JAW Codebook

Contents

1 Basic

1.1 vimrc

```
sy on
se sw=4 ts=4 sts=4 et nu sc hls
filet plugin indent on
no <F4> :vs input.txt<CR>
no <F5> :!./a.out<CR>
no <F6> :!./a.out < input.txt<CR>
no <F9> :g++ % -std=c++17 -O2 -g
    -fsanitize=undefined -Wall -Wextra -Wshadow
    -Wno-unused-result<CR>
no <expr> <silent> <Home> col('.') ==
    match(getline('.'),'\S') + 1 ? '0' : '^'
im <silent> <Home> <C-O><Home>
```

2 Data Structure

2.1 Disjoint Set

```
struct DisjointSet {
  // save() is like recursive
  // undo() is like return
  int n, fa[MXN], sz[MXN];
  vector<pair<int*,int>> h;
  vector<int> sp;
  void init(int tn) {
    n=tn;
    for (int i=0; i<n; i++) {
      fa[i]=i;
      sz[i]=1;
    }
    sp.clear(); h.clear();
  }
  void assign(int *k, int v) {
    h.PB({k, *k});
    *k=v;
  }
```

```cpp
    void save() { sp.PB(SZ(h)); }
    void undo() {
      assert(!sp.empty());
      int last=sp.back(); sp.pop_back();
      while (SZ(h)!=last) {
        auto x=h.back(); h.pop_back();
        *x.F=x.S;
      }
    }
    int f(int x) {
      while (fa[x]!=x) x=fa[x];
      return x;
    }
    void uni(int x, int y) {
      x=f(x); y=f(y);
      if (x==y) return ;
      if (sz[x]<sz[y]) swap(x, y);
      assign(&sz[x], sz[x]+sz[y]);
      assign(&fa[y], x);
    }
}djs;
```

2.2 Treap

```cpp
const int MEM = 16000004;
struct Treap {
  static Treap nil, mem[MEM], *pmem;
  Treap *l, *r;
  char val;
  int size;
  Treap () : l(&nil), r(&nil), size(0) {}
  Treap (char _val) :
    l(&nil), r(&nil), val(_val), size(1) {}
} Treap::nil, Treap::mem[MEM], *Treap::pmem =
    Treap::mem;


int size(const Treap *t) { return t->size; }
void pull(Treap *t) {
  if (!size(t)) return;
  t->size = size(t->l) + size(t->r) + 1;
}
Treap* merge(Treap *a, Treap *b) {
  if (!size(a)) return b;
  if (!size(b)) return a;
  Treap *t;
  if (rand() % (size(a) + size(b)) < size(a)) {
    t = new (Treap::pmem++) Treap(*a);
    t->r = merge(a->r, b);
  } else {
    t = new (Treap::pmem++) Treap(*b);
    t->l = merge(a, b->l);
  }
  pull(t);
  return t;
}
void split(Treap *t, int k, Treap *&a, Treap *&b) {
  if (!size(t)) a = b = &Treap::nil;
  else if (size(t->l) + 1 <= k) {
    a = new (Treap::pmem++) Treap(*t);
    split(t->r, k - size(t->l) - 1, a->r, b);
    pull(a);
  } else {
    b = new (Treap::pmem++) Treap(*t);
    split(t->l, k, a, b->l);
    pull(b);
  }
}


int nv;
Treap *rt[50005];

void print(const Treap *t) {
  if (!size(t)) return;
  print(t->l);
  cout << t->val;
  print(t->r);
}
```

```cpp
int main(int argc, char** argv) {
  IOS;
  rt[nv=0] = &Treap::nil;
  Treap::pmem = Treap::mem;
  int Q, cmd, p, c, v;
  string s;
  cin >> Q;
  while (Q--) {
    cin >> cmd;
    if (cmd == 1) {
      // insert string s after position p
      cin >> p >> s;
      Treap *tl, *tr;
      split(rt[nv], p, tl, tr);
      for (int i=0; i<SZ(s); i++)
        tl = merge(tl, new (Treap::pmem++)
            Treap(s[i]));
      rt[++nv] = merge(tl, tr);
    } else if (cmd == 2) {
      // remove c characters starting at position
      Treap *tl, *tm, *tr;
      cin >> p >> c;
      split(rt[nv], p-1, tl, tm);
      split(tm, c, tm, tr);
      rt[++nv] = merge(tl, tr);
    } else if (cmd == 3) {
      // print c characters starting at position p,
      //     in version v
      Treap *tl, *tm, *tr;
      cin >> v >> p >> c;
      split(rt[v], p-1, tl, tm);
      split(tm, c, tm, tr);
      print(tm);
      cout << "\n";
    }
  }
  return 0;
}
```

