Contents

1 DataStructure 1
- 1.1 sparse table 1
- 1.2 zkw tree 1
- 1.3 2Dstructure 2
- 1.4 LiChaoTree 2
- 1.5 LCT 3
- 1.6 treap 3
- 1.7 segment tree dynamic 4
- 1.8 segment tree array 4
- 1.9 ConvexHull 5

2 Math 5
- 2.1 rho 5
- 2.2 inversion 6
- 2.3 LL-Multiplication 6
- 2.4 CRT 6
- 2.5 geometry 7
- 2.6 FWT 7
- 2.7 FFT-precision 8
- 2.8 FFT 8
- 2.9 linear sieve 9
- 2.10 NTT 9
- 2.11 miller rabin 9

3 String 9
- 3.1 ac automation 9
- 3.2 zvalue 10
- 3.3 kmp 10
- 3.4 DC3 10
- 3.5 suffix array 11

4 Graph 12
- 4.1 clique 12
- 4.2 steiner 12
- 4.3 spfa 12
- 4.4 global-min-cut 12
- 4.5 centroid decomp 13
- 4.6 lca 13
- 4.7 ap 14
- 4.8 bridge 14
- 4.9 scc 14
- 4.10 dijkstra 14
- 4.11 DynamicConnectivity 15
- 4.12 hld 17

5 FlowAndMatching 17
- 5.1 hlpp 17
- 5.2 dinic 18
- 5.3 km o3 18
- 5.4 bipartite matching 19
- 5.5 LowerBoundFlow 19
- 5.6 matching 19
- 5.7 km o4 19
- 5.8 VKMV 20
- 5.9 mcmf 20
- 5.10 blossom 20

6 MISC 21
- 6.1 template 21
- 6.2 raw string 21
- 6.3 pb ds 21
- 6.4 Random 22
- 6.5 vimrc 22

1 DataStructure

1.1 sparse table

```
int st[MAXLG][MAXN];
void build(){
  for(int i=1;i<MAXLG;i++){
    for(int j=0;j<MAXN;j++){
      if(j+(1<<(i-1)) >= MAXN)continue;
      st[i][j] = min(st[i-1][j],st[i-1][j+(1<<(i-1))
    ]);
    }
  }
}

int query(int l,int r){ // [l,r]
  int E = __lg(r-l);
  return min(st[E][l],st[E][r-(1<<E)+1]);
}
```

1.2 zkw tree

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

const int MAXN = 100005;

int n, zkw[MAXN*2];

/*
    query: range max
    add: single change value
*/
void build () {
    for (int i=n-1; i>0; i--) {
        zkw[i] = max(zkw[i<<1], zkw[i<<1|1]);
    }
}

void chg (int x, int val) {
    for (zkw[x+=n]=val; x>1; x>>=1) {
        zkw[x>>1] = max(zkw[x], zkw[x^1]);
    }
}

int qry (int l, int r) {
    int ret = -0x3f3f3f3f;
    for (l+=n,r+=n; l<r; l>>=1, r>>=1) {
        if (l&1) {
            ret = max(ret, zkw[l++]);
        }
        if (r&1) {
            ret = max(ret, zkw[--r]);
        }
    }
    return ret;
}

int main () {
    cin >> n;
    for (int i=0; i<n; i++) {
        cin >> zkw[i+n];
    }

    build();
    int cmd;
    while (cin >> cmd) {
        int l, r, x, v;
        if (cmd == 1) {
            cin >> l >> r;
            cout << qry(l, r) << endl;
        } else {
            cin >> x >> v;
            chg(x, v);
        }
    }
}
```

1.3 2Dstructure

```
const int Zero = 0;
inline int opt(const int &a, const int &b){
  return a+b;
}

int height, width;
int qx, qy, qX, qY;

struct Seg{
  int val;
  Seg *lc, *rc;
};

struct Seg2D{
  Seg *O;
  Seg2D *lc, *rc;
};

Seg* build(int l, int r){
  Seg* ret = new Seg();
  if (l==r) {
    cin>>ret->val;
    return ret;
  }
  int mid  = (l+r)>>1;
  ret->lc = build(l,mid);
  ret->rc = build(mid+1,r);
  ret->val=opt(ret->lc->val, ret->rc->val);
  return ret;
}

Seg* merge(int l, int r, Seg *tl, Seg *tr){
  Seg* ret = new Seg();
  ret->val = opt( tl->val, tr->val);

  if (l!=r){
    int mid = (l+r)>>1;
    ret->lc = merge(l,mid,tl->lc,tr->lc);
    ret->rc = merge(mid+1,r,tl->rc,tr->rc);
  }

  return ret;
}


Seg2D* build2D(int l, int r){
  Seg2D* ret = new Seg2D();
  if (l==r){
    ret->O = build(1,width);
    return ret;
  }
  int mid  = (l+r)>>1;
  ret->lc = build2D(l,mid);
  ret->rc = build2D(mid+1,r);
  ret->O = merge(1,width,ret->lc->O,ret->rc->O);
  return ret;
}

int query(Seg* o, int l, int r, int L, int R){
  if (r<L || R<l) return Zero;
  if (L<=l && r<=R) return o->val;
  int mid = (l+r)>>1;
  int ql = query(o->lc,l,mid,L,R);
  int qr = query(o->rc,mid+1,r,L,R);
  return opt(ql,qr);
}

int query2D(Seg2D* o, int l, int r, int L, int R){
  if (r<L || R<l) return Zero;
  if (L<=l && r<=R) return query(o->O,1,width,qx,qX)
   ;
  int mid = (l+r)>>1;
  int ql = query2D(o->lc,l,mid,L,R);
  int qr = query2D(o->rc,mid+1,r,L,R);
  return opt(ql,qr);
}

int pX, pY, v;

void modify(Seg*o, int l, int r, int p, int v){
  if (l>p||r<p) return;
  if (l==r) {
    o->val=v;
    return;
  }
  int mid = (l+r)>>1;
  modify(o->lc,l,mid,p,v);
  modify(o->rc,mid+1,r,p,v);
  o->val = opt(o->lc->val, o->rc->val);
}


void modify2D(Seg2D*o, int l, int r, int p){
  if (l>p||r<p) return;
  if (l==r){
    modify(o->O, 1, width, pX,v);
    return;
  }
  int mid = (l+r)>>1;
  modify2D(o->lc,l,mid,p);
  modify2D(o->rc,mid+1,r,p);
  int ql = query(o->lc->O,1,width,pX,pX);
  int qr = query(o->rc->O,1,width,pX,pX);
  modify(o->O,1,width,pX, opt(ql,qr) );
}

int main(){
  ios::sync_with_stdio(false);
  cin.tie(0);
  int n, q; cin>>n>>q;
  width = n;
  height = n;
  Seg2D *S = build2D(1, height);
  while (q--){
    int cmd;
    cin>>cmd;
    if (cmd==1){
      cin>>qy>>qx>>qY>>qX;
      if (qY<qy) swap(qY, qy);
      if (qX<qx) swap(qx, qX);
      cout<<query2D(S, 1, height, qy, qY)<<'\n';
    }else{
      cin>>pY>>pX>>v;
      modify2D(S, 1, height, pY);
    }

  }
}
```

1.4 LiChaoTree

```
struct Vec {
    ll x, y;
    ll eval (ll pos) {
        return pos*x + y;
    }
};

struct Node {
    int l, r;
    Node *lc, *rc;
    Vec bst;

    Node (int _l, int _r) : l(_l), r(_r) {
        lc = rc = nullptr;
        bst = {0, INF};
    }
};
Node *root[MAXN];

Node *addLine (Vec nw, Node *nd) {
    int mid = (nd->l + nd->r) >> 1;
    bool lnw = nw.eval(nd->l) < nd->bst.eval(nd->l);
    bool mnw = nw.eval(mid) < nd->bst.eval(mid);

    Node *ret = new Node(*nd);
    if (mnw) {
        swap(nw, ret->bst);
    }
    if (ret->l == ret->r - 1) {
        return ret;
    } else if (lnw != mnw) { // left
        if (!ret->lc) {
            ret->lc = new Node(ret->l, mid);
        }
        ret->lc = addLine(nw, ret->lc);
    } else {
        if (!ret->rc) {
            ret->rc = new Node(mid, ret->r);
        }
        ret->rc = addLine(nw, ret->rc);
    }

    return ret;
}

ll eval (ll x, Node *nd) {
    if (!nd) {
        return INF;
    }
    ll ret = nd->bst.eval(x);
    int mid = (nd->l + nd->r) >> 1;
    if (x >= mid) {
        ret = min(ret, eval(x, nd->rc));
    } else {
        ret = min(ret, eval(x, nd->lc));
    }
    return ret;
}
```

1.5 LCT

```
// from https://github.com/edisonhello/
    waynedisonitau123
struct node {
    node *ch[2], *fa, *pfa;
    int sum, v, rev, id;
    node(int s, int id): id(id), v(s), sum(s), rev
    (0), fa(nullptr), pfa(nullptr) {
        ch[0] = nullptr;
        ch[1] = nullptr;
    }
    int relation() {
        return this == fa->ch[0] ? 0 : 1;
    }
    void push() {
        if (!rev) return;
        swap(ch[0], ch[1]);
        if (ch[0]) ch[0]->rev ^= 1;
        if (ch[1]) ch[1]->rev ^= 1;
        rev = 0;
    }
    void pull() {
        sum = v;
        if (ch[0]) sum += ch[0]->sum;
        if (ch[1]) sum += ch[1]->sum;
    }
    void rotate() {
        if (fa->fa) fa->fa->push();
        fa->push(), push(), swap(pfa, fa->pfa);
        int d = relation();
        node *t = fa;
        if (t->fa) t->fa->ch[t->relation()] = this;
        fa = t->fa, t->ch[d] = ch[d ^ 1];
        if (ch[d ^ 1]) ch[d ^ 1]->fa = t;
        ch[d ^ 1] = t, t->fa = this;
        t->pull(), pull();
    }
    void splay() {
        while (fa) {
            if (!fa->fa) {
                rotate();
                continue;
            }
            fa->fa->push(), fa->push();
            if (relation() == fa->relation()) fa->
    rotate(), rotate();
            else rotate(), rotate();
        }
    }
    void evert() { access(), splay(), rev ^= 1; }
    void expose() {
        splay(), push();
        if (ch[1]) {
            ch[1]->fa = nullptr, ch[1]->pfa = this;
            ch[1] = nullptr, pull();
        }
    }
    bool splice() {
        splay();
        if (!pfa) return false;
        pfa->expose(), pfa->ch[1] = this, fa = pfa;
        pfa = nullptr, fa->pull();
        return true;
    }
    void access() {
        expose();
        while (splice());
    }
    int query() { return sum; }
};

namespace lct {
node *sp[maxn];
void make(int u, int v) {
    // create node with id u and value v
    sp[u] = new node(v, u);
}
void link(int u, int v) {
    // u become v's parent
    sp[v]->evert();
    sp[v]->pfa = sp[u];
}
void cut(int u, int v) {
    // u was v's parent
    sp[u]->evert();
    sp[v]->access(), sp[v]->splay(), sp[v]->push();
    sp[v]->ch[0]->fa = nullptr;
    sp[v]->ch[0] = nullptr;
    sp[v]->pull();
}
void modify(int u, int v) {
    sp[u]->splay();
    sp[u]->v = v;
    sp[u]->pull();
}
int query(int u, int v) {
    sp[u]->evert(), sp[v]->access(), sp[v]->splay();
    return sp[v]->query();
}
int find(int u) {
    sp[u]->access();
    sp[u]->splay();
    node *p = sp[u];
    while (true) {
        p->push();
        if (p->ch[0]) p = p->ch[0];
        else break;
    }
    return p->id;
}}
```

1.6 treap

