CodeLibrary

Wentai Zhang[1]

August 10, 2011

[1]Contact me: rchardx@gmail.com

Contents

Chapter 1

Date Structure

1.1 Splay - NOI2005 Sequence

Last update: 2011-08-06 16:44:00

```
const int MAXN = 500000;

inline int max(int a,int b,int c) {
  return max(a,max(b,c));
}

struct node {
  int ky,lm,rm,sm,sz,co;
  node *pa,*lc,*rc;
  bool rv,pt,sb;
  node(int _ky,node *_pa,node *_lc,node *_rc,bool _sb = 0)
    : ky(_ky),pa(_pa),lc(_lc),rc(_rc),sb(_sb) {
    lm = rm = sm = INT_MIN;
    rv = pt = 0;
    co = sz = 0;
  }
  node() {}
};

node *root,*head,*tail,*null;
int n,m,dat[MAXN];

void update(node *x) {
  node *lc = x->lc,*rc = x->rc;
  x->sz = lc->sz+rc->sz+1;
  if (x->sb) {
    x->co = lc->co+rc->co;
    x->lm = max(lc->lm,rc->lm);
    x->rm = max(rc->rm,lc->rm);
    x->sm = max(lc->sm,rc->sm);
```

```
  }
  else {
    x->co = lc->co+rc->co+x->ky;
    x->lm = max(lc->lm,lc->co+x->ky+max(0,rc->lm));
    x->rm = max(rc->rm,rc->co+x->ky+max(0,lc->rm));
    x->sm = max(lc->sm,rc->sm,max(0,lc->rm)+x->ky+max(0,rc->lm));
  }
}

void dispose(node *&x) {
  if (x==null) return;
  dispose(x->lc);
  dispose(x->rc);
  delete x;
  x = null;
}

void lfix(node *x) {
  node *y = x->pa;
  x->pa = y->pa;
  if (y->pa) {
    if (y==y->pa->lc) y->pa->lc = x;
    else y->pa->rc = x;
  }
  y->lc = x->rc;
  x->rc->pa = y;
  x->rc = y;
  y->pa = x;
  update(y);
  update(x);
}

void rfix(node *x) {
  node *y = x->pa;
  x->pa = y->pa;
  if (y->pa) {
    if (y==y->pa->lc) y->pa->lc = x;
    else y->pa->rc = x;
  }
  y->rc = x->lc;
  x->lc->pa = y;
  x->lc = y;
  y->pa = x;
  update(y);
  update(x);
}

void splay(node *x,node *&_root) {
  node *lim = _root->pa;
  while (x->pa!=lim) {
```

```
    node *y = x->pa;
    if (y->pa!=lim) {
      node *z = y->pa;
      if (y==z->lc)
        if (x==y->lc) lfix(y),lfix(x);
        else rfix(x),lfix(x);
      else
        if (x==y->lc) lfix(x),rfix(x);
        else rfix(y),rfix(x);
    }
    else {
      if (x==y->lc) lfix(x);
      else rfix(x);
    }
  }
  _root = x;
}

void do_same(node *x,int _m) {
  if (x==null) return;
  x->ky = _m;
  x->pt = 1;
  x->co = _m*x->sz;
  x->sm = x->rm = x->lm = max(_m,x->co);
}

void do_reserve(node *x) {
  if (x==null) return;
  x->rv ^= 1;
  swap(x->lc,x->rc);
  swap(x->lm,x->rm);
}

void check(node *x) {
  if (x==null) return;
  if (x->rv) {
    do_reserve(x->lc);
    do_reserve(x->rc);
    x->rv = 0;
  }
  if (x->pt) {
    do_same(x->lc,x->ky);
    do_same(x->rc,x->ky);
    x->pt = 0;
  }
}

void build(node *pa,node *&x,int l,int r) {
  int m = (l+r)>>1;
  x = new node(dat[m],pa,null,null);
```

```
  if (l<m) build(x,x->lc,l,m-1);
  if (m<r) build(x,x->rc,m+1,r);
  update(x);
}

void splaykth(int k,node *&_root) {
  node *p = _root;
  check(p);
  while (k!=p->lc->sz+1) {
    if (k<p->lc->sz+1)
      p = p->lc;
    else {
      k -= p->lc->sz+1;
      p = p->rc;
    }
    check(p);
  }
  splay(p,_root);
}

node* readdata(int n) {
  for (int i = 0; i<n; i++) scanf("%d",dat+i);
  node *tmp = new node(dat[n-1],0,null,null);
  if (n>1) build(tmp,tmp->lc,0,n-2);
  update(tmp);
  return tmp;
}

inline node* get_itvl(int a,int b) {
  splaykth(b+1,head);
  splaykth(a-1,head->lc);
  return head->lc->rc;
}

int main() {
  scanf("%d%d",&n,&m);
  null = new node(0,0,0,0);
  head = new node(0,0,null,head,1); head->sz = 2;
  tail = new node(0,head,null,null,1); tail->sz = 1;
  root = readdata(n);
  root->rc = tail; tail->pa = root;
  head->rc = root; root->pa = head;
  splay(root,head);
  while (m--) {
    char str[32];
    scanf("%s",str);
    int posi,tot,col;
    node *p;
    switch (str[0]) {
      case 'I' :  //Insert
```

```
        scanf("%d%d",&posi,&tot);
        p = readdata(tot);
        splaykth(posi+1,head);
        p->rc = head->rc;
        head->rc->pa = p;
        p->pa = head;
        head->rc = p;
        splay(p,head);
        break;
      case 'D' :  //Delete
        scanf("%d%d",&posi,&tot);
        p = get_itvl(posi+1,posi+tot);
        p->pa->rc = null;
        p->pa = null;
        update(head->lc);
        update(head);
        dispose(p);
        break;
      case 'R' :  //Reverse
        scanf("%d%d",&posi,&tot);
        p = get_itvl(posi+1,posi+tot);
        do_reserve(p);
        check(p);
        splay(p,head);
        break;
      case 'G' :  //Interval's Operation
        scanf("%d%d",&posi,&tot);
        p = get_itvl(posi+1,posi+tot);
        printf("%d\n",p->co);
        break;
    }
  }
  return 0;
}
```