2.3 Heavy Light Decomposition

```cpp
// only one segment tree / 0-base
// should call init after input N
// getPathSeg return the segment in order u->v
// fa[root] = root

typedef pair<int,int> pii;

int N,fa[MXN],belong[MXN],dep[MXN],sz[MXN],que[MXN];
int step,line[MXN],stPt[MXN],edPt[MXN];
vector<int> E[MXN], chain[MXN];

void init() {
  REP(i,N) {
    E[i].clear();
    chain[i].clear();
  }
}
void DFS(int u){
  vector<int> &c = chain[belong[u]];
  for (int i=c.size()-1; i>=0; i--){
    int v = c[i];
    stPt[v] = step;
    line[step++] = v;
  }
  for (int i=0; i<(int)c.size(); i++){
    u = c[i];
    for (auto v : E[u]){
      if (fa[u] == v || (i && v == c[i-1])) continue;
      DFS(v);
    }
    edPt[u] = step-1;
  }
}
void build_chain(int st){
  int fr,bk;
```

2

```cpp
    fr=bk=0; que[bk++]=st; fa[st]=st; dep[st]=0;
    while (fr < bk){
      int u=que[fr++];
      for (auto v : E[u]){
        if (v == fa[u]) continue;
        que[bk++] = v;
        dep[v] = dep[u]+1;
        fa[v] = u;
      }
    }
    for (int i=bk-1,u,pos; i>=0; i--){
      u = que[i]; sz[u] = 1; pos = -1;
      for (auto v : E[u]){
        if (v == fa[u]) continue;
        sz[u] += sz[v];
        if (pos==-1 || sz[v]>sz[pos]) pos=v;
      }
      if (pos == -1) belong[u] = u;
      else belong[u] = belong[pos];
      chain[belong[u]].PB(u);
    }
    step = 0;
    DFS(st);
}
int getLCA(int u, int v){
    while (belong[u] != belong[v]){
      int a = chain[belong[u]].back();
      int b = chain[belong[v]].back();
      if (dep[a] > dep[b]) u = fa[a];
      else v = fa[b];
    }
    return sz[u] >= sz[v] ? u : v;
}
vector<pii> getPathSeg(int u, int v){
    vector<pii> ret1,ret2;
    while (belong[u] != belong[v]){
      int a = chain[belong[u]].back();
      int b = chain[belong[v]].back();
      if (dep[a] > dep[b]){
        ret1.PB({stPt[a],stPt[u]});
        u = fa[a];
      } else {
        ret2.PB({stPt[b],stPt[v]});
        v = fa[b];
      }
    }
    if (dep[u] > dep[v]) swap(u,v);
    ret1.PB({stPt[u],stPt[v]});
    reverse(ret2.begin(), ret2.end());
    ret1.insert(ret1.end(),ret2.begin(),ret2.end());
    return ret1;
}
// Usage
void build(){
    build_chain(0); //change root
    init(0,step,0); //init segment tree
}
int get_answer(int u, int v){
    int ret = -2147483647;
    vector<pii> vec = getPathSeg(u,v);
    for (auto it : vec)
      ; // check answer with segment [it.F, it.S]
    return ret;
}
```