```
struct Nd{
    int pri = rand();
    int val = 0, tag = 0, id = 0, idtg = 0, mx=0;
    Nd * lc=0, *rc = 0;
    Nd(int v, int pos) {
        val = mx=v; id = pos;
    }
};

inline void push(Nd *& o) {
    if (!o) return;
    if (o->tag) {
        o->val += o->tag;
        o->mx += o->tag;
        if (o->lc) o->lc->tag += o->tag;
        if (o->rc) o->rc->tag += o->tag;
        o->tag=0;
    }
    if (o->idtg) {
        o->id += o->idtg;
        if (o->lc) o->lc->idtg += o->idtg;
        if (o->rc) o->rc->idtg += o->idtg;
        o->idtg = 0;
    }
}

inline void pull(Nd *&o) {
    if (!o)return;
    o->mx = o->val;
    if (o->lc) o->mx = max(o->mx, o->lc->mx);
    if (o->rc) o->mx = max(o->mx, o->rc->mx);
}

Nd * merge(Nd *&A, Nd*&B) {
    push(A); push(B);
    if (!A) return B;
    if (!B) return A;
    if (A->pri > B->pri) {
        A->rc = merge(A->rc, B);
        push(A->lc);
        pull(A);
        return A;
    }else{
        B->lc = merge(A, B->lc);
        push(B->rc);
        pull(B);
        return B;
    }
}

void split(Nd *o, Nd * & A, Nd *& B, int id) {
    A=B=0;
    if (!o) return;
    push(o);
    if (o -> id < id) {
        A = o;
        split(o->rc, A->rc, B, id);
        push(A->lc);
        pull(A);
    }else{
        B = o;
        split(o->lc,A, B->lc, id);
        push(B->rc);
        pull(B);
    }
}
```

1.7 segment tree dynamic

```
struct Node {
    int l, r;
    Node *lc, *rc;
    int mx;
};
Node *root[MAXN];

int qry (int l, int r, Node *nd) {
    if (!nd) {
        return 0;
    } else if (nd->l == l && r == nd->r) {
        return nd->mx;
    } else {
        int mid = (nd->l + nd->r) >> 1;
        if (l >= mid) {
            return qry(l, r, nd->rc);
        } else if (r <= mid) {
            return qry(l, r, nd->lc);
        } else {
            return max(qry(l, mid, nd->lc), qry(mid,
 r, nd->rc));
        }
    }
}

void chg (int pos, int v, Node *nd) {
    if (nd->l == nd->r-1) {
        nd->mx = max(nd->mx, v);
    } else {
        int mid = (nd->l + nd->r) >> 1;
        if (pos >= mid) {
            if (!nd->rc) {
                nd->rc = new Node{mid, nd->r,
    nullptr, nullptr, 0};
            }
            chg(pos, v, nd->rc);
            nd->mx = max(nd->mx, nd->rc->mx);
        } else {
            if (!nd->lc) {
                nd->lc = new Node{nd->l, mid,
    nullptr, nullptr, 0};
            }
            chg(pos, v, nd->lc);
            nd->mx = max(nd->mx, nd->lc->mx);
        }
    }
}
```

1.8 segment tree array

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
#define REP(i, n) for(int i=0; i<n;i++)

const int MAXN = 100005;

int n, m, a[MAXN], len[MAXN*4], dt[MAXN*4], tag[MAXN
    *4];

void push (int o) {
    if (len[o] > 1 && tag[o] != 0) {
        tag[o<<1] += tag[o];
        tag[o<<1|1] += tag[o];
        dt[o] += tag[o] * len[o];
        tag[o] = 0;
    }
}

ll sum (int o) {
    return tag[o]*len[o] + dt[o];
}

void pull (int o) {
    dt[o] = sum(o<<1) + sum(o<<1|1);
}

void build (int o=1, int l=0, int r=n) {
    if (l == r - 1) {
        dt[o] = tag[o] = 0;
        len[o] = 1;
    } else {
        int mid = (l + r) >> 1;
        build(o<<1, l, mid);
        build(o<<1|1, mid, r);
        len[o] = len[o<<1] + len[o<<1|1];
        pull(o);
    }
}
```

```
ll query(int qL, int qR, int o=1, int nL=0, int nR=n
    ) {
    if (qR <= nL || qL >= nR || qL >= qR) {
        return 0;
    } else if (nL >= qL && nR <= qR) {
        return sum(o);
    } else {
        push(o);
        int mid = (nL + nR) >> 1;
        return query(qL, qR, o<<1, nL, mid) + query(
    qL, qR, o<<1|1, mid, nR);
    }
}

void modify(int qL, int qR, int val, int o=1, int nL
    =0, int nR=n) {
    if (qR <= nL || qL >= nR || qL >= qR) {
        return;
    } else if (nL >= qL && nR <= qR) {
        tag[o] += val;
    } else {
        push(o);
        int mid = (nL + nR) >> 1;
        modify(qL, qR, val, o<<1, nL, mid);
        modify(qL, qR, val, o<<1|1, mid, nR);
        pull(o);
    }
}

int main () {
    cin >> n;
    build();
    int cmd;
    while (cin >> cmd) {
        int l, r, v;
        if (cmd == 1) {
            cin >> l >> r >> v;
            modify(l, r, v);
        } else {
            cin >> l >> r;
            cout << query(l, r) << endl;
        }
    }
}

/*
10
1 0 3 3
0 0 5
1 2 4 2
0 0 5
*/
```

1.9 ConvexHull

```
// Lower Hull
bool QTYPE=0;
struct Line {
    mutable ll m, b, p;
    bool operator<(const Line& o) const {
        if (QTYPE) return p<o.p;
        return m < o.m;
    }
};

struct LineContainer : multiset<Line > {
    // (for doubles, use INF = 1/.0, div(a,b) = a/b)
    const ll INF = LLONG_MAX;
    ll div(ll A, ll B) { // floored division
        return A / B - ((A ^ B) < 0 && A % B); }
    bool isect(iterator x, iterator y) {
        if (y == end()) { x->p = INF; return false;
    }
        if (x->m == y->m) x->p = x->b > y->b ? INF :
     -INF;
        else x->p = div(y->b - x->b, x->m - y->m);
        return x->p >= y->p;
    }
    void add(ll m, ll b) {
        auto z = insert({m, b, 0}), y = z++, x = y;
        while (isect(y, z)) z = erase(z);
        if (x != begin() && isect(--x, y)) isect(x,
    y = erase(y));
        while ((y = x) != begin() && (--x)->p >= y->
    p)
            isect(x, erase(y));
    }
    ll query(ll x) {
        assert(!empty());
        QTYPE=1; auto l = *lower_bound({0,0,x});
    QTYPE = 0;
        return l.m * x + l.b;
    }
};
```

2 Math

2.1 rho

```
#include <bits/stdc++.h>
using namespace std;
#define ll long long
#define pii pair<int, int>
#define ull unsigned ll
#define f first
#define s second
#define FOR(i,a,b) for (int i=(a); i<(b); i++)
#define REP(i,n) for (int i=0; i<(n); i++)
#define RREP(i,n) for (int i=(n-1); i>=0; i--)
#define ALL(x) x.begin(),x.end()
#define SZ(x) (int)x.size()
#define SQ(x) (x)*(x)
#define MN(a,b) a = min(a,(__typeof__(a))(b))
#define MX(a,b) a = max(a,(__typeof__(a))(b))
#define pb push_back
#define SORT_UNIQUE(c) (sort(c.begin(),c.end()), c.
    resize(distance(c.begin(),unique(c.begin(),c.end
    ()))))
#ifdef BALBIT
#define IOS()
#define debug(...) do{\
    fprintf(stderr,"%s - %d (%s) = ",
    __PRETTY_FUNCTION__,__LINE__,#__VA_ARGS__);\
    _do(__VA_ARGS__);\
}while(0)
template<typename T>void _do(T &&_x){cerr<<_x<<endl
    ;}
template<typename T,typename ...S> void _do(T &&_x,S
    &&..._t){cerr<<_x<<" ,";_do(_t...);}
template<typename _a,typename _b> ostream& operator
    << (ostream &_s,const pair<_a,_b> &_p){return _s
    <<"("<<_p.X<<","<<_p.Y<<")";}
template<typename It> ostream& _OUTC(ostream &_s,It
    _ita,It _itb)
{
    _s<<"{";
    for(It _it=_ita;_it!=_itb;_it++)
    {
        _s<<(_it==_ita?"":",")<<*_it;
    }
    _s<<"}";
    return _s;
}
template<typename _a> ostream &operator << (ostream
    &_s,vector<_a> &_c){return _OUTC(_s,ALL(_c));}
template<typename _a> ostream &operator << (ostream
    &_s,set<_a> &_c){return _OUTC(_s,ALL(_c));}
template<typename _a> ostream &operator << (ostream
    &_s,deque<_a> &_c){return _OUTC(_s,ALL(_c));}
template<typename _a,typename _b> ostream &operator
    << (ostream &_s,map<_a,_b> &_c){return _OUTC(_s,
    ALL(_c));}
template<typename _t> void pary(_t _a,_t _b){_OUTC(
    cerr,_a,_b);cerr<<endl;}
#else
```

```
#define IOS() ios_base::sync_with_stdio(0);cin.tie
    (0);
#define endl '\n'
#define debug(...)
#define pary(...)
#endif

// #define int ll

const int iinf = 1<<29;
const ll inf = 1ll<<60;
const ll mod = 1e9+7;


void GG(){cout<<"-1\n"; exit(0);}

ll mpow(ll a, ll n, ll mo = mod){ // a^n % mod
    ll re=1;
    while (n>0){
        if (n&1) re = re*a %mo;
        a = a*a %mo;
        n>>=1;
    }
    return re;
}

ll inv (ll b, ll mo = mod){
    if (b==1) return b;
    return (mo-mo/b) * inv(mo%b) % mo;
}

const int maxn = 1e5+5;

#define lll __int128

lll c = 1;
lll g(lll x, lll n){
    return (x*x+c)%n;
}

lll gcd(lll a, lll b){
    if (b==0) return a;
    return gcd(b,a%b);
}

lll po(lll n){
    lll x = 2, y = 2, d = 1;
    while (d==1){
        x = g(x,n); y = g(g(y,n),n);
        d = gcd(x>y?x-y:y-x,n);
    }
    if (d==n) return -1;
    return d;
}

ll fac(ll n){
    if (n%2==0) return 2;
    lll ans = -1;
    for (int i = 0; i<5 && ans==-1; i++) {
        c++; if (c==2) c++;
        ans = po(n);
    }
    return ans;
}

main(){
    ll test = 1709049187;
    lll moo = test;
    ll ans = fac(moo);
    cout<<ans<<endl;

}
```

2.2 inversion

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
const ll mod = 10000007;

ll inv (ll b, ll mo = mod){
    if (b==1) return b;
    return (mo-mo/b) * inv(mo%b) % mo;
}

void extGCD(ll A,ll B,ll &x,ll &y) { // A p coprime
    if (B == 0) {
        x = 1;
        y = 0;
        assert(A == 1);
        return;
    }
    ll xx,yy;
    extGCD(B,A%B,xx,yy);
    x = yy;
    y = xx - A/B*yy;
    return;
}

ll ext_inv (ll a, ll p) { // a, p co-prime
    ll x, y;
    extGCD(a,p, x, y);
    x %= p;
    if (x < 0) {
        x += p;
    }
    assert(a * x % p);
    return x;
}
int main () {
    ll a, p;
    cin >> a >> p;
    ll ainv = ext_inv(a, p);
    cout << ainv << endl;
}
```

2.3 LL-Multiplication

