <bits/stdc++.h>
using namespace std;

char S[1000100];
char pat[64];

int main(void) {
  int T, N;
  vtx *root = NULL;
  scanf(”%d”, &T);
  while(T--) {
    myclr();
    root = new_vtx();
    scanf(”%s”, S);
    scanf(”%d”, &N);
    while(N--) {
      scanf(”%s”, pat);
      myins(root, pat);
    }
    build_ac(root);
    printf(”%d\n”, qry(root, S));
  }
  return 0;
}
```

6.6 Manacher

```
char t[MAXL<<1];
int p[MAXL<<1];
int manacher(char *s) {
  int i;
  int sl = strlen(s);
  int pos = 0, mxr = 0;
  int ret = 0;
  t[0] = ’^’;
  for(i = 0; i < sl; ++i) {
    t[i*2+1] = ’#’;
    t[i*2+2] = s[i];
  }
  t[sl*2+1] = ’#’;
  t[sl*2+2] = ’$’;
  sl = sl*2+2;
  for(i = 1; i < sl; ++i) {
    if(i <= mxr) {
      p[i] = min(p[2*pos – i], mxr–i+1);
    } else {
      p[i] = 1;
    }
    while( t[i–p[i]] == t[i+p[i]] ) p[i]++;
    if( i + p[i] – 1 > mxr ) {
      mxr = i+p[i]–1;
      pos = i;
    }
    ret = max(ret, p[i]–1);
  }
  return ret;
}
```

7 Utility

7.1 IO plug-in

```
/* I/O Plug–in, by Abreto <m@abreto.net>. */
#include <stdio.h>

#if ( _WIN32 || __WIN32__ || _WIN64 || __WIN64__ )
#define INT64 ”%I64d”
#else
#define INT64 ”%lld”
#endif

#if ( _WIN32 || __WIN32__ || _WIN64 || __WIN64__ )
#define UNS64 ”%I64u”
#else
#define UNS64 ”%llu”
#endif

#define ISDIGIT(x) ((x>=’0’)&&(x<=’9’))
int readn(int *n) {
  int c=0;
  *n=0;
  for(; !ISDIGIT(c); c=getchar());
  for(; ISDIGIT(c); c=getchar())*n=(*n)*10+c–’0’;
  return (*n);
}
void putn(int n) {
  int ns[16]= {0,n%10},nd=1;
  while(n/=10)ns[++nd]=n%10;
  while(nd)putchar(ns[nd––]+’0’);
}
```