Validator: NOI2005 Sequence

1.2 Suffix Array

Last update: 2011-08-10 22:29:35

```
const int MAXN = 20000+3;
const int MAXL = 20000+3;
const int RANGE = 88;

int wa[MAXL],wb[MAXL],wv[MAXL],ws[MAXL];
int sa[MAXL],rank[MAXL],hei[MAXL];
int st[MAXL],tar[MAXN];
int n,len;
```

```
bool cmp(int *r, int i, int j, int l) {
  return r[i]==r[j] && r[i+l]==r[j+l];
}

void calc_sa() {
  int i,j,*x = wa,*y = wb,*t,m = 200,p;
  for (i = 0; i<=m; i++) ws[i] = 0;
  for (i = 1; i<=len; i++) ws[x[i] = st[i]]++;
  for (i = 1; i<=m; i++) ws[i] += ws[i-1];
  for (i = len; i>0; i--) sa[ws[x[i]]--] = i;
  for (j = 1,p = 0; j<len && p<len; j *= 2,m = p) {
    for (p = 0,i = len-j+1; i<=len; i++) y[++p] = i;
    for (i = 1; i<=len; i++)
      if (sa[i]>j) y[++p] = sa[i]-j;
    for (i = 0; i<=m; i++) ws[i] = 0;
    for (i = 1; i<=len; i++) wv[i] = x[y[i]];
    for (i = 1; i<=len; i++) ws[wv[i]]++;
    for (i = 1; i<=m; i++) ws[i] += ws[i-1];
    for (i = len; i>0; i--) sa[ws[wv[i]]--] = y[i];
    t = x; x = y; y = t; x[sa[1]] = 1;
    for (i = 2,p = 1; i<=len; i++)
      x[sa[i]] = cmp(y,sa[i-1],sa[i],j)?p:++p;
  }
  for (i = 1; i<=len; i++)
    rank[sa[i]] = i;
}

void calc_hei() {
  int h = 0,i,j;
  for (i = 1; i<=len; i++)
    if (rank[i]==1) hei[1] = h = 0;
    else {
      if (h) h--;
      j = sa[rank[i]-1];
      while (st[i+h]==st[j+h]) h++;
      hei[rank[i]] = h;
    }
}

memset(st,0,sizeof(st));
len = 0;
for (int i = 2; i<=n; i++)
  st[++len] = ...;
st[++len] = 0; st[len+1] = 0;
calc_sa();
calc_hei();
```

Validator: POJ1743

1.3 Treap

Last update: 2011-07-18 21:41:22

```
class treap {
  public:
    void insert(int key) {
        add(root,key);
    }
    void erase(int key) {
        del(root,key);
    }
    treap () {
        null = new node(0,INT_MAX,0,0);
        root = null;
    }
  private:
    struct node {
        node *lc,*rc;
        int lev,dat;
        node (int _dat,int _lev,node *_lc,node *_rc)
        : dat(_dat),lev(_lev),lc(_lc),rc(_rc) {}
        node () {}
    };
    node *root,*null;
    void rotl(node *&cur) {
        node *tmp = cur->lc;
        cur->lc = tmp->rc;
        tmp->rc = cur;
        cur = tmp;
    }
    void rotr(node *&cur) {
        node *tmp = cur->rc;
        cur->rc = tmp->lc;
        tmp->lc = cur;
        cur = tmp;
    }
    void add(node *&cur,int key) {
        if (cur==null) cur = new node(key,rand(),null,null);
        else if (cur->dat<key) {
            add(cur->rc,key);
            if (cur->rc->lev<cur->lev)
                rotr(cur);
      }
      else if (cur->dat>key) {
            add(cur->lc,key);
            if (cur->lc->lev<cur->lev)
                rotl(cur);
        }
    }
    void del(node *&cur,int key) {
```

```
        if (cur->dat<key) del(cur->rc,key);
        else if (cur->dat>key) del(cur->lc,key);
        else {
            if (cur->lc->lev<cur->rc->lev) rotl(cur);
            else rotr(cur);
            if (cur!=null) del(cur,key);
            else {
                delete cur->lc;
                cur->lc = 0;
            }
        }
    }
};
```

Validator: NONE

1.4 Leftist Heap

Last update: 2011-07-20 22:34:08

```
struct node {
    int key, dist;
    node *lch, *rch;
    node (int k) { key = k; dist = 0; lch = rch = NULL; };
};

class leftlist {
  private :
    node *ROOT;
    int tot;
    int Get_dist(node *a) { return a==NULL ? -1: a->dist; }
    void SWAP(node *&a, node *&b) { node *tmp = a; a = b; b = tmp; }
    node *Merge(node *a, node *b) {
        if (a == NULL) return b;
        if (b == NULL) return a;
        if (a->key < b->key) SWAP(a, b);
            a->rch = Merge(a->rch, b);
        if (Get_dist(a->lch) < Get_dist(a->rch)) SWAP(a->lch, a->rch);
            a->dist = Get_dist(a->rch) + 1;
        return a;
    }
  public :
    void clear() { ROOT = NULL; tot=0; }
    node *root() { return ROOT; }
    int size() { return tot; }
    bool empty() { return tot == 0; }
    void insert(int k) { ROOT = Merge(ROOT, new node (k)); tot++; }
    int top() { return ROOT != NULL ? ROOT->key : -0x7fffffff; }
    void pop() {
```

```
        if (ROOT == NULL) for(;;); //damn it
        node *tmp = ROOT;
        ROOT = Merge(ROOT->lch, ROOT->rch);
        delete tmp; tot--;
    }
    void merge(leftlist &t) {
        tot += t.size();
        ROOT = Merge(ROOT, t.root());
        t.clear();
    }
};
```

Validator: NONE

1.5 DFA(Trie)

Last update: 2011-07-21 23:51:17

```
const int MAXT = 100+1;

struct node_t {
  int ch[50],suf;
  bool dgr;
};

node_t trie[MAXT];
int que[MAXT];
int n,co;

int main() {
    memset(trie,0,sizeof(trie)); co = 1;
    // ... Do some insertions.
    int f = 0,r = 0;
    for (int i = 0; i<n; i++)
        if (trie[0].ch[i]!=0)
            que[r++] = trie[0].ch[i];
    while (f<r) {
        int t = que[f++],_suf = trie[t].suf;
        trie[t].dgr |= trie[_suf].dgr;
        if (!trie[t].dgr)
            for (int i = 0; i<n; i++)
                if (trie[t].ch[i]==0)
                    trie[t].ch[i] = trie[_suf].ch[i];
                else {
                    trie[trie[t].ch[i]].suf = trie[_suf].ch[i];
                    que[r++] = trie[t].ch[i];
                }
    }
}
```