```
ll mul1(ll a, ll b, ll n){
    __int128 x = a, y = b;
    return (ll)(x*y%n);
} // A little faster than mul2

ll mul2(ll a,ll b,ll n){
    a%=n,b%=n;
    ll y=(ll)((long double)a*b/n+0.5);
    ll r=(a*b-y*n)%n;
    return r<0?r+n:r;
}
```

2.4 CRT

```
ll mod;
ll mul(ll v1,ll v2,ll md=mod) {
    return v1 * v2 % md;
}

void normal(ll &v1) {
    v1 %= mod;
    if (v1 < 0) {
        v1 += mod;
    }
}

ll extGCD(ll n1,ll n2,ll &x1,ll &x2) {
    if (n1 == 0) {
        x2 = 1;
        x1 = 0;
        return n2;
    }
    ll cx1,cx2;
    ll ret = extGCD(n2%n1,n1,cx1,cx2);
    x2 = cx1;
    x1 = cx2 - n2/n1*cx1;
    return ret;
}

void crt (ll a, ll n, ll b, ll m) {
    ll r1,r2;
    ll gcd = extGCD(n,m,r1,r2);
    if ((b-a) % gcd != 0) {
        cout << "no solution" << endl;
        return;
    }
    mod = n * m / gcd;

    ll ans = mul(mul(r1,(b-a)/gcd,m/gcd),n) + a;
    normal(ans);
    cout << ans << " " << mod <<  endl;
}
```

2.5 geometry

```
const double PI = acos(-1);

struct Point{
  double x, y;

  bool operator < (const Point &b) const {
    return tie(x,y) < tie(b.x,b.y);
    //return atan2(y,x) < atan2(b.y,b.x);
  }
  Point operator + (const Point &b) const {
    return {x+b.x,y+b.y};
  }
  Point operator - (const Point &b) const {
    return {x-b.x,y-b.y};
  }
  Point operator * (const double d) const {
    return {x*d,y*d};
  }
  Point operator / (const double d) const {
    return {x/d,y/d};
  }
  double operator * (const Point &b) const {
    return x*b.x + y*b.y;
  }
  double operator % (const Point &b) const { //
    Cross!
    return x*b.y - y*b.x;
  }
  Point(double xx, double yy): x(xx), y(yy){  }
};

double Length( const Point &p ){
  return sqrt( p.x*p.x + p.y*p.y );
}

int ori(const Point &a, const Point &b, const Point
    &c){
  int tmp = (c-a)%(b-a);
  if (tmp==0) return 0; //Collinear
  return tmp>0? 1: -1;
}

bool collinear(const Point &a, const Point &b, const
     Point &c){
  return ori(a, b, c)==0;
}

bool btw(const Point &a, const Point &b, const Point
    &c){
  return(a-c)*(b-c)<=0;
}

typedef Point Vector;

double Angle( const Vector &a, const Vector &b ){
  double A = Length(a);
  double B = Length(b);
  double v = a*b;
  double theta = acos( v/A/B );
  return theta;
}

Vector rot(Vector vec, double a){
  return Vector(cos(a)*vec.x-sin(a)*vec.y, sin(a)*
    vec.x+cos(a)*vec.y);
}

Vector Normal(const Vector &v){
  return v / Length(v);
}

Point intersect_at(const Point &p, const Vector &v,
    const Point &q, const Vector &w){
  Vector u = q-p;
  return p+v*(u%w)/(q%w);
}

bool cmp(const Point&a, const Point &b){
  return a<b;
  //Sort by x first, then by y.
}

vector<Point> convex_hull(vector<Point>arr){
  sort (arr.begin(), arr.end(), cmp);
  vector<Point> p;
  int m = 0; // size of p
  for (int i=0; i<arr.size(); i++){ // Lower hull
    //cout<<"On the "<<i<<"-th one. "<<arr[i].x<<'
    '<<arr[i].y<<'\n';
    while (m>=2&&(p[m-1]-p[m-2])%(arr[i]-p[m-2])<0){
      //Get rid of a previous point
      //cout<<"Got rid of "<<p[m-1].x<<' '<<p[m-1].y
    <<'\n';
      p.pop_back(); m--;
    }
    p.push_back(arr[i]); m++;
  }
  //cout<<"Onto upper hull"<<'\n';
  int tmp = m+1; //the size of lower hull +1
  for (int i=arr.size()-2; i>=0; i--){
    //cout<<"On the "<<i<<"-th one. "<<arr[i].x<<'
    '<<arr[i].y<<'\n';
    while (m>=tmp&&(p[m-1]-p[m-2])%(arr[i]-p[m-2])
    <0){
      //cout<<"Got rid of "<<p[m-1].x<<' '<<p[m-1].y
    <<'\n';
      p.pop_back(); m--;
    }
    p.push_back(arr[i]); m++;
  }
  //cout<<m<<'\n';
  if (arr.size()>1) p.pop_back(); //Repeated
  return p;
}

//Segment banana

double signedArea(Point p[], int n){
  double re = 0.0;
  for (int i=0; i<n; i++){
    re+=p[i]%p[(i+1)%n];
  }
  return re/2.0; //Cross returns twice the triangle'
    s area
}

bool intersect(const Point a, const Point b, const
    Point c, const Point d){
  int abc = ori(a, b, c);
  int abd = ori(a, b, d);
  int cda = ori(c, d, a);
  int cdb = ori(c, d, b);
  if (abc==0&&abd==0){
    return btw(a,b,c)||btw(a,b,d)||btw(c,d,a)||btw(c
    ,d,b);
  }else return (abc*abd<=0&&cda*cdb<=0);
}
```

2.6 FWT

```
// from https://github.com/edisonhello/
    waynedisonitau123
void xorfwt(int v[], int l, int r) {
    if (r - l == 1) return;
    int m = l + r >> 1;
    xorfwt(v, l, m), xorfwt(v, m, r);
    for (int i = l, j = m; i < m; ++i, ++j) {
        int x = v[i] + v[j];
        v[j] = v[i] - v[j], v[i] = x;
    }
}

void xorifwt(int v[], int l, int r) {
    if (r - l == 1) return;
    int m = l + r >> 1;
    for (int i = l, j = m; i < m; ++i, ++j) {
        int x = (v[i] + v[j]) / 2;
        v[j] = (v[i] - v[j]) / 2, v[i] = x;
    }
    xorifwt(v, l, m), xorifwt(v, m, r);
}

void andfwt(int v[], int l, int r) {
    if (r - l == 1) return;
    int m = l + r >> 1;
    andfwt(v, l, m), andfwt(v, m, r);
    for (int i = l, j = m; i < m; ++i, ++j) v[i] +=
    v[j];
}

void andifwt(int v[], int l, int r) {
    if (r - l == 1) return;
    int m = l + r >> 1;
    andifwt(v, l, m), andifwt(v, m, r);
    for (int i = l, j = m; i < m; ++i, ++j) v[i] -=
    v[j];
}

void orfwt(int v[], int l, int r) {
    if (r - l == 1) return;
    int m = l + r >> 1;
    orfwt(v, l, m), orfwt(v, m, r);
    for (int i = l, j = m; i < m; ++i, ++j) v[j] +=
    v[i];
}

void orifwt(int v[], int l, int r) {
    if (r - l == 1) return;
    int m = l + r >> 1;
    orifwt(v, l, m), orifwt(v, m, r);
    for (int i = l, j = m; i < m; ++i, ++j) v[j] -=
    v[i];
}
```

2.7 FFT-precision

```
#include <bits/stdc++.h>
using namespace std;
#define SZ(v) int(v.size())
#define REP(i,n) for(int i=0;i<n;i++)
#define REP1(i,n) for(int i=1;i<=n;i++)

const int MAXN = 1<<20;
typedef complex<double> cd;

const double pi = acos(-1);
vector<int> bs;
cd omg[MAXN+3];

void FFT (vector<cd> &v, int d) {
    for (int i=1,j=SZ(v)>>1; i<SZ(v)-1; i++) {
        if (i < j) {
            swap(v[i], v[j]);
        }
        int k = SZ(v)>>1;
        while (k <= j) {
            j -= k;
            k >>= 1;
        }
        if (k > j) {
            j += k;
        }
    }

    for (int h=2; h<=SZ(v); h<<=1) {
        for (int i=0; i<SZ(v); i+=h) {
            for (int k=i; k<i+h/2; k++) {
                int idx = k-i;
                int r = k+h/2;
                cd x = v[k] - omg[d > 0 ? idx*(MAXN/
h) : MAXN-idx*(MAXN/h)] * v[r];
                v[k] = v[k] + omg[d > 0 ? idx*(MAXN/
h) : MAXN-idx*(MAXN/h)] * v[r];
                v[r] = x;
            }
        }
    }

    if (d < 0) {
        REP (i, SZ(v)) {
            v[i] /= SZ(v);
        }
    }
}

void build_omg() {
    omg[0] = omg[MAXN] = 1;
    REP1 (i, MAXN-1) {
        omg[i] = polar(1.0, i*pi*2/MAXN);
    }
}

vector<int> mul (vector<int> &v1, vector<int> &v2) {
    int n = 1;
    while (n < SZ(v1) + SZ(v2)) {
        n <<= 1;
    }
    vector<cd> x(n), y(n);
    REP (i, SZ(v1)) {
        x[i] = v1[i];
    }
    REP (i, SZ(v2)) {
        y[i] = v2[i];
    }
    FFT(x, 1);
    FFT(y, 1);
    REP (i, n) {
        x[i] *= y[i];
    }
    FFT(x, -1);
    vector<int> ret(n);
    REP (i, n) {
        ret[i] = min(1, (int)round(x[i].real()));
    }
    while (SZ(ret)>1 && ret.back() == 0) {
        ret.pop_back();
    }
    return ret;
}

int main () {

}
```

2.8 FFT

```
const double PI = acos(-1.0);
#define cd complex<double>

void FFT(vector<cd> &a, bool rev=0){
    int n = SZ(a);
    for (int i = 1, j = 0; i<n; i++){
        int bit = n>>1;
        while (j>=bit) j-=bit, bit>>=1; j+=bit;
        if (i<j) swap(a[i], a[j]);
    }
    for (int B = 2; B<=n; B*=2){
        double ang = 2 * PI / B * (rev?-1:1);
        cd w0 (cos(ang), sin(ang));
        for (int i = 0; i<n; i+=B){
            cd w (1,0);
            for (int j = 0; j<B/2; j++){
                cd u = a[i+j], v = w*a[i+j+B/2];
                a[i+j] = u+v, a[i+j+B/2] = u-v;
                w *= w0;
            }
        }
    }
    if (rev) REP(i,n) a[i] /= n;
}

vector<ll> mul (vector<ll> a, vector<ll> b){
    int n = 1; while (n < SZ(a) + SZ(b)) n*=2;
    vector<cd> x(n), y(n);
    REP(i, SZ(a)) x[i] = cd(a[i],0); REP(j, SZ(b)) y
    [j] = cd(b[j],0);
    FFT(x); FFT(y);
    REP(i, n) x[i] *= y[i];
    FFT(x,1);
    vector<ll> re(n);
    REP(i,n) re[i] = min((ll)(round(x[i].real())),1
    ll);
    while (re.size()>1 && re.back()==0) re.pop_back
    (); return re;
}
```

2.9 linear sieve

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

const int MAXC = 1000006;
bool sieve[MAXC];
vector<int> prime;

void linear_sieve() {
    for (int i=2; i<MAXC; i++) {
        if (!sieve[i]) prime.emplace_back(i);
        for (int j=0; i*prime[j]<MAXC; j++) {
            sieve[i*prime[j]] = true;
            if (i % prime[j] == 0) {
                break;
            }
        }
    }
}

int main () {
    linear_sieve();
    for (int i=0; i<20; i++) {
        cout << prime[i] << " \n"[i==19];
    }
}
```