```
#include <cstdio>

class abio {
  static const unsigned BUF_SZ = 65536;
  FILE *istream, *ostream;
  char ibuf[BUF_SZ], obuf[BUF_SZ];
  bool reached_eof;
  size_t ip, isz;
  size_t op, osz;
  inline void clear_ibuf(void) {
    ip = isz = 0u;
  }
  inline void clear_obuf(void) {
    op = osz = 0u;
  }
  inline void clear_buffer(void) {
    reached_eof = false;
    clear_ibuf();
    clear_obuf();
  }
  inline size_t read_buffer(void) {
    isz = std::fread(ibuf, sizeof(char), BUF_SZ, istream);
    ip = 0;
    return isz;
  }
  inline size_t write_buffer(void) {
    if(osz) {
      size_t ret = std::fwrite(obuf+op, sizeof(char), osz-op, ostream);
      op += ret;
      if(op == osz) clear_obuf();
      return ret;
    }
    return 0;
  }
  inline abio &reach_eof(void) {
    reached_eof = true;
    return (*this);
  }
public:
  static const char endl = ’\n’;
  abio(FILE *input = stdin, FILE *output = stdout) {
    this->istream = input;
    this->ostream = output;
    clear_buffer();
  }
  abio(const char *input, const char *output) {
    this->istream = std::fopen(input, ”r”);
    this->istream = std::fopen(output, ”w+”);
    clear_buffer();
  }
  ~abio(void) {
    write_buffer();
    std::fclose(istream);
    std::fclose(ostream);
  }
  operator bool() const {
    return (!reached_eof);
  }
  inline int getchar(void) {
    if(isz == ip) read_buffer();
    if(isz == ip) return EOF;
    return ibuf[ip++];
  }
  inline int putchar(int ch) {
    if(osz == BUF_SZ) write_buffer();
    if(osz == BUF_SZ) return EOF;
    return (obuf[osz++] = ch);
  }
  abio &read_int(int &x) {
    int flag = 0, ch = getchar();
    for (; (EOF!=ch)&&((ch<’0’)||(ch>’9’)); ch=getchar()) if (’-’ == ch) flag = 1;
    if (EOF == ch) return (this->reach_eof());
    x = 0;
    for (; (ch>=’0’)&&(ch<=’9’); ch=getchar()) x = x * 10 + (ch - ’0’);
    if ( flag ) x *= (-1);
    return (*this);
  }
  abio &read_ll(long long int &x) {
    int flag = 0, ch = getchar();
    for (; (EOF!=ch)&&((ch<’0’)||(ch>’9’)); ch=getchar()) if (’-’ == ch) flag = 1;
    if (EOF == ch) return (this->reach_eof());
    x = 0ll;
    for (; (ch>=’0’)&&(ch<=’9’); ch=getchar()) x = x * 10ll + (ch - ’0’);
    if ( flag ) x *= (-1ll);
    return (*this);
  }
  abio &read_unsigned(unsigned &x) {
    int ch = getchar();
    for(; (EOF != ch) && ((ch < ’0’) || (ch > ’9’)); ch = getchar());
    if (EOF == ch) return (this->reach_eof());
    x = 0u;
    for(; (ch >= ’0’) && (ch <= ’9’); ch = getchar()) x = x * 10u + (ch - ’0’);
    return (*this);
  }
  abio &read_ull(unsigned long long int &x) {
    int ch = getchar();
    for(; (EOF != ch) && ((ch < ’0’) || (ch > ’9’)); ch = getchar());
    if (EOF == ch) return (this->reach_eof());
    x = 0ull;
    for(; (ch >= ’0’) && (ch <= ’9’); ch = getchar()) x = x * 10ull + (ch - ’0’);
    return (*this);
  }
  /* set interrupt as ’\n’ to read a whole line. */
  abio &read_s(char *s, const char interrupt = ’␣’) {
    int ch = getchar();
    while((EOF!=ch)&&(ch<’!’||ch>’~’))ch=getchar();
    if(EOF==ch) return (this->reach_eof());
    for(; (EOF!=ch)&&(interrupt!=ch)&&(ch>=’␣’&&ch<=’~’); ch=getchar())(*s++)=ch;
    (*s)=0;
    return (*this);
  }
  abio &write_int(int x, char append = 0) {
    int d[20],nd=0;
    if(0==x) putchar(’0’);
    if(x<0) {
      putchar(’-’);
      x=-x;
    }
    while(x) {
      d[nd++]=x%10;
      x/=10;
    }
    while(nd--)putchar(’0’+d[nd]);
    if(append)putchar(append);
    return (*this);
  }
  abio &write_ll(long long int x, char append = 0) {
    int d[20],nd=0;
    if(0==x) putchar(’0’);
    if(x<0) {
      putchar(’-’);
      x=-x;
    }
    while(x) {
      d[nd++]=x%10;
      x/=10;
    }
    while(nd--)putchar(’0’+d[nd]);
    if(append)putchar(append);
    return (*this);
  }
  abio &write_unsigned(unsigned x, char append = 0) {
    int d[20],nd=0;
    if(0==x) putchar(’0’);
    while(x) {
      d[nd++]=x%10;
      x/=10;
    }
    while(nd--)putchar(’0’+d[nd]);
    if(append)putchar(append);
    return (*this);
  }
  abio &write_ull(unsigned long long int x, char append = 0) {
    int d[20],nd=0;
    if(0==x) putchar(’0’);
    while(x) {
      d[nd++]=x%10;
      x/=10;
    }
    while(nd--)putchar(’0’+d[nd]);
    if(append)putchar(append);
    return (*this);
  }
  abio &write_s(const char *s, char append = 0) {
    while(*s) putchar(*s++);
    if(append) putchar(append);
    return (*this);
  }
  abio &operator>>(char &ch) {
    ch = getchar();
    if(EOF==ch) return (this->reach_eof());
    return (*this);
  }
  abio &operator>>(int &x) {
    return read_int(x);
  }
  abio &operator>>(long long int &x) {
    return read_ll(x);
  }
  abio &operator>>(unsigned &x) {
    return read_unsigned(x);
  }
  abio &operator>>(unsigned long long int &x) {
    return read_ull(x);
  }
  abio &operator>>(char *s) {
    return read_s(s);
  }
  abio &operator<<(const char ch) {
    putchar(ch);
    return (*this);
  }
  abio &operator<<(const int x) {
    return write_int(x);
  }
  abio &operator<<(const long long int x) {
    return write_ll(x);
  }
  abio &operator<<(const unsigned x) {
    return write_unsigned(x);
  }
  abio &operator<<(const unsigned long long int x) {
    return write_ull(x);
  }
  abio &operator<<(const char *s) {
    return write_s(s);
  }
} io;
```

7.2 Random Numbers

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

const int N = 3000000;

double average_distance(const vector<int> &permutation) {
  double distance_sum = 0;

  for (int i = 0; i < N; i++)
    distance_sum += abs(permutation[i] - i);

  return distance_sum / N;
}

int main() {
  /* use mt19937_64 if you want 64-bit random numbers */
  mt19937 rng(chrono::steady_clock::now().time_since_epoch().count());
  vector<int> permutation(N);

  for (int i = 0; i < N; i++)
    permutation[i] = i;

  shuffle(permutation.begin(), permutation.end(), rng);
  cout << average_distance(permutation) << '\n';

  for (int i = 0; i < N; i++)
    permutation[i] = i;

  for (int i = 1; i < N; i++)
    swap(permutation[i], permutation[uniform_int_distribution<int>(0, i)(rng)]);

  cout << average_distance(permutation) << '\n';
}

// -------------
// rand() * rand() % M
// or
// rand() << 16 | rand()
// maybe also work in some scene;
```