2.4 Link-Cut Tree

```cpp
const int MXN = 100005;
const int MEM = 100005;

struct Splay {
    static Splay nil, mem[MEM], *pmem;
    Splay *ch[2], *f;
    int val, rev, size;
    Splay () : val(-1), rev(0), size(0) {
      f = ch[0] = ch[1] = &nil;
    }
```

```cpp
    Splay (int _val) : val(_val), rev(0), size(1) {
      f = ch[0] = ch[1] = &nil;
    }
    bool isr() {
      return f->ch[0] != this && f->ch[1] != this;
    }
    int dir() {
      return f->ch[0] == this ? 0 : 1;
    }
    void setCh(Splay *c, int d) {
      ch[d] = c;
      if (c != &nil) c->f = this;
      pull();
    }
    void push() {
      if (rev) {
        swap(ch[0], ch[1]);
        if (ch[0] != &nil) ch[0]->rev ^= 1;
        if (ch[1] != &nil) ch[1]->rev ^= 1;
        rev=0;
      }
    }
    void pull() {
      size = ch[0]->size + ch[1]->size + 1;
      if (ch[0] != &nil) ch[0]->f = this;
      if (ch[1] != &nil) ch[1]->f = this;
    }
} Splay::nil, Splay::mem[MEM], *Splay::pmem =
    Splay::mem;
Splay *nil = &Splay::nil;

void rotate(Splay *x) {
    Splay *p = x->f;
    int d = x->dir();
    if (!p->isr()) p->f->setCh(x, p->dir());
    else x->f = p->f;
    p->setCh(x->ch[!d], d);
    x->setCh(p, !d);
    p->pull(); x->pull();
}

vector<Splay*> splayVec;
void splay(Splay *x) {
    splayVec.clear();
    for (Splay *q=x;; q=q->f) {
      splayVec.push_back(q);
      if (q->isr()) break;
    }
    reverse(begin(splayVec), end(splayVec));
    for (auto it : splayVec) it->push();
    while (!x->isr()) {
      if (x->f->isr()) rotate(x);
      else if (x->dir()==x->f->dir())
          rotate(x->f),rotate(x);
      else rotate(x),rotate(x);
    }
}

Splay* access(Splay *x) {
    Splay *q = nil;
    for (;x!=nil;x=x->f) {
      splay(x);
      x->setCh(q, 1);
      q = x;
    }
    return q;
}
void evert(Splay *x) {
    access(x);
    splay(x);
    x->rev ^= 1;
    x->push(); x->pull();
}
void link(Splay *x, Splay *y) {
// evert(x);
    access(x);
    splay(x);
```

```
    evert(y);
    x->setCh(y, 1);
}
void cut(Splay *x, Splay *y) {
//  evert(x);
    access(y);
    splay(y);
    y->push();
    y->ch[0] = y->ch[0]->f = nil;
}

int N, Q;
Splay *vt[MXN];

int ask(Splay *x, Splay *y) {
    access(x);
    access(y);
    splay(x);
    int res = x->f->val;
    if (res == -1) res=x->val;
    return res;
}
int main(int argc, char** argv) {
    scanf("%d%d", &N, &Q);
    for (int i=1; i<=N; i++)
        vt[i] = new (Splay::pmem++) Splay(i);
    while (Q--) {
        char cmd[105];
        int u, v;
        scanf("%s", cmd);
        if (cmd[1] == 'i') {
            scanf("%d%d", &u, &v);
            link(vt[v], vt[u]);
        } else if (cmd[0] == 'c') {
            scanf("%d", &v);
            cut(vt[1], vt[v]);
        } else {
            scanf("%d%d", &u, &v);
            int res=ask(vt[u], vt[v]);
            printf("%d\n", res);
        }
    }

    return 0;
}
```

3 Graph

3.1 BCC Edge

```
struct BccEdge {
    static const int MXN = 100005;
    struct Edge { int v,eid; };
    int n,m,step,par[MXN],dfn[MXN],low[MXN];
    vector<Edge> E[MXN];
    DisjointSet djs;
    void init(int _n) {
        n = _n; m = 0;
        for (int i=0; i<n; i++) E[i].clear();
        djs.init(n);
    }
    void add_edge(int u, int v) {
        E[u].PB({v, m});
        E[v].PB({u, m});
        m++;
    }
    void DFS(int u, int f, int f_eid) {
        par[u] = f;
        dfn[u] = low[u] = step++;
        for (auto it:E[u]) {
            if (it.eid == f_eid) continue;
            int v = it.v;
            if (dfn[v] == -1) {
                DFS(v, u, it.eid);
                low[u] = min(low[u], low[v]);
            } else {
```

```
                low[u] = min(low[u], dfn[v]);
            }
        }
    }
    void solve() {
        step = 0;
        memset(dfn, -1, sizeof(int)*n);
        for (int i=0; i<n; i++) {
            if (dfn[i] == -1) DFS(i, i, -1);
        }
        djs.init(n);
        for (int i=0; i<n; i++) {
            if (low[i] < dfn[i]) djs.uni(i, par[i]);
        }
    }
}graph;
```