2.10 NTT

```
void NTT(vector<ll> &a, ll mo, bool rev=0){
    // mo has to be 2^k * c + 1
    int n = SZ(a);
    while ((n&(-n))!=n) {
        a.pb(0); n++;
    }
    for (int i = 1, j = 0; i<n; i++){
        int bit = n>>1;
        while (j>=bit) j-=bit, bit>>=1; j+=bit;
        if (i<j) swap(a[i], a[j]);
    }
    for (int B = 2; B<=n; B*=2){
        ll w0 = mpow(3,(mo-1)/(B),mo);
        for (int i = 0; i<n; i+=B){
            ll w = 1;
            for (int j = 0; j<B/2; j++){
                ll u = a[i+j], v = w*a[i+j+B/2]%mo;
                a[i+j] = u+v, a[i+j+B/2] = u-v;
                if (a[i+j]>=mo) a[i+j]-=mo; if (a[i+
j+B/2]<0) a[i+j+B/2]+=mo;
                w = w*w0%mod;
            }
        }
    }
    if (rev) {
        reverse(next(a.begin()),a.end());
        ll invn = inv(n,mo);
        REP(i,n) a[i] = a[i]*invn%mod;
    }
}

vector<ll> mul (vector<ll> a, vector<ll> b, ll mo =
    mod){
    int n = 1; while (n < SZ(a) + SZ(b)) n*=2;
    vector<ll> x(n), y(n);
    REP(i, SZ(a)) x[i] = a[i]; REP(j, SZ(b)) y[j] =
    b[j];
    NTT(x,mo); NTT(y,mo);
    REP(i, n) x[i] = x[i] * y[i] % mo;
    NTT(x,mo,1);
    while (x.size()>1 && x.back()==0) x.pop_back();
    return x;
}
```

2.11 miller rabin

```
ll mul1(ll a, ll b, ll n){ // Better
    __int128 x = a, y = b;
    return (ll)(x*y%n);
}

ll mul2(ll a,ll b,ll n){ // Slightly worse
    a%=n,b%=n;
    ll y=(ll)((long double)a*b/n+0.5);
    ll r=(a*b-y*n)%n;
    return r<0?r+n:r;
}

ll mpow(ll a,ll b,ll mod){//a^b%mod
    ll ans=1;
    for(;b;a=mul1(a,a,mod),b>>=1)
        if(b&1)ans=mul1(ans,a,mod);
    return ans;
}
int sprp[3]={2,7,61};//int
int llsprp
    [7]={2,325,9375,28178,450775,9780504,1795265022};
    //unsinged long long

bool isprime(ll n){
    if(n==2)return 1;
    if(n<2||n%2==0)return 0;
    int t=0;
    ll u=n-1;
    for(;u%2==0;++t)u>>=1;
    for(int i=0;i<5;++i){ // Increase for more
    accuracy
        ll a=llsprp[i]%n;
        if(a==0||a==1||a==n-1)continue;
        ll x=mpow(a,u,n);
        if(x==1||x==n-1)continue;
        for(int j=1;j<t;++j){
            x=mul1(x,x,n);
            if(x==1)return 0;
            if(x==n-1)break;
        }
        if(x==n-1)continue;
        return 0;
    }
    return 1;
}
```

3 String

3.1 ac automation

```
const int K = 26,MAXN = 100005;;
struct Trie {
    int nxt[K], go[K], pid, pch, leaf = -1, link =
    -1, lst = -1;
    Trie (int _pid=0, int _pch=0) {
        memset(nxt, -1, sizeof(nxt));
        memset(go, -1, sizeof(go));
        pid = _pid;
        pch = _pch;
    }
};
vector<Trie> trie(1);
vector<int> occ[MAXN];

void addString (string &str, int id) {
    int nd = 0;
    for (auto c : str) {
        int cid = c - 'a';
        if (trie[nd].nxt[cid] == -1) {
            trie[nd].nxt[cid] = SZ(trie);
            trie.emplace_back(nd, cid);
        }
        nd = trie[nd].nxt[cid];
    }
    trie[nd].leaf = id;
}

int go (int nd, int cid);

int getLink (int nd) {
    if (trie[nd].link == -1) {
        if (nd == 0 || trie[nd].pid == 0) {
            trie[nd].link = 0;
        } else {
            trie[nd].link = go(getLink(trie[nd].pid)
    , trie[nd].pch);
        }
    }
    return trie[nd].link;
}

int getLast (int nd) {
    if (trie[nd].lst == -1) {
        if (trie[getLink(nd)].leaf == -1) {
            trie[nd].lst = nd == 0 ? 0 : getLast(
    getLink(nd));
        } else {
            trie[nd].lst = getLink(nd);
        }
    }
    return trie[nd].lst;
}

int go (int nd, int cid) {
    if (trie[nd].go[cid] == -1) {
        if (trie[nd].nxt[cid] != -1) {
            trie[nd].go[cid] = trie[nd].nxt[cid];
        } else {
            trie[nd].go[cid] = nd == 0 ? 0 : go(
    getLink(nd), cid);
        }
    }
    return trie[nd].go[cid];
}

void query (string &str) {
    int nd = 0;
    int sid = 0;
    for (auto c : str) {
        int cid = c - 'a';
        nd = go(nd, cid);

        int ptr = nd;
        while (ptr != 0) {
            if (trie[ptr].leaf != -1) {
                occ[trie[ptr].leaf].emplace_back(sid
    );
            }
            ptr = getLast(ptr);
            }

        sid++;
    }
}
```

3.2 zvalue

```
#include <bits/stdc++.h>
using namespace std;
const int MAXN = 2000006;

int z[MAXN];
string a;
void init(string x) {
    a = x;
    std::memset(z, 0, sizeof z);
}
void z_build() {
    z[0] = 0;
    for (int i = 1, bst = 0; a[i]; i++) {
        if (bst + z[bst] < i) {
            z[i] = 0;
        } else {
            z[i] = min(z[i - bst], bst + z[bst] - i)
    ;
        }
        while (a[z[i]] == a[z[i] + i]) {
            z[i]++;
        }
        if (i + z[i] > bst + z[bst]) {
            bst = i;
        }
    }
}

int mat(string x,string y) {
    int ret = 0;
    init(x+'$'+y);
    z_build();
    for (int i=int(x.size())+1;i<=int(x.size()+y.
    size());i++) {
        ret +=  (z[i] == int(x.size()));
    }
    return ret;
}

int main () {
    string a, b;
    cout << mat(a, b) << endl;
}
```

3.3 kmp

```
int app(string s, string t){ // Returns number of
    times s appears in t
  int n = s.length(), m = t.length();
  if (n>m) return 0;
  vector<int> f(n); f[0]=-1;
  for (int i = 1; i<n; i++){
    f[i] = f[i-1];
    while (f[i]!=-1 && s[f[i]+1]!=s[i]) f[i] = f[f[i
    ]];
    if (s[f[i]+1]==s[i]) f[i]++;
  }
  int j = 0, re = 0;
  for (int i = 0; i<m; i++){
    if (t[i] == s[j]) j++;
    else if (j) j = f[j-1]+1, i--;
    if (j==n) re++, j = f[j-1]+1;
  }
  return re;
}
```

3.4 DC3

```
// from https://github.com/edisonhello/
    waynedisonitau123
namespace DC3{
```

```
#pragma GCC diagnostic push
#pragma GCC diagnostic ignored "-Wsign-compare"

#define SG(v,i) ((i)>=int(v.size())?0:v[i])
    inline bool smaller(int a, int b, vector<int> &r
    ){
        if(SG(r,a+0) != SG(r,b+0)) return SG(r,a+0)<
    SG(r,b+0);
        if(SG(r,a+1) != SG(r,b+1)) return SG(r,a+1)<
    SG(r,b+1);
        return SG(r,a+2)<SG(r,b+2);
    }

    int cc[100005];
    inline vector<int> sort(vector<int> &r, int o,
    vector<int> &ix, int m){
        vector<int> rt(ix.size());
        for(int z=0;z<o;++z) r.push_back(0);
        for(int i=0;i<=m;++i) cc[i] = 0;
        for(int i=0;i<ix.size();++i) ++cc[r[ix[i]+o
    ]];
        for(int i=0;i<=m;++i) cc[i+1] += cc[i];
        for(int i=ix.size()-1;i>=0;--i) rt[--cc[r[ix
    [i]+o]]] = ix[i];
        for(int z=0;z<o;++z) r.pop_back();
        return rt;
    }

    vector<int> dc3(vector<int> &v, int n, int m){
        int c1 = (n+1)/3;
        vector<int> i12;
        for(int i=0;i<n;++i){
            if(i%3==0)continue;
            i12.push_back(i);
        }
        i12 = sort(v, 2, i12, m);
        i12 = sort(v, 1, i12, m);
        i12 = sort(v, 0, i12, m);

        int nr = 1;
        vector<int> r12(i12.size());
#define GRI(x) ((x)/3 + ((x)%3==2?c1:0))
        r12[GRI(i12[0])] = 1;
        for(int i=1;i<i12.size();++i){
            if(smaller(i12[i-1], i12[i], v)) r12[GRI
    (i12[i])] = ++nr;
            else r12[GRI(i12[i])] = nr;
        }

#define GEI(x) ((x)<c1?(x)*3+1:(x-c1)*3+2)
        if(nr != i12.size()){
            i12 = dc3(r12, i12.size(), nr);

            for(int i=0;i<i12.size();++i) r12[i12[i
    ]] = i+1;
            for(int &i: i12) i = GEI(i);
        }

        vector<int> i0;
        if(n%3==1) i0.push_back(n-1);
        for(int i=0;i<i12.size();++i) if(i12[i]%3 ==
     1) i0.push_back(i12[i]-1);
        i0 = sort(v, 0, i0, m);

        vector<int> ret(v.size());
        int ptr12=0, ptr0=0, ptr=0;
        while(ptr12<i12.size() && ptr0<i0.size()){
            if(i12[ptr12]%3 == 1){
                if([&](int i, int j) -> bool{
                    if(SG(v,i) != SG(v,j)) return SG
    (v,i)<SG(v,j);
                    return SG(r12,GRI(i+1))<SG(r12,
    GRI(j+1));
                }(i12[ptr12], i0[ptr0]))ret[ptr++] =
     i12[ptr12++];
                else ret[ptr++] = i0[ptr0++];
            }
            else{
                if([&](int i, int j) -> bool{
                    if(SG(v,i+0) != SG(v,j+0))
    return SG(v,i+0)<SG(v,j+0);
                    if(SG(v,i+1) != SG(v,j+1))
    return SG(v,i+1)<SG(v,j+1);
                    return SG(r12,GRI(i+2))<SG(r12,
    GRI(j+2));
                }(i12[ptr12], i0[ptr0]))ret[ptr++] =
     i12[ptr12++];
                else ret[ptr++] = i0[ptr0++];
            }
        }
        while(ptr12<i12.size()) ret[ptr++] = i12[
    ptr12++];
        while(ptr0<i0.size()) ret[ptr++] = i0[ptr0
    ++];

        return ret;
    }
    vector<int> build(string str){
        vector<int> val(str.size()+1, 0);
        for(int i=0;i<str.size();++i) val[i] = str[i
    ];
        return dc3(val, val.size(), 255);
    }
#pragma GCC diagnostic pop
}
```