Validator: Many

1.6 Code of QTREE

Last update: 2011-07-23 16:48:59

```
#include <cstdio>
#include <cstring>
#include <climits>
#include <algorithm>

using namespace std;

const int MAXN = 10000+5;
const int MAXE = MAXN*2;
const int MAXS = MAXN*2;

struct edge {
  int tar,nxt;
};

struct node {
  int l,r,mmin,mmax,lc,rc;
};

node st[MAXS];
edge e[MAXE];
int head[MAXN],n,val[MAXN],co_e,m,ptr;
int cnt[MAXN],wgt[MAXN],dep[MAXN],Q[MAXN],fa[MAXN];
int seg[MAXN],pos[MAXN],lis[MAXN],root[MAXN];
int lca[18][MAXN],lg[MAXN];
bool usd[MAXN];

void update(int x) {
  if (st[x].lc!=-1) {
    st[x].mmax = max(st[st[x].lc].mmax,st[st[x].rc].mmax);
    st[x].mmin = min(st[st[x].lc].mmin,st[st[x].rc].mmin);
  }
  else st[x].mmin = st[x].mmax = val[fa[lis[st[x].r]]>>1];
}

void build(int l,int r) {
  int x = m++;
  st[x].l = l; st[x].r = r;
  if (l+1<r) {
    st[x].lc = m; build(l,(l+r)>>1);
    st[x].rc = m; build((l+r)>>1,r);
    update(x);
  }
  else {
```

```
    st[x].lc = st[x].rc = -1;
    st[x].mmin = st[x].mmax = val[fa[lis[r]]>>1];
  }
}

void bfs() {
  int ft = 0,tl = 0;
  Q[tl++] = 0; fa[0] = -1;
  memset(usd,0,sizeof(usd));
  memset(cnt,0,sizeof(cnt));
  usd[0] = 1; dep[0] = 0;
  while (ft<tl) {
    int x = Q[ft++];
    for (int i = head[x]; i!=-1; i = e[i].nxt)
      if (!usd[e[i].tar]) {
        Q[tl++] = e[i].tar;
        usd[e[i].tar] = 1;
        fa[e[i].tar] = i^1;
        dep[e[i].tar] = dep[x]+1;
      }
  }
  for (int i = n-1,x; i>=0; i--) {
    wgt[x = Q[i]] = -1; cnt[x] = 1;
    for (int j = head[x]; j!=-1; j = e[j].nxt)
      if (dep[e[j].tar]>dep[x]) {
        cnt[x] += cnt[e[j].tar];
        if (wgt[x]<0 || cnt[wgt[x]]<cnt[e[j].tar])
          wgt[x] = e[j].tar;
      }
  }
  memset(usd,0,sizeof(usd));
  m = 0; ptr = 0;
  for (int i = 0,x; i<n; i++)
    if (!usd[Q[i]]) {
      int lptr = ptr;
      seg[x = Q[i]] = m;
      while (x!=-1) {
        pos[x] = ptr;
        lis[ptr++] = x;
        usd[x] = 1;
        root[x] = Q[i];
        x = wgt[x];
      }
      build(lptr,ptr-1);
    }
}

void doLCA() {
  lg[1] = 0;
  for (int i = 2; i<=n; i++)
```

```
    lg[i] = lg[i>>1]+1;
  for (int i = 0; i<n; i++)
    if (fa[i]<0) lca[0][i] = -1;
    else lca[0][i] = e[fa[i]].tar;
  for (int dp = 1; (1<<dp)<n; dp++)
    for (int i = 0; i<n; i++)
      if (lca[dp-1][i]<0) lca[dp][i] = -1;
      else lca[dp][i] = lca[dp-1][lca[dp-1][i]];
}

void init() {
  scanf("%d\n",&n);
  co_e = 0;
  memset(head,-1,sizeof(head));
  for (int i = 0,u,v; i<n-1; i++) {
    scanf("%d%d%d\n",&u,&v,val+i); u--; v--;
    e[co_e] = (edge){v,head[u]}; head[u] = co_e++;
    e[co_e] = (edge){u,head[v]}; head[v] = co_e++;
  }
  bfs();
  doLCA();
}

void modify(int x,int l,int r,int va) {
  if (l<=st[x].l && st[x].r<=r) {
    val[fa[lis[st[x].r]]>>1] = va;
    update(x);
    return;
  }
  int m = (st[x].l+st[x].r)>>1;
  if (l<m) modify(st[x].lc,l,r,va);
  if (r>m) modify(st[x].rc,l,r,va);
  update(x);
}

int getLCA(int u,int v) {
  if (dep[u]>dep[v]) swap(u,v);
  if (dep[u]<dep[v]) {
    int dlt = dep[v]-dep[u];
    while (dlt) {
      v = lca[__builtin_ctz(dlt)][v];
      dlt ^= dlt&-dlt;
    }
  }
  if (u==v) return u;
  for (int dp = lg[dep[u]]; dp>=0; dp--)
    if (lca[dp][u]!=lca[dp][v])
      u = lca[dp][u],v = lca[dp][v];
  return lca[0][u];
}
```

```
int query(int x,int l,int r) {
  if (l<=st[x].l && st[x].r<=r)
    return st[x].mmax;
  int m = (st[x].l+st[x].r)>>1,la = INT_MIN,lb = INT_MIN;
  if (l<m) la = query(st[x].lc,l,r);
  if (r>m) lb = query(st[x].rc,l,r);
  return max(la,lb);
}

int solveQ(int u,int v) {
  int ret = INT_MIN;
  while (root[u]!=root[v]) {
    if (v!=root[v]) ret = max(ret,query(seg[root[v]],pos[root[v]],pos[v]));
    ret = max(ret,val[fa[root[v]]>>1]);
    v = e[fa[root[v]]].tar;
  }
  if (u!=v) ret = max(ret,query(seg[root[v]],pos[u],pos[v]));
  return ret;
}

char buf[128],*o;
inline int getint() {
  while (!isdigit(*o)&&*o!='-') ++o;
  int r = 0,f = *o=='-'?++o,1:0;
  while (isdigit(*o)) r=r*10+*o++-'0';
  return (f?-r:r);
}

void work() {
  for (int u,v,k,c; o=gets(buf),buf[0]!='D';) {
    if (buf[0]=='C') { o += 6;
      k = getint()-1; c = getint();
      u = e[k<<1].tar; v = e[(k<<1)+1].tar;
      if (dep[u]>dep[v]) swap(u,v);
      if (root[u]==root[v])
        modify(seg[root[u]],pos[u],pos[v],c);
      else val[k] = c;
    }
    else { o += 5;
      u = getint()-1; v = getint()-1;
      c = getLCA(u,v);
      printf("%d\n",max(solveQ(c,u),solveQ(c,v)));
    }
  }
}

int main() {
  int ntest;
  scanf("%d\n",&ntest);
```

```
  while (ntest--) {
    init();
    work();
  }
  return 0;
}
```