3.2 BCC Vertex

```
struct BccVertex {
    int n,nBcc,step,root,dfn[MXN],low[MXN];
    vector<int> E[MXN], ap;
    vector<pii> bcc[MXN];
    int top;
    pii stk[MXN];
    void init(int _n) {
        n = _n;
        nBcc = step = 0;
        for (int i=0; i<n; i++) E[i].clear();
    }
    void add_edge(int u, int v) {
        E[u].PB(v);
        E[v].PB(u);
    }
    void DFS(int u, int f) {
        dfn[u] = low[u] = step++;
        int son = 0;
        for (auto v:E[u]) {
            if (v == f) continue;
            if (dfn[v] == -1) {
                son++;
                stk[top++] = {u,v};
                DFS(v,u);
                if (low[v] >= dfn[u]) {
                    if(v != root) ap.PB(v);
                    do {
                        assert(top > 0);
                        bcc[nBcc].PB(stk[--top]);
                    } while (stk[top] != pii(u,v));
                    nBcc++;
                }
                low[u] = min(low[u], low[v]);
            } else {
                if (dfn[v] < dfn[u]) stk[top++] = pii(u,v);
                low[u] = min(low[u],dfn[v]);
            }
        }
        if (u == root && son > 1) ap.PB(u);
    }
    // return the edges of each bcc;
    vector<vector<pii>> solve() {
        vector<vector<pii>> res;
        for (int i=0; i<n; i++) {
            dfn[i] = low[i] = -1;
        }
        ap.clear();
        for (int i=0; i<n; i++) {
            if (dfn[i] == -1) {
                top = 0;
                root = i;
                DFS(i,i);
            }
        }
        REP(i,nBcc) res.PB(bcc[i]);
        return res;
    }
}graph;
```

3.3 Strongly Connected Components

```cpp
struct Scc{
  int n, nScc, vst[MXN], bln[MXN];
  vector<int> E[MXN], rE[MXN], vec;
  void init(int _n){
    n = _n;
    for (int i=0; i<n; i++){
      E[i].clear();
      rE[i].clear();
    }
  }
  void add_edge(int u, int v){
    E[u].PB(v);
    rE[v].PB(u);
  }
  void DFS(int u){
    vst[u]=1;
    for (auto v : E[u])
      if (!vst[v]) DFS(v);
    vec.PB(u);
  }
  void rDFS(int u){
    vst[u] = 1;
    bln[u] = nScc;
    for (auto v : rE[u])
      if (!vst[v]) rDFS(v);
  }
  void solve(){
    nScc = 0;
    vec.clear();
    for (int i=0; i<n; i++) vst[i] = 0;
    for (int i=0; i<n; i++)
      if (!vst[i]) DFS(i);
    reverse(vec.begin(),vec.end());
    for (int i=0; i<n; i++) vst[i] = 0;
    for (auto v : vec){
      if (!vst[v]){
        rDFS(v);
        nScc++;
      }
    }
  }
};
```