8 Appendix

8.1 C++ Reference

8.1.1 STL

8.2 Java Reference

8.2.1 Basic

Structure

```
import java.io.*;
import java.util.*;
import java.math.*;

public class Main {
  public static final int maxn = 50050;
  public static int[] int_array = new int[maxn]; /** Array */
  public static int[] another_arr = {1, 2, 3, 5};
  public static void main(String[] args) throws Exception {
    Scanner cin = new Scanner(System.in);
    int a = cin.nextInt(), b = cin.nextInt();
    System.out.println(a + b);
    for (int i = 0; i < 4; i++)
      System.out.println(another_arr[i]);
  }
}
```

Constant

```
    public static final int A = 0;
```

Array

```
    int [] a = new int[5];
    int [] b = {10, 35, 45, 89, 90};
```

Bit op

The Java programming language also provides operators that perform bitwise and bit shift operations on integral types. The operators discussed in this section are less commonly used. Therefore, their coverage is brief; the intent is to simply make you aware that these operators exist.

The unary bitwise complement operator "∼" inverts a bit pattern; it can be applied to any of the integral types, making every "0" a "1" and every "1" a "0". For example, a byte contains 8 bits; applying this operator to a value whose bit pattern is "00000000" would change its pattern to "11111111".

The signed left shift operator "<<" shifts a bit pattern to the left, and the signed right shift operator ">>" shifts a bit pattern to the right. The bit pattern is given by the left-hand operand, and the number of positions to shift by the right-hand operand. The unsigned right shift operator ">>>" shifts a zero into the leftmost position, while the leftmost position after ">>" depends on sign extension.

The bitwise & operator performs a bitwise AND operation.

The bitwiseˆ operator performs a bitwise exclusive OR operation.

The bitwise | operator performs a bitwise inclusive OR operation.

8.2.2 BigInteger

Immutable arbitrary-precision integers. All operations behave as if BigIntegers were represented in two's-complement notation (like Java's primitive integer types). BigInteger provides analogues to all of Java's primitive integer operators, and all relevant methods from java.lang.Math. Additionally, BigInteger provides operations for modular arithmetic, GCD calculation, primality testing, prime generation, bit manipulation, and a few other miscellaneous operations.