Validator: [SPOJ]QTREE

1.7 Dynamic Trees

Last update: 2011-08-06 16:43:09

```
#include <cstdio>
#include <algorithm>
#include <cstring>

using namespace std;

const int MAXN = 30000+5;

struct node {
  int w,s,l,r,f,v;
  node() {
    l = r = f = -1;
    w = s = v = 0;
  }
};

node t[MAXN];
int fa[MAXN],n,Q;
int lis[MAXN],m;

int getsum(int x) {
  return x<0?0:t[x].s;
}

void update(int x) {
  t[x].s = getsum(t[x].l)+getsum(t[x].r)+t[x].w;
}

int root(int x) {
  return fa[x]==x?x:(fa[x] = root(fa[x]));
}


void lup(int u) {
  int v = t[u].f;
  if (v<0) return;
```

```
  int f = t[v].f;
  if (f!=-1) {
    if (t[f].l==v) t[f].l = u;
    else if (t[f].r==v) t[f].r = u;
  }
  t[u].f = f;
  t[v].l = t[u].r;
  if (t[u].r!=-1) t[t[u].r].f = v;
  t[u].r = v; t[v].f = u;
  update(v); update(u);
}

void rup(int u) {
  int v = t[u].f;
  if (v<0) return;
  int f = t[v].f;
  if (f!=-1) {
    if (t[f].l==v) t[f].l = u;
    else if (t[f].r==v) t[f].r = u;
  }
  t[u].f = f;
  t[v].r = t[u].l;
  if (t[u].l!=-1) t[t[u].l].f = v;
  t[u].l = v; t[v].f = u;
  update(v); update(u);
}

void splay(int u) {
  m = 0;
  for (int v = u; 1; ) {
    lis[m++] = v;
    if (t[v].f!=-1 && (t[t[v].f].l==v || t[t[v].f].r==v))
      v = t[v].f;
    else break;
  }
  for (int i = m-1,j; i>=0; i--)
    if (t[j = lis[i]].v) {
      swap(t[j].l,t[j].r);
      t[j].v = 0;
      if (t[j].l!=-1) t[t[j].l].v ^= 1;
      if (t[j].r!=-1) t[t[j].r].v ^= 1;
    }
  while (t[u].f!=-1 && (t[t[u].f].l==u || t[t[u].f].r==u)) {
    int v = t[u].f;
    if (t[v].f!=-1 && (t[t[v].f].l==v || t[t[v].f].r==v)) {
      int f = t[v].f;
      if (t[f].l==v) {
        if (t[v].l==u) lup(v);
        else rup(u);
        lup(u);
```

```
      }
      else {
        if (t[v].r==u) rup(v);
        else lup(u);
        rup(u);
      }
    }
    else {
      if (t[v].l==u) lup(u);
      else rup(u);
      break;
    }
  }
}

int access(int u) {
  splay(u);
  t[u].r = -1;
  update(u);
  int v = t[u].f;
  while (v!=-1) {
    splay(v);
    t[v].r = u;
    update(v);
    u = v;
    v = t[u].f;
  }
  return u;
}

void make_root(int u) {
  access(u);
  splay(u);
  //t[u].v = 1;  BE CAUTIOUS
}


int find_root(int u) {
  make_root(u);
  while (t[u].l!=-1) u = t[u].l;
  splay(u);
  return u;
}

void cut(int u) {
  make_root(u);
  int v = t[u].l,f = t[u].f;
  t[u].l = -1;
  t[v].f = f;
}
```

```
bool check(int x,int u) {
  if (u==-1) return 1;
  if (u==x) return 0;
  make_root(u);
  while (t[x].f!=-1 && (t[t[x].f].l==x || t[t[x].f].r==x))
    x = t[x].f;
  if (u==x) return 0;
  return 1;
}

int main() {
  scanf("%d",&n);
  for (int i = 0,j; i<n; i++) {
    fa[i] = i; scanf("%d",&j);
    t[i].s = t[i].w = j;
  }
  scanf("%d",&Q);
  char id[16];
  while (Q--) {
    scanf("%s",id);
    int u,v,x;
    switch (id[0]) {
      case 'b' : //Join
        scanf("%d%d",&u,&v); u--; v--;
        if (root(u)==root(v)) printf("no\n");
        else {
          printf("yes\n");
          fa[root(u)] = root(v);
          make_root(u);
          t[u].f = v;
        }
        break;
      case 'p' : //Change
        scanf("%d%d",&u,&x); u--;
        splay(u); t[u].w = x; update(u);
        break;
      case 'e' : //Sum
        scanf("%d%d",&u,&v); u--; v--;
        if (root(u)!=root(v)) printf("impossible\n");
        else {
          make_root(u);
          v = access(v);
          printf("%d\n",t[v].s);
        }
        break;
    }
  }
  return 0;
}
```

Validator: [CRO2009]OTOCI

Chapter 2

Graph

2.1 Cut-points

Last update: 2011-07-18 21:24:11

```
const int MAXN = 1000;

vector<int> g[MAXN];
int dfn[MAXN],cnt,fa[MAXN],low[MAXN],root;
bool usd[MAXN],iscut[MAXN];

void dfs(int u) {
    int child = 0;
    usd[u] = 1; dfn[u] = low[u] = cnt++;
    vector<int>::iterator i;
    for (i = g[u].begin(); i!=g[u].end(); i++) {
        int v = *i;
        if (!usd[v]) {
            fa[v] = u; child++; dfs(v);
            if (low[v]<low[u]) low[u] = low[v];
            if (u!=root && low[v]>=dfn[u]) iscut[u] = 1;
        }
        else if (fa[u]!=v)
            if (low[u]>dfn[v]) low[u] = dfn[v];
    }
    if (u==root && child>1) iscut[u] = 1;
}

int main() {
    ......
    cnt = 0;
    memset(iscut,0,sizeof(iscut));
    memset(usd,0,sizeof(usd));
    ......
}
```