3.5 suffix array

```
struct SuffixArray {
    string s;
    ll n;
    vector<ll> sa,rk,hei,t;
    SuffixArray(string si): s(si),n(SZ(s)),sa(n),rk(
    n),hei(n),t(n) {
        REP (i,n) {
            rk[sa[i]=i] = s[i];
        }
        t[n-1] = -1;
        for (ll h=1;t[n-1] != n-1; h <<= 1) {
            auto cmp = [&](ll i,ll j) {
                if (rk[i] != rk[j]) {
                    return rk[i] < rk[j];
                } else {
                    return (i+h < n && j+h < n) ? (
    rk[i+h] < rk[j+h]) : (i > j);
                }
            };
            sort(ALL(sa),cmp);
            t[0] = 0;
            REP1 (i,n-1) {
                t[i] = t[i-1] + cmp(sa[i-1],sa[i]);
            }
            REP (i,n) {
                rk[sa[i]] = t[i];
            }
        }
        ll con = 0;
        REP (i,n) {
            if (rk[i] == 0) {
                hei[0] = con = 0;
            } else {
                if (con) {
                    con--;
                }
                while (s[i+con] == s[sa[rk[i]-1]+con
    ]) {

                    con++;
                }
                hei[rk[i]] = con;
            }
        }
    }
    ll operator [] (ll idx) {
        return sa[idx];
    }
};
```

4 Graph

4.1 clique

```
typedef vector<bitset<200>> vb;
struct Maxclique {
  double limit=0.025, pk=0;
  struct Vertex { int i, d=0; };
  typedef vector<Vertex> vv;
  vb e;
  vv V;
  vector<vi> C;
  vi qmax, q, S, old;
  void init(vv& r) {
    trav(v,r) v.d = 0;
    trav(v, r) trav(j, r) v.d += e[v.i][j.i];
    sort(all(r), [](auto a, auto b) { return a.d > b
    .d; });
    int mxD = r[0].d;
    rep(i,0,sz(r)) r[i].d = min(i, mxD) + 1;
  }
  void expand(vv& R, int lev = 1) {
    S[lev] += S[lev - 1] - old[lev];
    old[lev] = S[lev - 1];
    while (sz(R)) {
      if (sz(q) + R.back().d <= sz(qmax)) return;
      q.push_back(R.back().i);
      vv T;
      trav(v,R) if (e[R.back().i][v.i]) T.push_back
    ({v.i});
      if (sz(T)) {
        if (S[lev]++ / ++pk < limit) init(T);
        int j = 0, mxk = 1, mnk = max(sz(qmax) - sz(
   q) + 1, 1);
        C[1].clear(), C[2].clear();
        trav(v, T) {
          int k = 1;
          auto f = [&](int i) { return e[v.i][i]; };
          while (any_of(all(C[k]), f)) k++;
          if (k > mxk) mxk = k, C[mxk + 1].clear();
          if (k < mnk) T[j++].i = v.i;
          C[k].push_back(v.i);
        }
        if (j > 0) T[j - 1].d = 0;
        rep(k,mnk,mxk + 1) trav(i, C[k])
          T[j].i = i, T[j++].d = k;
        expand(T, lev + 1);
      } else if (sz(q) > sz(qmax)) qmax = q;
      q.pop_back(), R.pop_back();
    }
  }
  vi maxClique() { init(V), expand(V); return qmax;
    }
  Maxclique(vb conn) : e(conn), C(sz(e)+1), S(sz(C))
    , old(S) {
    rep(i,0,sz(e)) V.push_back({i});
  }
};
```

4.2 steiner

```
// http://sunmoon-template.blogspot.com/2017/04/
    steiner-tree-problem-in-graphs.html
// choose r nodes in n nodse
//answer is max(dp[(1<<r)-1][k]) k=0~n-1
//p is the terminal set
//O( n^3 + n*3^r + n^2*2^r )
#define REP(i,n) for(int i=0;i<(int)n;++i)
const int MAXN=30,MAXM=8;// 0-base
const int INF=0x3f3f3f3f;
int dp[1<<MAXM][MAXN];
int g[MAXN][MAXN];
void init(){memset(g,0x3f,sizeof(g));}
void add_edge(int u,int v,int w){
  g[u][v]=g[v][u]=min(g[v][u],w);
}
void steiner(int n,int r,int *p){
  REP(k,n)REP(i,n)REP(j,n)
    g[i][j]=min(g[i][j],g[i][k]+g[k][j]);
  REP(i,n)g[i][i]=0;
  REP(i,r)REP(j,n)dp[1<<i][j]=g[p[i]][j];
  for(int i=1;i<(1<<r);++i){
    if(!(i&(i-1)))continue;
    REP(j,n)dp[i][j]=INF;
    REP(j,n){
      int tmp=INF;
      for(int s=i&(i-1);s;s=i&(s-1))
        tmp=min(tmp,dp[s][j]+dp[i^s][j]);
      REP(k,n)dp[i][k]=min(dp[i][k],g[j][k]+tmp);
    }
  }
}
```

4.3 spfa

```
int spfa(vector<vector<pii> > &g){ // G contains
    pair<to, cost>
    int n = SZ(g);
    int s = 0, t = n-1; // Starting node, ending
    node
    queue<int> q ({s});
    vector<int> vis(n,0); // Don't use vector<bool>
    vector<int> dist(n,inf);
    fill(ALL(dist), inf); dist[s] = 0;
    while (!q.empty()){
        int v = q.front(); q.pop();
        vis[v] = 0;
        for (auto &xx : g[v]) {
            int u = xx.f, w = xx.s;
            if (dist[u] > dist[v] + w){
                dist[u] = dist[v] + w;
                if (!vis[u]){
                    q.push(u); vis[u] = 1;
                }
            }
        }
    }
    return dist[t];
}
```

4.4 global-min-cut

```
// from https://raw.githubusercontent.com/Jinkela-
    Xiao-Zuan-Feng-Mountaineer/Codebook/master/Graph
    /%E5%85%A8%E5%B1%80%E6%9C%80%E5%B0%8F%E5%89%B2.
    cpp
const int INF=0x3f3f3f3f;
template<typename T>
struct stoer_wagner{// 0-base
  static const int MAXN=150;
  T g[MAXN][MAXN],dis[MAXN];
  int nd[MAXN],n,s,t;
  void init(int _n){
    n=_n;
    for(int i=0;i<n;++i)
      for(int j=0;j<n;++j)g[i][j]=0;
  }
  void add_edge(int u,int v,T w){
    g[u][v]=g[v][u]+=w;
  }
  T min_cut(){
    T ans=INF;
    for(int i=0;i<n;++i)nd[i]=i;
    for(int ind,tn=n;tn>1;--tn){
      for(int i=1;i<tn;++i)dis[nd[i]]=0;
      for(int i=1;i<tn;++i){
        ind=i;
        for(int j=i;j<tn;++j){
          dis[nd[j]]+=g[nd[i-1]][nd[j]];
          if(dis[nd[ind]]<dis[nd[j]])ind=j;
        }
        swap(nd[ind],nd[i]);
      }
      if(ans>dis[nd[ind]])ans=dis[t=nd[ind]],s=nd[
    ind-1];
      for(int i=0;i<tn;++i)
```

```
        g[nd[ind-1]][nd[i]]=g[nd[i]][nd[ind-1]]+=g[
    nd[i]][nd[ind]];
    }
    return ans;
  }
};
```

4.5 centroid decomp

```
int n;
vector<vector<pii> > edge;
struct CentroidDecomp {
    struct Info {
        ll dis=0, sz=0, mi=0;
    };
    vector<Info> info;
    vector<int> dead, dep, anc, vis, sz;
    vector<vector<ll> > dis;

    CentroidDecomp () : info(n), dead(n), dep(n),
    anc(n), vis(n), sz(n) {
        int lgg = __lg(n) + 2;
        dis.resize(lgg, vector<ll>(n, 0));
        build(0, 0, -1);
    }

    int center (int nd) {
        vector<int> que = {nd};
        vis[nd] = true;
        int hd = 0;
        while (hd < SZ(que)) {
            int cur = que[hd++];
            for (auto e : edge[cur]) {
                if (!vis[e.X] && !dead[e.X]) {
                    que.eb(e.X);
                    vis[e.X] = true;
                }
            }
        }
        reverse(ALL(que));

        int cen = -1;
        for (int v : que) {
            sz[v] = 1;
            vis[v] = false;
            bool flag = true;
            for (auto e : edge[v]) {
                if (!dead[e.X] && !vis[e.X]) {
                    sz[v] += sz[e.X];
                    flag &= sz[e.X] * 2 <= SZ(que);
                }
            }
            flag &= sz[v] * 2 >= SZ(que);
            if (flag) cen = v;
        }
        return cen;
    }

    void build (int nd, int d, int rt) {
        int cen = center(nd);
        assert(cen != -1);
        dead[cen] = true;
        dep[cen] = d;
        anc[cen] = rt;

        vector<int> que = {cen};
        int hd = 0;
        while (hd < SZ(que)) {
            int cur = que[hd++];
            for (auto e : edge[cur]) {
                if (!vis[e.X] && !dead[e.X]) {
                    que.eb(e.X);
                    vis[e.X] = true;
                    dis[d][e.X] = dis[d][cur] + e.Y;
                }
            }
        }
        for (int v : que) vis[v] = false;
        for (auto e : edge[cen]) {
            if (!dead[e.X]) {
                build(e.X, d+1, cen);
            }
        }
    }

    void upd (int nd) {
        for (int x=nd; x!=-1; x=anc[x]) {
            info[x].dis += dis[dep[x]][nd];
            info[x].sz += 1;
            if (anc[x] != -1) info[x].mi += dis[dep[
    x]-1][nd];
        }
    }

    ll qry (int nd) {
        ll res = info[nd].dis;
        for (int x=nd; anc[x]!=-1; x=anc[x]) {
            res += dis[dep[x]-1][nd] * (info[anc[x
    ]].sz - info[x].sz);
            res += info[anc[x]].dis;
            res -= info[x].mi;
        }
        return res;
    }
};
```

4.6 lca

```
#include <bits/stdc++.h>
using namespace std;
const int MAXN = 15003;
const int MAXLG = __lg(MAXN) + 2;
int n,q,a,b;

int anc[MAXLG][MAXN];
int dep[MAXN];
vector<int> edge[MAXN];
void dfs(int nd,int par){
  anc[0][nd] = par;
  dep[nd] = dep[par] + 1;
  for(int v:edge[nd]){
    if(v!=par) dfs(v,nd);
  }
}
void build_lca(){
  for(int i=1;i<MAXLG;i++){
    for(int j=0;j<n;j++){
      anc[i][j] = anc[i-1][anc[i-1][j]];
    }
  }
}

int query(int u,int v){
  if(dep[u] < dep[v])swap(u,v);
  for(int i=MAXLG-1;i>=0;i--){
    if(dep[anc[i][u]] >= dep[v]) u = anc[i][u];
  }
  if(u==v)return u;

  for(int i=MAXLG-1;i>=0;i--){
    if(anc[i][u] != anc[i][v]) {
      u = anc[i][u];
      v = anc[i][v];
    }
  }

  return anc[0][u];
}
int main(){
  cin>>n>>q;
  for(int i=0;i<n-1;i++) cin>>a>>b,edge[a].
    emplace_back(b),edge[b].emplace_back(a);

  dfs(0,0);
  build_lca();
  for(int i=0;i<q;i++){
    cin>>a>>b;
    cout<<query(a,b)<<endl;
  }
}


// Doubling LCA
```