API

- ZERO, ONE, TEN

- Constructors

```
public BigInteger(String val, int radix)
```

- primes

```
/* prob of composite <= 2^(-100) */
public BigInteger nextProbablePrime()
/* return true if the probability that it is prime exceeds (1 - 1/(2^certainty)) */
public boolean isProbablePrime(int certainty)
```

- valueOf
- 算术运算

```
public BigInteger add(BigInteger val)
public BigInteger subtract(BigInteger val)
public BigInteger multiply(BigInteger val)
public BigInteger divide(BigInteger val)
public BigInteger[] divideAndRemainder(BigInteger val) // [quotient, remainder]
public BigInteger remainder(BigInteger val)
public BigInteger pow(int exponent)
public BigInteger gcd(BigInteger val)
public BigInteger abs()
public BigInteger negate()
public int signum() // -1, 0 or 1
public BigInteger mod(BigInteger m) // always returns a non-negative BigInteger.
public BigInteger modPow(BigInteger exponent, BigInteger m)
public BigInteger modInverse(BigInteger m)
```

- 位运算

```
public BigInteger shiftLeft(int n)
public BigInteger shiftRight(int n)
public BigInteger and(BigInteger val)
public BigInteger or(BigInteger val)
public BigInteger xor(BigInteger val)
public BigInteger not()
public boolean testBit(int n)
public BigInteger setBit(int n)
public BigInteger clearBit(int n)
public BigInteger flipBit(int n)
public int getLowestSetBit() // lowbit
public int bitLength()
public int bitCount()
```

- 比较

```
public int compareTo(BigInteger val) // -1, 0 or 1 if this () val
public BigInteger min(BigInteger val)
public BigInteger max(BigInteger val)
```

- transform

```
public int hashCode()
public String toString(int radix)
public int intValue()
public long longValue()
```

8.2.3 BigDecimal

Immutable, arbitrary-precision signed decimal numbers. A BigDecimal consists of an arbitrary precision integer unscaled value and a 32-bit integer scale. If zero or positive, the scale is the number of digits to the right of the decimal point. If negative, the unscaled value of the number is multiplied by ten to the power of the negation of the scale. The value of the number represented by the BigDecimal is therefore $(unscaledValue \times 10^{-scale})$.

API

- Constructors

```
public BigDecimal(String val)
public BigDecimal(double val)
public BigDecimal(BigInteger val)
public BigDecimal(int val)
public static BigDecimal valueOf(long unscaledVal, int scale) // u * 10^(-scale)
public static BigDecimal valueOf(double val)
```

- arithmetic operation

```
public BigDecimal add(BigDecimal augend)
public BigDecimal subtract(BigDecimal subtrahend)
public BigDecimal multiply(BigDecimal multiplicand)
public BigDecimal divide(BigDecimal divisor)
public BigDecimal divideToIntegralValue(BigDecimal divisor)
public BigDecimal remainder(BigDecimal divisor)
public BigDecimal[] divideAndRemainder(BigDecimal divisor)
public BigDecimal pow(int n)
public BigDecimal abs()
public BigDecimal negate()
public int signum()
public int scale()
public int precision()
```

- transform

```
// RoundingMode.
//    UP, DOWN, CEILING, FLOOR, HALF_UP, HALF_DOWN, HALF_EVEN
public MathContext(int setPrecision[, RoundingMode setRoundingMode])
public BigDecimal round(MathContext mc)
public int hashCode()
public String toString()
public String toPlainString()
public double doubleValue()
```

- comparison

```
public int compareTo(BigDecimal val)
public BigDecimal min(BigDecimal val)
public BigDecimal max(BigDecimal val)
```

8.2.4 Sorting

java.util.Arrays

```
public static void sort(int[] a[, int fromIndex, int toIndex]) // ascending numerical
    order.
// or parallelSort ?
```

8.3 Environment test items

- 一秒运算次数，带模一秒跑多少
- 行末空格
- `assert(0)` 是 WA 还是 RE
- `RAND_MAX` 大小