Validator: POJ1523

2.2 SCC(Tarjan)

Last update: 2011-07-18 22:35:53

```
void dfs(int index) {
    low[index] = dfn[index] = cnt++;
    used[index] = 1;
    del[index] = 0;
    st[top++] = index;
    for (int i = last[index]; i!=-1; i = e[i].nxt)
        if (!used[e[i].y]) {
            dfs(e[i].y);
            if (low[index]>low[e[i].y]) low[index] = low[e[i].y];
        }
        else if (!del[e[i].y]) if (low[index]>dfn[e[i].y])
                                low[index] = dfn[e[i].y];
    if (low[index]==dfn[index]) {
        do {
            root[st[--top]] = index;
            del[st[top]] = 1;
        } while (st[top]!=index);
    }
}
```

Validator: POJ2723

2.3 Maximum matchings(Hungary)

```
bool hungary(int x) {
    if (usd[x]) return 0;
    usd[x] = 1;
    for (node *i = g[x]; i; i = i->nxt) {
        int k = i->tar;
        if (idx[k]==-1 || hungary(idx[k])) {
            idx[k] = x;
            return 1;
        }
    }
    return 0;
}

void solve() {
    memset(idx,-1,sizeof(idx));
    int ans = 0;
    for (int i = 0; i<n; i++) {
        memset(usd,0,sizeof(usd));
```

```
        if (hungry(i)) ans++;
    }
}
```

Validator: NONE

2.4 KM Algorithms

Last update: 2011-07-18 22:58:14

```
const int MAXN = 150;

int w[MAXN][MAXN],slack[MAXN];
int lx[MAXN],ly[MAXN],link[MAXN],n;
bool usdx[MAXN],usdy[MAXN];

bool hungary(int x) {
    usdx[x] = 1;
    for (int y = 0; y<n; y++) {
        if (usdy[y]) continue;
        int t = lx[x]+ly[y]-w[x][y];
        if (t==0) {
            usdy[y] = 1;
            if (link[y]==-1 || hungary(link[y])) {
                link[y] = x;
                return 1;
            }
        }
        else if (slack[y]>t) slack[y] = t;
    }
    return 0;
}

void KM() {
    memset(link,-1,sizeof(link));
    memset(lx,0,sizeof(lx));
    memset(ly,0,sizeof(ly));
    for (int i = 0; i<n; i++)
        for (int j = 0; j<n; j++)
            if (w[i][j]>lx[i]) lx[i] = w[i][j];
    for (int x = 0; x<n; x++) {
        for (int i = 0; i<n; i++) slack[i] = INT_MAX;
        for (;;) {
            memset(usdx,0,sizeof(usdx));
            memset(usdy,0,sizeof(usdy));
            if (hungary(x)) break;
            int minval = INT_MAX;
            for (int i = 0; i<n; i++)
                if (!usdy[i] && minval>slack[i])
```

```
                    minval = slack[i];
            for (int i = 0; i<n; i++)
                if (usdx[i]) lx[i] -= minval;
            for (int i = 0; i<n; i++)
                if (usdy[i]) ly[i] += minval;
                else slack[i] -= minval;
        }
    }
}
```

Validator: URAL1076

2.5 Kth Shortest Path

Last update: 2011-07-18 21:24:40

```
struct node {
    int val,x;
    node(int _val,int _x)
    : val(_val),x(_x) {}
    node() {}
};

struct cmp {
    bool operator () (const node &a,const node &b) {
        return a.val>b.val;
    }
};

const int MAXN = 1000;

priority_queue<node,vector<node>,cmp> Q;
vector<node> F[MAXN],B[MAXN];
int n,m,S,T,K,dist[MAXN],co[MAXN];
bool usd[MAXN];

typedef vector<node>::iterator vni;

void dijkstra() {
    for (int i = 0; i<n; i++) dist[i] = INT_MAX;
    dist[T] = 0;
    memset(usd,0,sizeof(usd));
    for (int i = 0; i<n; i++) {
        int maxd = INT_MAX,maxv = -1;
        for (int j = 0; j<n; j++)
            if (!usd[j] && dist[j]<maxd)
                maxd = dist[maxv = j];
        usd[maxv] = 1;
        for (vni j = B[maxv].begin(); j!=B[maxv].end(); j++)
```

```
            if (!usd[j->x] && dist[j->x]>maxd+j->val)
                dist[j->x] = maxd+j->val;
    }
}

int search() {
    Q.push(node(dist[S],S));
    memset(co,0,sizeof(co));
    while (!Q.empty() && co[T]<K) {
        node u = Q.top(); Q.pop();
        co[u.x]++;
        if (u.x==T && co[T]==K) return u.val;
        for (vni i = F[u.x].begin(); i!=F[u.x].end(); i++)
            if (co[i->x]<K) Q.push(node(u.val-dist[u.x]+i->val+dist[i->x],i->x))
                ;
    }
    return -1;
}

int main() {
    scanf("%d%d",&n,&m);
    for (int a,b,c; m--; ) {
        scanf("%d%d%d",&a,&b,&c); a--; b--;
        F[a].push_back(node(c,b));
        B[b].push_back(node(c,a));
    }
    scanf("%d%d%d",&S,&T,&K); S--; T--;
    dijkstra();
    if (S==T) K++;
    printf("%d\n",search());
    return 0;
}
```