4.7 ap

```
/*
from: http://sunmoon-template.blogspot.com
*/
#include <bits/stdc++.h>
using namespace std;

const int MAXN = 100005;

std::vector<int> G[MAXN];// 1-base
std::vector<int> bcc[MAXN];
int low[MAXN],vis[MAXN],Time;
int bcc_id[MAXN],bcc_cnt;// 1-base
bool is_cut[MAXN];//bcc_id is ndef if is_cut
int st[MAXN],top;
void dfs(int u,int pa=-1){
  int v,child=0;
  low[u]=vis[u]=++Time;
  st[top++]=u;
  for(size_t i=0;i<G[u].size();++i){
    if(!vis[v=G[u][i]]){
      dfs(v,u),++child;
      low[u]=std::min(low[u],low[v]);
      if(vis[u]<=low[v]){
        is_cut[u]=1;
        bcc[++bcc_cnt].clear();
        int t;
        do{
          bcc_id[t=st[--top]]=bcc_cnt;
          bcc[bcc_cnt].push_back(t);
        }while(t!=v);
        bcc_id[u]=bcc_cnt;
        bcc[bcc_cnt].push_back(u);
      }
    }else if(vis[v]<vis[u]&&v!=pa)//reverse
      low[u]=std::min(low[u],vis[v]);
  }
  if(pa==-1&&child<2)is_cut[u]=0;//u for root
}
inline void bcc_init(int n){
  Time=bcc_cnt=top=0;
  for(int i=1;i<=n;++i){
    G[i].clear();
    vis[i]=0;
    is_cut[i]=0;
    bcc_id[i]=0;
  }
}

int main () {
    int n, m;
    cin >> n >> m;
    bcc_init(n);
    for (int i=0; i<m; i++) {
        int u, v;
        cin >> u >> v;
        G[u].emplace_back(v);
        G[v].emplace_back(u);
    }

    dfs(1);
    for (int i=1; i<=n; i++) {
        cout << (is_cut[i] ? -1 : bcc_id[i]) << " \n
    "[i==n];
    }
}
```

4.8 bridge

```
/*
from: http://sunmoon-template.blogspot.com
*/
#include <bits/stdc++.h>
using namespace std;

const int MAXN = 100005;
struct edge{
  int u,v;
  bool is_bridge;
  edge(int u=0,int v=0):u(u),v(v),is_bridge(0){}
};
std::vector<edge> E;
std::vector<int> G[MAXN];// 1-base
int low[MAXN],vis[MAXN],Time;
int bcc_id[MAXN],bridge_cnt,bcc_cnt;// 1-base
int st[MAXN],top;// for bcc
inline void add_edge(int u,int v){
  G[u].push_back(E.size());
  E.push_back(edge(u,v));
  G[v].push_back(E.size());
  E.push_back(edge(v,u));
}
void dfs(int u,int re=-1){// re is last edge
  int v;
  low[u]=vis[u]=++Time;
  st[top++]=u;
  for(size_t i=0;i<G[u].size();++i){
    int e=G[u][i];v=E[e].v;
    if(!vis[v]){
      dfs(v,e^1);//e^1 reverse
      low[u]=std::min(low[u],low[v]);
      if(vis[u]<low[v]){
        E[e].is_bridge=E[e^1].is_bridge=1;
        ++bridge_cnt;
      }
    }else if(vis[v]<vis[u]&&e!=re)
      low[u]=std::min(low[u],vis[v]);
  }
  if(vis[u]==low[u]){// build bcc
    ++bcc_cnt;// 1-base
    do bcc_id[v=st[--top]]=bcc_cnt;
    while(v!=u);
  }
}
inline void bcc_init(int n){
  Time=bcc_cnt=bridge_cnt=top=0;
  E.clear();
  for(int i=1;i<=n;++i){
    G[i].clear();
    vis[i]=0;
    bcc_id[i]=0;
  }
}

int main () {
    int n, m;
    cin >> n >> m;
    bcc_init(n);
    for (int i=0; i<m; i++) {
        int u, v;
        cin >> u >> v;
        add_edge(u, v);
    }

    dfs(1);
    for (int i=1; i<=n; i++) {
        cout << bcc_id[i] << " \n"[i==n];
    }
}
```

4.9 scc

4.10 dijkstra

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
typedef pair<int,int> pii;
#define REP(i,n) for(int i=0;i<n;i++)
```

```
#define REP1(i,n) for(int i=1;i<=n;i++)
#define X first
#define Y second
const int MAXN = 1000003;
const int INF = (int)0x3f3f3f3f;
int n,m,s,g,a,b,v;


int dis[MAXN];
bool vis[MAXN];
vector<pii> e[MAXN];

int dijkstra (int s, int t) {
    memset(dis,INF,(n+1)*4);
    memset(vis,0,(n+1)*4);

    dis[s] = 0;
    priority_queue<pii,vector<pii>,greater<pii>> pq;
    pq.emplace(0,s);
    REP(i,n){
      int found = -1;
      while(pq.size()&&vis[found=pq.top().Y])pq.pop
    ();
      if(found==-1)break;
      vis[found]=1;
      for(auto vp:e[found]){
        if(dis[vp.X]>dis[found]+vp.Y){
          dis[vp.X] = dis[found]+vp.Y;
          pq.emplace(dis[vp.X],vp.X);
        }
      }
    }
}

void add_edge (int f, int t, int w) {
    e[f].emplace_back(t, w);
}

int main(){
  ios_base::sync_with_stdio(0);cin.tie(0);
  while(cin>>n>>m>>s>>g){
    REP(i,m){
      cin>>a>>b>>v;
      add_edge(a, b, v);
    }

    cout<<(dis[g]==INF?-1:dis[g])<<'\n';
  }
}
```

4.11 DynamicConnectivity

```
template <typename pNode>
struct Emap {
    vector<map<int, pNode> > data;
    inline void init(int n) { data.resize(n); }
    inline pNode& operator[](const pii p) { return
    data[p.ff][p.ss]; }
    inline bool has(const pii p) { return data[p.ff
    ].find(p.ss) != data[p.ff].end(); }
    inline void erase(const pii p) { data[p.ff].
    erase(p.ss); }
};

class ETT {
public:
    int n, llv;

private:
    struct Node;
    typedef Node* pNode;
    typedef pair<pNode, pNode> ppN;

    static int size(pNode p) { return p ? p->sz : 0;
     }

    struct Node {
        pii e;
        pNode L = NULL, R = NULL, P = NULL;
        int sz = 1;
        int llv;
        set<int> Edges[2];
        bool hasE[2] = {};
        inline bool isLeft() { return P && P->L ==
this; }
        inline bool isRight() { return P && P->R ==
this; }
        inline bool isRoot() { return !isLeft() && !
isRight(); }
        inline pNode& get(bool i) { return !i ? L :
R; } // 0 - LEFT
        inline pNode setCH(bool i, pNode ch) {
            ch->P = this;
            get(i) = ch;
            return this;
        }
        inline pNode getLast() { return R ? R->
getLast() : this; }
        inline pNode up() {
            sz = 1 + size(L) + size(R);
            for (int i = 0; i < 2; i++) {
                hasE[i] = !Edges[i].empty();
                if (L)
                    hasE[i] |= L->hasE[i];
                if (R)
                    hasE[i] |= R->hasE[i];
            }
            return this;
        }
        Node(pii p, int l) : e{ p }, llv{ l } {}
        pNode find_first(bool lt) {
            if (L && L->hasE[lt])
                return L->find_first(lt);
            if (!Edges[lt].empty())
                return this;
            return R->find_first(lt);
        }
        inline void insertEdge(bool lt, int d) {
            splay();
            Edges[lt].insert(d);
            up();
        }
        inline void eraseEdge(bool lt, int d) {
            splay();
            Edges[lt].erase(d);
            up();
        }
        inline void rotate(const bool dir) {
            pNode x = get(!dir);
            get(!dir) = x->get(dir);
            x->get(dir) = this;
            x->P = P;
            if (P) {
                if (P->L == this) P->L = x;
                if (P->R == this) P->R = x;
            }
            P = x;
            if (get(!dir))
                get(!dir)->P = this;
            up();
            x->up();
        }
        inline void rotateTop() { P->rotate(isLeft()
); }
        inline void splay(pNode rt = NULL) {
            while (P != rt) {
                if (P->P != rt)
                    ((P->isLeft() == isLeft()) ? P :
 this)->rotateTop();
                rotateTop();
            }
        }
    };
    pNode setFirst(pNode p) {
        if (!p) return p;
        p->splay();
        if (!p->R) swap(p->R, p->L); // , p->up();
        else if (p->L) {
            p->getLast()->splay(p);
            p->R->setCH(1, p->L)->up();
            p->L = NULL;
            p->up();
        }
        return p;
    }
    Emap<pNode> Epos;
    vector<Node> Ppos;
    inline bool onSameTree(pNode a, pNode b) {
        return a && b && (a == b || (a->splay(), b->
    splay(), a->P));
    }
    inline pNode create(pii e) {
        return Epos.has(e) ? Epos[e] : Epos[e] = new
     Node(e, llv);
    }

public:
    ETT(int _n, int lv) : n{_n}, llv{lv} {
        Epos.init(n);
        for (int i = 0; i < n; i++)
            Ppos.emplace_back(make_pair(i, i), llv);
    }
    inline bool onSameTree(int a, int b) { return
    onSameTree(&Ppos[a], &Ppos[b]); }
    inline bool hasEdge(pii e) { return Epos.has(e);
     }
    inline void link(pii p) {
        pNode l = setFirst(&Ppos[p.ff]), r =
    setFirst(&Ppos[p.ss]);
        create(swp(p))->setCH(0, create(p)->setCH(0,
     l)->setCH(1, r)->up())->up();
    }
    int cnt = 0;
    inline void link(int a, int b) { link({ a, b });
     }
    void cut(pii p) {
        if (!hasEdge(p))
            return;
        pNode fs = Epos[p], ls = Epos[swp(p)];
        setFirst(fs);
        if (fs->R)
            fs->R->P = NULL;
        fs->R = NULL;
        ls->splay();
        if (ls->L)
            ls->L->P = NULL;
        if (ls->R)
            ls->R->P = NULL;
        Epos.erase(p);
        Epos.erase(swp(p));
        delete fs;
        delete ls;
    }
    inline void cut(int a, int b) { cut({ a, b }); }
    inline int size(int a) { return Ppos[a].splay(),
     (Ppos[a].sz + 2) / 3; }
    inline void addEdge(bool lt, pii e) {
        Ppos[e.ff].insertEdge(lt, e.ss);
        Ppos[e.ss].insertEdge(lt, e.ff);
        if (lt)
            link(e);
    }
    inline void remEdge(bool lt, pii e, bool ct =
    true) {
        Ppos[e.ff].eraseEdge(lt, e.ss);
        Ppos[e.ss].eraseEdge(lt, e.ff);
        if (lt && ct)
            cut(e);
    }
    void forEach(bool lt, int start, function<bool(
    pii)> func) {
        Ppos[start].splay();
        for (pNode i = &Ppos[start]; i && i->hasE[lt
    ]; i = i->R) {
            (i = i->find_first(lt))->splay();
            for (auto j : i->Edges[lt])
                if (func({ i->e.ff, j }))
                    return;
            i->splay();
        }
    }
};

struct DyG {
    vector<ETT> ETTs;
    map<pair<int, int>, int> lvl;
    int lgn, n;
    inline int& level(pii p) { return lvl[norm(p)];
    }
    inline bool hasEdge(pii p) { return lvl.find(
    norm(p)) != lvl.end(); }
    inline void eraselvl(pii p) { lvl.erase(norm(p))
    ; }
    DyG(int _n) : n{n}, lgn{__lg(n)} {
        for (int i = 0; i <= lgn; i++)
            ETTs.emplace_back(n, i);
    }
    inline bool isConnected(int a, int b) { return
    ETTs[lgn].onSameTree(a, b); }
    inline void decrlvl(bool lt, pii e, int l = -1)
    {
        int& lv = level(e);
        ETTs[lv].remEdge(lt, e, false);
        ETTs[--lv].addEdge(lt, e);
    }
    inline void add(int a, int b) {
        if (hasEdge({ a, b })) return;
        level({ a, b }) = lgn;
        ETTs[lgn].addEdge(!isConnected(a, b), { a, b
     });
    }
    void remove(pii e) {
        if (!hasEdge(e)) return;
        int l = level(e);
        eraselvl(e);
        bool hasEdge = ETTs[lgn].hasEdge(e);
        ETTs[l].remEdge(hasEdge, e);
        if (!hasEdge) return;
        for (int i = l; i <= lgn; i++) {
            ETTs[i].cut(e);
            if (ETTs[i].size(e.ff) > ETTs[i].size(e.
    ss))
                e = { e.ss, e.ff };
            set<pii> tobe;
            ETTs[i].forEach(true, e.ff, [&](pii p) {
                tobe.insert(norm(p));
                return false;
            });
            for (auto p : tobe)
                decrlvl(true, p, i);
            tobe.clear();
            pii ans = { -1, -1 };

            ETTs[i].forEach(false, e.ff, [&](pii p)
    {
                if (ETTs[i].onSameTree(p.ss, e.ss))
                    return ans = p, true;
                tobe.insert(norm(p));
                return false;
            });

            for (auto p : tobe)
                decrlvl(false, p, i);
            if (ans != (pii) { -1, -1 }) {
                ETTs[i].remEdge(false, ans);
                ETTs[i].addEdge(true, ans);
                for (int lv = i + 1; lv <= lgn; lv
    ++)
                    ETTs[lv].cut(e), ETTs[lv].link(
    ans);
                return;
            }
        }
    }
    inline void remove(int a, int b) { remove({ a, b
     }); }
```

```
};
```

4.12 hld

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

const int MAXN = 10003;

struct edge{
  int u,v,w,n;
}e[MAXN*2];

int t,n,a,b,c;
int dep[MAXN],sz[MAXN],fat[MAXN],son[MAXN],top[MAXN
    ];
int in[MAXN],cnt,idx,head[MAXN];
int sg[MAXN*2];
char cmd[10];

void add_edge(int u,int v,int w){
  e[cnt].u = u;
  e[cnt].v = v;
  e[cnt].w = w;
  e[cnt].n = head[u];
  head[u] = cnt++;
}

void dfs1 (int nd,int par) {
  dep[nd] = dep[par] + 1;
  sz[nd] = 1;
  fat[nd] = par;
  son[nd] = 0;
  for (int i=head[nd];i!=-1;i=e[i].n) {
    if (e[i].v==par) continue;
    dfs1(e[i].v,nd);
    sz[nd] += sz[e[i].v];
    if(sz[e[i].v] > sz[son[nd]]) son[nd] = e[i].v;
  }
}

void dfs2 (int nd,int tp) {
  in[nd] = idx++;
  top[nd] = tp;
  if (son[nd]) dfs2(son[nd],tp);
  for (int i=head[nd];i!=-1;i=e[i].n) {
    if (e[i].v==fat[nd] || e[i].v==son[nd]) continue
    ;
    dfs2(e[i].v,e[i].v);
  }
}

int qpath (int x,int y) {
  int ret = 0;
  while (top[x] != top[y]) {
    if (dep[top[x]] < dep[top[y]]) swap(x,y);
    // ret = max(ret,query(in[top[x]],in[x]+1));
    x = fat[top[x]];
  }
  if(x==y)return ret;
  if (dep[x] < dep[y]) swap(x,y);
//   ret = max(ret,query(in[son[y]],in[x]+1));
  return ret;
}
```

5 FlowAndMatching

5.1 hlpp

```
// from https://www.lagou.com/lgeduarticle/82099.
    html
#include<bits/stdc++.h>
#define il inline
#define inc(i,j,k) for(int i=j;i<=k;++i)
#define ra(i,u) for(int i=head[u];i!=-1;i=a[i].nxt)
#define ll long long
#define inf 0x3f3f3f3f
using namespace std;
const int maxm=120010;
const int maxn=2010;
struct node
{
    int to,nxt,flow;
}a[maxm<<1];
int head[maxn],gap[maxn],h[maxn],e[maxn];
bool vis[maxn];
int cnt=-1,n,m,st,ed;
struct cmp {il bool operator () (int x,int y)const{
    return h[x]<h[y];}};
priority_queue<int,vector<int>,cmp> pq;
queue<int> q;
il void add(int u,int v,int w)
{
    a[++cnt].to=v;
    a[cnt].nxt=head[u];
    a[cnt].flow=w;
    head[u]=cnt;
}
il bool bfs()
{
    memset(h,inf,sizeof(h));
    h[ed]=0;
    q.push(ed);
    while(!q.empty())
    {
        int t=q.front();
        q.pop();
        ra(i,t)
        {
            int v=a[i].to;
            if(a[i^1].flow && h[v]>h[t]+1)
            {
                h[v]=h[t]+1;
                q.push(v);
            }
        }
    }
    return h[st]!=inf;
}
il void push(int u)
{
    ra(i,u)
    {
        int v=a[i].to;
        if((a[i].flow) && (h[v]+1==h[u]))
        {
            int df=min(e[u],a[i].flow);
            a[i].flow-=df;
            a[i^1].flow+=df;
            e[u]-=df;
            e[v]+=df;
            if((v!=st)&&(v!=ed)&&(!vis[v]))
            {
                pq.push(v);
                vis[v]=1;
            }
            if(!e[u])break;
        }
    }
}
il void relabel(int u)
{
    h[u]=inf;
    ra(i,u)
    {
        int v=a[i].to;
        if((a[i].flow)&&(h[v]+1<h[u]))h[u]=h[v]+1;
    }
}
inline int hlpp()
{
    if(!bfs())return 0;
    h[st]=n;
    memset(gap,0,sizeof(gap));
    inc(i,1,n) if(h[i]!=inf)gap[h[i]]++;
    ra(i,st)
    {
        int v=a[i].to;
        if(int f=a[i].flow)
        {
            a[i].flow-=f;a[i^1].flow+=f;
            e[st]-=f;e[v]+=f;
            if(v!=st&&v!=ed&&!vis[v])
            {
                pq.push(v);
                vis[v]=1;
            }
        }
    }
    while(!pq.empty())
    {
        int t=pq.top();pq.pop();
        vis[t]=0;push(t);
        if(e[t])
        {
            gap[h[t]]--;
            if(!gap[h[t]])
            {
                inc(v,1,n)
                {
                    if(v!=st&&v!=ed&&h[v]>h[t]&&h[v
    ]<n+1)
                    {
                        h[v]=n+1;
                    }
                }
            }
            relabel(t);gap[h[t]]++;
            pq.push(t);vis[t]=1;
        }
    }
    return e[ed];
}
signed main()
{
    memset(head,-1,sizeof(head));
    scanf("%d%d%d%d",&n,&m,&st,&ed);
    inc(i,1,m)
    {
        int x,y;
        ll f;
        scanf("%d%d%lld",&x,&y,&f);
        add(x,y,f);
        add(y,x,0);
    }
    ll maxf=hlpp();
    printf("%lld",maxf);
    return 0;
}
```

5.2 dinic

```
struct Dinic{
    struct Edge{
        int to, rev; ll cap, flow=0;
        Edge(int to,int rev, ll cap) : to(to), rev(
    rev), cap(cap) {}
    };

    vector<vector<Edge> > g;
    int n;
    int s, t;
    vector<int> level, ptr;
    Dinic(int n, int s, int t):n(n),s(s),t(t){
        level.resize(n,-1); ptr.resize(n); g.resize(
    n);
    }
    void add(int v, int u, ll cap){
        g[v].pb({u,SZ(g[u]),cap});
        g[u].pb({v,SZ(g[v])-1,0});
    }
    bool bfs(){ // Build layers with edges on the
    residual graph that aren't full
        queue<int> q({s});
        level[s] = 0;
        while (!q.empty() && level[t] == -1){
            int v = q.front(); q.pop();
            for (auto &e : g[v]){
                if (e.cap - e.flow ==0) continue;
                int u = e.to;
                if (level[u]==-1){
                    level[u] = level[v]+1; q.push(u)
;
                }
            }
        } return level[t]!=-1;
    }
    ll dfs(int v, ll amt){ // Returns flow amount of
     any flow on bfs graph
        if (amt == 0 || v==t) return amt;
        for (; ptr[v] <SZ(g[v]); ptr[v]++){
            Edge &e = g[v][ptr[v]];
            int u = e.to;
            if (level[u] == level[v]+1){
                ll tt = dfs(u,min(amt, e.cap - e.
flow));
                if (tt==0) continue;
                e.flow+=tt; g[e.to][e.rev].flow-=tt;
 return tt;
            }
        } return 0;
    }
    ll mf(){
        ll re = 0;
        while (bfs()){
            while (ll amt = dfs(s,inf)) re += amt;
// Basically ford fulkerson, but on layered
graph
            fill(ALL(level), -1); fill(ALL(ptr), 0);
        } return re;
    }
};

signed main(){
    int n = 100;
    int N = n+5; int s = N-1, t = N-2;
    Dinic dd (N,s,t);
    int mf = dd.mf();
}
```

5.3 km o3

```
// from http://sunmoon-template.blogspot.com
    /2016/05/kuhn-munkres-algorithm.html
#define MAXN 100
#define INF INT_MAX
int g[MAXN][MAXN],lx[MAXN],ly[MAXN],slack_y[MAXN];
int px[MAXN],py[MAXN],match_y[MAXN],par[MAXN];
int n;
void adjust(int y){
  match_y[y]=py[y];
  if(px[match_y[y]]!=-2)
    adjust(px[match_y[y]]);
}
bool dfs(int x){
  for(int y=0;y<n;++y){
    if(py[y]!=-1)continue;
    int t=lx[x]+ly[y]-g[x][y];
    if(t==0){
      py[y]=x;
      if(match_y[y]==-1){
        adjust(y);
        return 1;
      }
      if(px[match_y[y]]!=-1)continue;
      px[match_y[y]]=y;
      if(dfs(match_y[y]))return 1;
    }else if(slack_y[y]>t){
      slack_y[y]=t;
      par[y]=x;
    }
  }
  return 0;
}
inline int km(){
  memset(ly,0,sizeof(int)*n);
  memset(match_y,-1,sizeof(int)*n);
  for(int x=0;x<n;++x){
    lx[x]=-INF;
    for(int y=0;y<n;++y){
      lx[x]=max(lx[x],g[x][y]);
    }
  }
  for(int x=0;x<n;++x){
    for(int y=0;y<n;++y)slack_y[y]=INF;
    memset(px,-1,sizeof(int)*n);
    memset(py,-1,sizeof(int)*n);
    px[x]=-2;
    if(dfs(x))continue;
    bool flag=1;
    while(flag){
      int cut=INF;
      for(int y=0;y<n;++y)
        if(py[y]==-1&&cut>slack_y[y])cut=slack_y[y];
      for(int j=0;j<n;++j){
        if(px[j]!=-1)lx[j]-=cut;
        if(py[j]!=-1)ly[j]+=cut;
        else slack_y[j]-=cut;
      }
      for(int y=0;y<n;++y){
        if(py[y]==-1&&slack_y[y]==0){
          py[y]=par[y];
          if(match_y[y]==-1){
            adjust(y);
            flag=0;
            break;
          }
          px[match_y[y]]=y;
          if(dfs(match_y[y])){
            flag=0;
            break;
          }
        }
      }
    }
  }
  int ans=0;
  for(int y=0;y<n;++y)if(g[match_y[y]][y]!=-INF)ans
    +=g[match_y[y]][y];
  return ans;
}
```