Validator: POJ2449

2.6 Maxflow - SAP

Last update: 2011-07-18 21:24:53

```
const int MAXE = ...; //CAUTIOUS!

struct edge {
    int tar,f;
    edge *nxt,*bck;
    edge(int _tar,int _f,edge *_nxt)
    : tar(_tar),f(_f),nxt(_nxt) {}
    edge() {}
};
```

```
edge e[MAXE],*g[MAXN],*bakg[MAXN];
int co_edge,dist[MAXN],co[MAXN+1]; //Cautious!!
int S,T,N,n,m;

void addedge(int a,int b,int c) {
    e[co_edge] = edge(b,c,g[a]); g[a] = &e[co_edge++];
    e[co_edge] = edge(a,0,g[b]); g[b] = &e[co_edge++];
    g[a]->bck = g[b]; g[b]->bck = g[a];
}

int adjust(int u,int dlt) {
    if (u==T) return dlt;
    int tmp = dlt,v;
    for (edge *&i = g[u]; i; i = i->nxt)
        if (i->f>0 && dist[v = i->tar]+1==dist[u]) {
            int _dlt = adjust(v,min(tmp,i->f));
            i->f -= _dlt; i->bck->f += _dlt; tmp -= _dlt;
            if (dist[S]==N || tmp==0) return dlt-tmp;
        }
    int minval = N;  //CAUTIOUS!
    for (edge *i = g[u] = bakg[u]; i; i = i->nxt)
        if (i->f>0 && dist[i->tar]+1<minval)
            minval = dist[i->tar]+1;
    if (--co[dist[u]]==0) dist[S] = N;
    else ++co[dist[u] = minval];
    return dlt-tmp;
}

int SAP() {
    S = 0; T = n-1; N = n;
    memcpy(bakg,g,sizeof(g));
    memset(dist,0,sizeof(dist));
    memset(co,0,sizeof(co));
    co[0] = N;
    int maxflow = 0;
    while (dist[S]<N) maxflow += adjust(S,INT_MAX);
    return maxflow;
}

int main() {
    ......
    memset(g,0,sizeof(g));
    co_edge = 0;
    ......
}
```

2.7 Minimum-cost Flow

Last update: 2011-08-10 22:21:26

```
const int MAXH = 100+5;
const int MAXN = MAXH*2+2+5;
const int MAXE = (MAXH*MAXH+MAXH*2)*2;

struct edge {
  int flw,cst,tar;
  edge *bck,*nxt;
  edge(int _tar,int _cst,int _flw,edge *_nxt)
    : tar(_tar),cst(_cst),flw(_flw),nxt(_nxt) {}
  edge() {}
};

struct node {
  int p,q;
};

edge e[MAXE],*g[MAXN];
int dis[MAXN],ans,ptr;
int s,t,n,m;
bool usd[MAXN];

void addedge(int s,int t,int c,int f = 1) {
  e[ptr] = edge(t,c,f,g[s]);  g[s] = &e[ptr++];
  e[ptr] = edge(s,-c,0,g[t]); g[t] = &e[ptr++];
  g[s]->bck = g[t]; g[t]->bck = g[s];
}

int adjust(int x,int delta) {
  if (x==s) {
    ans += dis[t]*delta;
    return delta;
  }
  usd[x] = 1;
  for (edge *i = g[x]; i!=NULL; i = i->nxt)
    if (i->flw && !usd[i->tar] && dis[i->tar]+i->cst==dis[x])
      if (int _delta = adjust(i->tar,delta>i->flw?i->flw:delta)) {
        i->flw -= _delta; i->bck->flw += _delta;
        return _delta;
      }
  return 0;
}

bool relabel() {
  int delta = INT_MAX,tmp;
  for (int i = 0; i<=t; i++) if (usd[i])
    for (edge *j = g[i]; j!=NULL; j = j->nxt) if (!usd[j->tar])
      if (j->flw && (tmp = j->cst-dis[i]+dis[j->tar])<delta)
```

```
        delta = tmp;
  if (delta==INT_MAX) return 0;
  for (int i = 0; i<=t; i++)
    if (usd[i]) dis[i] += delta;
  return 1;
}

ans = 0; t = (s = ...)+1; ptr = 0;
memset(g,0,sizeof(g));
for (int i = 0; i<cx; i++) addedge(i,s,0);
for (int i = 0; i<cy; i++) addedge(t,cx+i,0);
for (int i = 0; i<cx; i++)
    for (int j = 0; j<cy; j++)
        addedge(cx+j,i,dist(i,j));
memset(dis,0,sizeof(dis));
do
    do memset(usd,0,sizeof(usd));
    while (adjust(t,INT_MAX));
while (relabel());
printf("%d\n",ans);
```

Validator: POJ2195

2.8 Minimum Cut without Source and Sink

Last update: 2011-07-22 00:05:30

```
const int MAXN = 500;

bool del[MAXN],usd[MAXN];
int C[MAXN][MAXN],n,m,Q[MAXN],dis[MAXN];

memset(C,0,sizeof(C));
for (int i = 0,a,b,c; i<m; i++) {
    scanf("%d%d%d",&a,&b,&c);
    C[a][b] += c;
    C[b][a] += c;
}
memset(del,0,sizeof(del));
int ans = n*n;
for (int i = 0; i<n-1; i++) {
    int co = 0;
    for (int j = 0; j<n; j++)
        if (!del[j]) Q[co++] = j;
    memset(usd,0,sizeof(usd));
    memset(dis,0,sizeof(dis));
    int prev = -1;
    for (int j = 0; j<co; j++) {
        int maxd = -1,maxv;
```

```
        for (int k = 0; k<co; k++)
            if (!usd[k] && dis[k]>maxd)
                maxd = dis[maxv = k];
        if (j==co-1) {
            if (ans>maxd) ans = maxd;
            del[Q[maxv]] = 1;
            for (int k = 0; k<co; k++)
                if (k!=prev && k!=maxv)
                    C[Q[prev]][Q[k]] = (C[Q[k]][Q[prev]] += C[Q[k]][Q[maxv]]);
        }
        usd[prev = maxv] = 1;
        for (int k = 0; k<co; k++)
            if (!usd[k]) dis[k] += C[Q[maxv]][Q[k]];
    }
}
```

Validator: POJ2914

2.9 Dijkstra + Heap

Last update: 2011-07-23 16:51:52

```
const int MAXN = 10000+5;

typedef pair<int,int> pii;
typedef vector<pii>::iterator iter;

vector<pii> road[MAXN];
int n,dis[MAXN],size;
int heap[MAXN],hash[MAXN];

inline void trylo(int u) {
  int tmp = heap[u],i = u;
  while (i*2+1<size) {
    int j = i*2+1;
    if (dis[heap[j+1]]<dis[heap[j]]) j++;
    if (dis[heap[j]]<dis[tmp]) {
      heap[i] = heap[j];
      hash[heap[i]] = i;
      i = j;
    }
    else break;
  }
  heap[i] = tmp;
  hash[heap[i]] = i;
}

inline void tryhi(int u) {
  int tmp = heap[u],i = u;
```

```
  while (i>0) {
    int j = (i-1)/2;
    if (dis[tmp]<dis[heap[j]]) {
      heap[i] = heap[j];
      hash[heap[i]] = i;
      i = j;
    }
    else break;
  }
  heap[i] = tmp;
  hash[heap[i]] = i;
}

inline void push(int u) {
  heap[size] = u; hash[u] = size++;
  tryhi(size-1);
}

inline int htop() {
  int ret = heap[0];
  hash[heap[0]] = -1;
  heap[0] = heap[size-1];
  hash[heap[0]] = 0;
  size--;
  trylo(0);
  return ret;
}

int dijkstra(int x,int y) {
  memset(dis,127,sizeof(dis));
  dis[x] = 0;
  size = 0;
  memset(hash,-1,sizeof(hash));
  push(x);
  while (size) {
    int u = htop();
    for (iter i = road[u].begin(); i!=road[u].end(); ++i) {
      int v = i->second;
      if (dis[v]>dis[u]+i->first) {
        dis[v] = dis[u]+i->first;
        if (hash[v]==-1) push(v);
        else tryhi(hash[v]);
      }
    }
  }
  return dis[y];
}

scanf("%d",&n);
for (int i = 0,m; i<n; i++) {
```

```
    scanf("%d",&m);
    road[i].clear();
    for (int j = 0,t,c; j<m; j++) {
      scanf("%d%d",&t,&c); t--;
      road[i].push_back(pii(c,t));
    }
  }
```

Validator: [SPOJ]SHPATH

Chapter 3

Math

3.1 GCD and Extended GCD

Last update: 2011-07-18 21:25:03

```
int gcd(int a,int b) {
    return b==0?a:gcd(b,a%b);
}

int egcd(int a,int b,int &x,int &y) {
    if (b==0) {
        x = 1; y = 0;
        return a;
    }
    int d = egcd(b,a%b,x,y);
    int t = x; x = y; y = t-a/b*y;
    return d;
}
```

Validator: NONE

3.2 Modular Equation and Modular Equation System

Last update: 2011-07-18 21:25:11

```
void modular_eq(int a,int b,int n) {  //SOLVE : a = b (mod n)
    int x,y,d = egcd(a,n,x,y);
    if (b%d) return;
    int e = x*(b/d)%n;
    for (int i = 0; i<d; i++)
        ans[i] = (e+i*n/d)%n;
}

int modular_sys(int *b,int *m,int k) {  //SOLVE: x = b[i] (mod m[i])  i=1~k
```

```
    int M = 1;
    for (int i = 0; i<k; i++) M *= m[i];
    int ans = 0;
    for (int i = 0; i<k; i++) {
        int Mi = M/m[i],pi,qi;
        egcd(Mi,m[i],pi,qi);
        ans = (ans+Mi*pi*b[i])%M;
    }
    if (ans<0) ans += M;
    return ans;
}
```

Validator: NONE

3.3 Miller-Rabin Primality Test

Last update: 2011-07-21 23:55:14

```
const int NBASE = 5;
const int base[NBASE] = {2,3,5,7,11};

//n < 1,373,653, a = 2 and 3;
//n < 9,080,191, a = 31 and 73;
//n < 4,759,123,141, a = 2, 7, and 61;
//n < 2,152,302,898,747, a = 2, 3, 5, 7, and 11;
//n < 3,474,749,660,383, a = 2, 3, 5, 7, 11, and 13;
//n < 341,550,071,728,321, a = 2, 3, 5, 7, 11, 13, and 17.

bool miller(int x) {
    if (x<=1) return 0;
    if (x==2) return 1;
    if (x==3) return 1;
    if ((x&1)==0) return 0;
    int s = 0,d = x-1; // x-1 = 2^s * d
    for (; d>0 && (d&1)==0; d >>= 1) s++;
    for (int i = 0; i<NBASE && base[i]<x-1; i++) {
        long long t = exp(base[i],d,x);
        if (t==1 || t==x-1) continue;
        bool good = 0;
        for (int j = 1; !good && j<s; j++) {
            t = (t*t)%x;
            if (t==1) return 0;
            if (t==x-1) good = 1;
        }
        if (!good) return 0;
    }
    return 1;
}
```

Validator: from Wikipedia

3.4 Multiply with MOD

Last update: 2011-08-10 22:20:20

```
LL mul(LL a,LL b,LL c) {  //a*b%c
    LL ret = 0,tmp = a;
    while (b<0) b += c;
    while (tmp<0) tmp += c;
    while (b) {
        if (b&0x1) if ((ret += tmp)>=c) ret -= c;
        if ((tmp <<= 1)>=c) tmp -= c;
        b >>= 1;
    }
    return ret%c;
}
```

Validator: ACM Asia 2008 Chengdu - Toy

Chapter 4

Misc

4.1 Stable Marriage Problem

Last update: 2011-07-23 16:58:24

```
1  Initialize all m ∈ M and w ∈ W to free;
2  while ∃ free man m who still has a woman w to propose to do
3      w = m's highest ranked such woman who he has not proposed to yet;
4      if w is free then
5          (m, w) become engaged;
6      end
7      else if some pair (m', w) already exists then
8          if w prefers m to m' then
9              (m, w) become engaged;
10             m' becomes free;
11         end
12         else
13             (m', w) remain engaged;
14         end
15     end
16 end
```

Algorithm 1: Gale-Shapley Algorithm

Validator: NONE

4.2 KMP and Extended KMP

Last update: 2011-08-06 17:52:43

```
pi[1] = 0;
```

```
for (int i = 2,j = 0; i<=n; i++) {
    while (j && s[j+1]!=s[i]) j = pi[j];
    if (s[j+1]==s[i]) j++;
    pi[i] = j;
}

ptr = 0;
for (int i = 1; i<=m; i++) { //CAUTIOUS
    while (ptr>0 && s[ptr+1]!=t[i]) ptr = pi[ptr];
    if (s[ptr+1]==t[i]) ptr++;
    if (ptr==n) {
        //FIND A SOLUTION
        ptr = pi[ptr];
    }
}

void prefix(char *t,int *nxt) {
    int i = 0,max = 0;
    int n = strlen(t);
    while (t[i]==t[i+1]) ++i;
    nxt[1] = i; max = 1;
    for (i = 2; i<n; ++i) {
        int len = max+nxt[max]-1;
        int l = nxt[i-max];
        if (l<len-i+1) nxt[i] = l;
        else {
            int j = len-i+1;
            if (j<0) j = 0;
            while (t[i+j]==t[j] && i+j<n) ++j;
            nxt[i] = j;
            max = i;
        }
    }
}

void KMP(char *s,char *t,int *ext,int *nxt) {
    int i = 0,max;
    int n = strlen(s),m = strlen(t);
    prefix(t,nxt);
    while (t[i]==s[i] && i<n && i<m) ++i;
    ext[0] = i; max = 0;
    for (i = 1; i<n; ++i) {
        int len = max+ext[max]-1;
        int l = nxt[i-max];
        if (l<len-i+1) ext[i] = l;
        else {
            int j = len-i+1;
            if (j<0) j = 0;
            while (s[i+j]==t[j] && j<m && i+j<n) ++j;
            ext[i] = j;
```

```
            max = i;
        }
    }
}
```