5.4 bipartite matching

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

const int MAXN = 1003;
int mx[MAXN],my[MAXN];
bool vy[MAXN];
vector<int> edge[MAXN];

int n, m;
int greedy_matching()
{
    int c = 0;
    for (int x=1; x<=n; ++x) {
        if (mx[x] == -1) {
            for (auto y : edge[x]) {
                if (my[y] == -1) {
                        mx[x] = y; my[y] = x;
                        c++;
                        break;
                }
            }
        }
    }
    return c;
}

bool DFS(int x)
{
    for (auto y : edge[x]) {
        if (!vy[y]) {
            vy[y] = true;
            if (my[y] == -1 || DFS(my[y]))
            {
                mx[x] = y; my[y] = x;
                return true;
            }
        }
    }
    return false;
}

int bipartite_matching()
{
    memset(mx, -1, sizeof(mx));
    memset(my, -1, sizeof(my));

    int c = greedy_matching();

    for (int x=1; x<=n; ++x)
        if (mx[x] == -1)
        {
            memset(vy, false, sizeof(vy));
            if (DFS(x)) c++;
        }
    return c;
}


int main () {
    cin >> n >> m;
    int ecnt;
    cin >> ecnt;
    while (ecnt--) {
        int f,t;
        cin >> f >> t;
        edge[f].emplace_back(t);
    }

    cout << bipartite_matching() << endl;
}
```

5.5 LowerBoundFlow

```
// Determining solution for bounded flow system
    without source and sink
int n, m; cin>>n>>m;
vector<int> sumin(n,0), sumout(n,0);
int N = n+5; int SS = N-1, TT = N-2; // New source
    and new sink
Dinic dd(N,SS,TT); // Need to call Dinic
    implementation
ll totlow = 0;
REP(cnt, m){
    int a, b, l, u; cin>>a>>b>>l>>u; a--; b--; // l
    is lower bound, u is upper bound
    sumout[a] += l; sumin[b] += l;
    dd.add(a,b,u-l); totlow+=l;
}
// For bounded flow with source and sink, simply add
     edge from t to s with infinite capacity and do
    the same thing
REP(i,n){
    dd.add(SS,i,sumin[i]); dd.add(i,TT,sumout[i]);
}
ll f = dd.mf();
if (f == totlow)
    cout<<"YES\n";
else
    cout<<"NO\n";
```

5.6 matching

5.7 km o4

```
const int mxn = 100;

bool vx[mxn], vy[mxn]; // Visited x or y
int my[mxn]; // Match of y
```

```cpp
ll slk[mxn], lx[mxn], ly[mxn]; // Slack (on y),
    value on x, value on y
int g[mxn][mxn]; // Adjacency matrix with weights
int n;

bool dfs(int v){
    vx[v] = 1;
    REP(i,n){
        if (vy[i]) continue;
        if (g[v][i] == lx[v] + ly[i]) {
            vy[i] = 1;
            if (my[i]==-1 || dfs(my[i])){
                my[i] = v; return 1;
            }
        }else{
            MN(slk[i], lx[v] + ly[i] - g[v][i]);
        }
    }
    return 0;
}

ll mxmch(){
    REP(i,n) REP(j,n) MX(lx[i], g[i][j]);
    fill(my, my+n, -1);
    REP(i,n){
        while (1){
            fill(vx, vx+n, 0); fill(vy, vy+n, 0);
    fill(slk, slk+n, inf);
            if (dfs(i)) break;
            ll hv = *min_element(slk, slk+n);
            REP(i,n) if (vx[i]) lx[i] -= hv;
            REP(i,n) if (vy[i]) ly[i] += hv;
        }
    }
    ll re= 0;
    REP(i,n) re += g[my[i]][i];
    return re;
}
```

5.8 VKMV

```cpp
const int MX = 507;

ll a[MX][MX];

using T = ll;
T hungary(int n, int m) { // N is size of left set,
    M is size of right set
    vector<T> u(n + 1), v(m + 1);
    vector<int> p(m + 1), way(m + 1);
    for (int i = 1; i <= n; ++i) {
        p[0] = i;
        int j0 = 0;
        vector<T> minv (m + 1, INF);
        vector<char> used (m + 1, 0);
        while (p[j0] != 0) {
            used[j0] = 1;
            int i0 = p[j0], j1 = 0;
            T d = INF;
            for (int j = 1; j <= m; ++j)
                if (!used[j]) {
                    T cur = a[i0][j] - u[i0] - v[j];
                    if (cur < minv[j])
                        minv[j] = cur, way[j] = j0;
                    if (minv[j] < d)
                        d = minv[j], j1 = j;
                }
            for (int j = 0; j <= m; ++j)
                if (used[j])
                    u[p[j]] += d, v[j] -= d;
                else
                    minv[j] -= d;
            j0 = j1;
        }
        do {
            int j1 = way[j0];
            p[j0] = p[j1];
            j0 = j1;
        } while (j0);
    }

    vector<int> ans (n + 1);
    for (int j = 1; j <= m; ++j)
        ans[p[j]] = j;
    T cost = -v[0];
    return cost;
}
```

5.9 mcmf

```cpp
struct MCMF{
    int n, s, t;
    struct Edge{
        int to, rev;
        ll cost, cap, flow=0; // Can have negative
    flow!!!!!
        Edge(int to, int rev, ll cost, ll cap): to(
    to), rev(rev), cost(cost), cap(cap) {}
    };
    vector<int> par, id;
    vector<ll> dist;
    vector<vector<Edge> > g;
    MCMF(int n,int s,int t): n(n), s(s), t(t){
        par.resize(n); id.resize(n); dist.resize(n,
    inf);
        g.resize(n);
    }
    void add(int v, int u, ll f, ll c){
        g[v].pb({u,SZ(g[u]),c,f});
        g[u].pb({v,SZ(g[v])-1,-c,0});
    }
    bool spfa(){ // SPFA
        queue<int> q ({s});
        vector<int> vis(n,0);
        fill(ALL(dist), inf); dist[s] = 0;
        while (!q.empty()){
            int v = q.front(); q.pop();
            vis[v] = 0;
            for (int i = 0; i<SZ(g[v]); i++){
                Edge &e = g[v][i];
                if (e.cap - e.flow==0) continue;
                if (dist[e.to] > dist[v] + e.cost){
                    dist[e.to] = dist[v] + e.cost;
                    par[e.to] = v; id[e.to] = i;
                    if (!vis[e.to]){
                        q.push(e.to); vis[e.to] = 1;
                    }
                }
            }
        }
        return dist[t] != inf;
    }
    pair<ll, ll> mf(){
        pair<ll, ll> re = {0,0};
        while (spfa()){
            ll famt = inf;
            for (int v = t; v!=s; v = par[v]){
                Edge &e = g[par[v]][id[v]];
                MN(famt, e.cap - e.flow);
            }
            for (int v = t; v!=s; v = par[v]){
                Edge &e = g[par[v]][id[v]];
                e.flow += famt;
                g[e.to][e.rev].flow -= famt;
            }
            re.f += famt;
            re.s += dist[t] * famt;
        }
        return re;
    }
};
```

5.10 blossom

```cpp
// from sunmoon template
#define MAXN 505
vector<int>g[MAXN];
int pa[MAXN],match[MAXN],st[MAXN],S[MAXN],vis[MAXN];
int t,n;
inline int lca(int u,int v){
  for(++t;;swap(u,v)){
    if(u==0)continue;
    if(vis[u]==t)return u;
    vis[u]=t;
    u=st[pa[match[u]]];
  }
}
#define qpush(u) q.push(u),S[u]=0
inline void flower(int u,int v,int l,queue<int> &q){
  while(st[u]!=l){
    pa[u]=v;
    if(S[v=match[u]]==1)qpush(v);
    st[u]=st[v]=l,u=pa[v];
  }
}
inline bool bfs(int u){
  for(int i=1;i<=n;++i)st[i]=i;
  memset(S+1,-1,sizeof(int)*n);
  queue<int>q;qpush(u);
  while(q.size()){
    u=q.front(),q.pop();
    for(size_t i=0;i<g[u].size();++i){
      int v=g[u][i];
      if(S[v]==-1){
        pa[v]=u,S[v]=1;
        if(!match[v]){
          for(int lst;u;v=lst,u=pa[v])
            lst=match[u],match[u]=v,match[v]=u;
          return 1;
        }
        qpush(match[v]);
      }else if(!S[v]&&st[v]!=st[u]){
        int l=lca(st[v],st[u]);
        flower(v,u,l,q),flower(u,v,l,q);
      }
    }
  }
  return 0;
}
inline int blossom(){
  memset(pa+1,0,sizeof(int)*n);
  memset(match+1,0,sizeof(int)*n);
  int ans=0;
  for(int i=1;i<=n;++i)
    if(!match[i]&&bfs(i))++ans;
  return ans;
}
```

6 MISC

6.1 template

```
#include <bits/stdc++.h>
#pragma GCC optimize("unroll-loops,no-stack-
    protector")
using namespace std;
typedef long long ll;
typedef long long lld;
typedef long double ld;
typedef pair<int,int> pii;
typedef pair<ll,ll> pll;
typedef pair<ld,ld> pdd;
#define ALL(a) a.begin(),a.end()
#define all(a) (a).begin(), (a).end()
#define SZ(a) ((int)a.size())
#define F first
#define S second
#define ff first
#define ss second
#define REP(i,n) for(int i=0;i<((int)n);i++)
#define eb emplace_back
#define pb push_back
#define MP(a,b) make_pair(a,b)
#define SORT_UNIQUE(c) (sort(c.begin(),c.end()), c.
    resize(distance(c.begin(),unique(c.begin(),c.end
    ()))))
#define GET_POS(c,x) (int)(lower_bound(c.begin(),c.
    end(),x)-c.begin())
#define EL cout<<'\n'
#define BS(a,x) binary_search(ALL(a),x)

template<typename T> void _do(T &&x){cerr<<x<<endl;}
template<typename T, typename ...S> void _do(T &&x,
    S &&...y){cerr<<x<<", ";_do(y...);}
template<typename It> ostream& _printRng(ostream &os
    ,It bg,It ed)
{
    for(It it=bg;it!=ed;it++) {
        os<<(it==bg?"":" ")<<*it;
    }
    return os;
}
template<typename T1,typename T2>
ostream& operator<<(ostream& out,pair<T1,T2> P){
  out<<'('<<P.F<<", "<<P.S<<')';
  return out;
}
template<typename T> ostream &operator << (ostream &
    os,vector<T> &v){return _printRng(os,v.begin(),
    v.end());}
#ifdef uta
#define debug(...) fprintf(stderr,"#%d: %s = ",
    __LINE__,#__VA_ARGS__),_do(__VA_ARGS__);
#define IOS
#else
#define debug(...)
#define IOS ios_base::sync_with_stdio(0); cin.tie(0)
#define endl '\n'
#endif

const ll maxn=300005;
const ll maxlg=20;
const ll INF64=1e18;
const int INF=0x3f3f3f3f;
const ll MOD=ll(1e9+7);
const ld PI=acos(-1);
const ld eps=1e-9;
//const ll p=880301;
//const ll P=31;

ll mypow(ll a,ll b){
  ll res=1LL;
  while(b){
    if(b&1) res=res*a%MOD;
    a=a*a%MOD;
    b>>=1;
  }
  return res;
}


int main(){
  IOS;


  return 0;
}
```

6.2 raw string

```
#include <bits/stdc++.h>
using namespace std;
int main () {
    string str1 = R"(\"'"^&*()))";
    cout << str1 << endl;
}
```

6.3 pb ds

```
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace __gnu_pbds;
tree<int,null_type,less<int>, rb_tree_tag,
    tree_order_statistics_node_update> rk_tree;
```

6.4 Random

```
main(){
    IOS();
    mt19937 rng(chrono::steady_clock::now().
    time_since_epoch().count());
    // Basically the same as rand()
    vector<int> v(10); iota(ALL(v),1);
    shuffle(ALL(v), rng); // Use instead of
    random_shuffle
    for (int x : v) cout<<x<<' ';
    cout<<"Random number [0,100): "<<rng()%100<<endl
    ;
}
```

6.5 vimrc

```
set nu rnu bs=2 cin cul et sw=4 sts=4 ts=4 hls
syntax on
inoremap {<CR> {<CR>}<Esc>O
nnoremap <Space> :noh<CR>
```