Validator: POJ1961

4.3 Palindrome

Last update: 2011-07-18 21:25:19

```
void calc(int pos,char *s,int *len,int slen) {
    bool flag = 1;
    int ans = 0;
    for (int i = 1; flag; i++)
        if (pos-i>=0 && pos+i<slen)
            if (s[pos-i]==s[pos+i]) {
                i += min(len[pos-i],len[pos+i]);
                ans = i;
            }
            else flag = 0;
        else flag = 0;
    len[pos] = ans;
}

void dfs(int l,int r,char *s,int *len,int slen) {
    if (l>r) return;
    int mid = (l+r)/2;
    calc(mid,s,len,slen);
    dfs(l,mid-1,s,len,slen);
    dfs(mid+1,r,s,len,slen);
}

void palin(char *s,int *len) {
    int l = strlen(s);
    memset(len,0,sizeof(int)*(l*2+2));
    for (int j = l-1; j>=0; j--) {
        s[j*2+1] = s[j];
        s[j*2] = '␣';
    }
    s[l*2] = '␣';
    s[l*2+1] = 0;
    dfs(0,l*2,s,len,l*2+1);
}
```

Validator: POJ3974

4.4 Hash

Last update: 2011-07-20 22:26:02

```
int hash_int(int *key) {
    hash = 0;
    for (int i = 1; i<=n; i++)
        hash = ((hash<<2)+(key[i]>>4))^(key[i]<<10);
    return hash*MOD;
}

int Elfhash(const char *key) {
    unsigned long h = 0;
    while (*key) {
        h = (h<<4)+*key++;
        unsigned long g = h&0xF0000000L;
        if (g) h ^= g>>24;
        h &= ~g;
    }
    return h%M;
}
```

Validator: No need of it

4.5 HighPrecision

Last update: 2011-07-20 22:25:51

```
const int MAXL = ...; //CAUTIOUS!

int compare(int *a, int *b) {
    if (a[0]>b[0]) return 1;
    else if (a[0]<b[0]) return -1;
    for (int i = a[0]; i>0; i--)
        if (a[i]>b[i]) return 1;
        else if (a[i]<b[i]) return -1;
    return 0;
}

void extend(int *x) {
    x[0]++;
    for (int i = x[0]; i>1; i--)
        x[i] = x[i-1];
    x[1] = 0;
    if (!x[x[0]]) x[0]--;
}

void subtract(int *a, int *b) {
    for (int i = 1; i<=a[0]+1; i++) {
        a[i] -= b[i];
```

```
        if (a[i]<0) {
            a[i] += 10;
            a[i+1]--;
        }
    }
    while (a[0]>0 && !a[a[0]]) a[0]--;
}

void divide(int *a, int *b, int *c, int *d) {
    for (int i = 0; i<MAXL; i++) c[i] = 0;
    for (int i = 0; i<MAXL; i++) d[i] = 0;
    d[0] = 1;
    for (int i = a[0]; i>0; i--) {
        extend(d);
        d[1] = a[i];
        while (compare(d,b)>=0) {
            subtract(d,b);
            c[i]++;
        }
    }
    for (c[0] = a[0]; c[0]>0 && !c[c[0]]; c[0]--);
}
```

Validator: NONE

4.6 Multiplicative Inverse

Last update: 2011-07-23 16:29:53

```
int[] inv = new int[MAXN];
inv[1] = 1;
for (int i = 2; i<MAXN; i++)
    inv[i] = inv[MOD%i]*(MOD-MOD/i)%MOD; //MOD must be a prime
```

Validator: a problem from TopCoder

4.7 Faster Input

Last update: 2011-07-23 16:30:52

```
char buf[BUF_SIZE],*o;
inline int getint() {
  while (!isdigit(*o) && *o!='-') ++o;
  int r = 0,f = *o=='-'?++o,1:0;
  while (isdigit(*o)) r = r*10+*o++-'0';
  return (f?-r:r);
}
//o = gets(buf);
```

Validator: No need of it

4.8 Volume of a Tetrahedron

Last update: 2011-07-23 16:35:34

$$T = a^2c^2d^2 + b^2c^2d^2 + a^2b^2f^2 + a^2c^2f^2 + a^2d^2f^2 + c^2d^2f^2 + a^2b^2e^2 + b^2c^2e^2 + b^2d^2e^2 + c^2d^2e^2 + a^2f^2e^2 + b^2f^2e^2$$

$$V = \frac{\sqrt{T - a^2b^2d^2 - a^2c^2e^2 - b^2c^2f^2 - d^2f^2e^2 - c^4d^2 - c^2d^4 - a^4f^2 - a^2f^4 - b^4e^2 - b^2e^4}}{12}$$

```
#define s(x) (x*x)

double a,b,c,d,e,f;

scanf("%lf%lf%lf%lf%lf%lf",&a,&b,&c,&d,&e,&f);
double v = s(a*b*e)+s(a*b*f)+s(a*c*d)+s(a*c*f)+s(a*d*f)+s(a*f*e)
          +s(b*c*d)+s(b*c*e)+s(b*d*e)+s(b*f*e)+s(c*d*e)+s(c*d*f)
          -s(a*b*d)-s(a*c*e)-s(b*c*f)-s(d*e*f)
          -s(c*c*d)-s(c*d*d)-s(a*a*f)-s(a*f*f)-s(b*b*e)-s(b*e*e);
v = sqrt(v)/12;
```

Validator: [SPOJ]TETRA

Chapter 5

Java

5.1 Input and Output

Last update: 2011-07-18 21:25:27

```
public static void main (String [] args) throws IOException {
    BufferedReader f = new BufferedReader(new FileReader(".."));
    PrintWriter out = new PrintWriter(new BufferedWriter(new FileWriter("..")));
    StringTokenizer st = new StringTokenizer(f.readLine());
                        // Get line, break into tokens
    out.println(i1+i2);                             // output result
    out.close();                                    // close the output file
    System.exit(0);                                 // don't omit this!
}
```

Provided by USACO Java Tips