3.4 DMST_with_sol

```cpp
const int INF = 1029384756;

struct edge_t{
  int u,v,w;
  set< pair<int,int> > add, sub;
  edge_t() : u(-1), v(-1), w(0) {}
  edge_t(int _u, int _v, int _w) {
    u = _u; v = _v; w = _w;
    add.insert({u, v});
  }
  edge_t& operator += (const edge_t& obj) {
    w += obj.w;
    FOR (it, obj.add) {
      if (!sub.count(*it)) add.insert(*it);
      else sub.erase(*it);
    }
    FOR (it, obj.sub) {
      if (!add.count(*it)) sub.insert(*it);
      else add.erase(*it);
    }
    return *this;
  }
  edge_t& operator -= (const edge_t& obj) {
    w -= obj.w;
    FOR (it, obj.sub) {
      if (!sub.count(*it)) add.insert(*it);
      else sub.erase(*it);
    }
    for (auto it : obj.add) {
      if (!add.count(it)) sub.insert(it);
      else add.erase(it);
    }
    return *this;
  }
}eg[MXN*MXN],prv[MXN],EDGE_INF(-1,-1,INF);
int N,M;
int cid,incyc[MXN],contracted[MXN];
vector<int> E[MXN];

edge_t dmst(int rt){
  edge_t cost;
  for (int i=0; i<N; i++){
    contracted[i] = incyc[i] = 0;
    prv[i] = EDGE_INF;
  }
  cid = 0;
  int u,v;
  while (true){
    for (v=0; v<N; v++){
      if (v != rt && !contracted[v] && prv[v].w ==
          INF) break;
    }
    if (v >= N) break; // end
    for (int i=0; i<M; i++){
      if (eg[i].v == v && eg[i].w < prv[v].w)
        prv[v] = eg[i];
    }
    if (prv[v].w == INF) // not connected
      return EDGE_INF;
    cost += prv[v];
    for (u=prv[v].u; u!=v && u!=-1; u=prv[u].u);
    if (u == -1) continue;
    incyc[v] = ++cid;
    for (u=prv[v].u; u!=v; u=prv[u].u){
      contracted[u] = 1;
      incyc[u] = cid;
    }
    for (int i=0; i<M; i++){
      if (incyc[eg[i].u] != cid && incyc[eg[i].v] ==
          cid){
        eg[i] -= prv[eg[i].v];
      }
    }
    for (int i=0; i<M; i++){
      if (incyc[eg[i].u] == cid) eg[i].u = v;
      if (incyc[eg[i].v] == cid) eg[i].v = v;
      if (eg[i].u == eg[i].v) eg[i--] = eg[--M];
    }
    for (int i=0; i<N; i++){
      if (contracted[i]) continue;
      if (prv[i].u>=0 && incyc[prv[i].u] == cid)
        prv[i].u = v;
    }
    prv[v] = EDGE_INF;
  }
  return cost;
}

void solve(){
  edge_t cost = dmst(0);
  for (auto it : cost.add){ // find a solution
    E[it.F].PB(it.S);
    prv[it.S] = edge_t(it.F,it.S,0);
  }
}
```

3.5 Dominator Tree

```cpp
// idom[n] is the unique node that strictly
    dominates n but does
// not strictly dominate any other node that
    strictly dominates n.
// idom[n] = 0 if n is entry or the entry cannot
    reach n.
struct DominatorTree{
  static const int MAXN = 200010;
```

```cpp
int n,s;
vector<int> g[MAXN],pred[MAXN];
vector<int> cov[MAXN];
int dfn[MAXN],nfd[MAXN],ts;
int par[MAXN];
int sdom[MAXN],idom[MAXN];
int mom[MAXN],mn[MAXN];

inline bool cmp(int u,int v) { return dfn[u] <
    dfn[v]; }

int eval(int u) {
  if(mom[u] == u) return u;
  int res = eval(mom[u]);
  if(cmp(sdom[mn[mom[u]]],sdom[mn[u]]))
    mn[u] = mn[mom[u]];
  return mom[u] = res;
}

void init(int _n, int _s) {
  n = _n;
  s = _s;
  REP1(i,1,n) {
    g[i].clear();
    pred[i].clear();
    idom[i] = 0;
  }
}
void add_edge(int u, int v) {
  g[u].push_back(v);
  pred[v].push_back(u);
}
void DFS(int u) {
  ts++;
  dfn[u] = ts;
  nfd[ts] = u;
  for(int v:g[u]) if(dfn[v] == 0) {
    par[v] = u;
    DFS(v);
  }
}
void build() {
  ts = 0;
  REP1(i,1,n) {
    dfn[i] = nfd[i] = 0;
    cov[i].clear();
    mom[i] = mn[i] = sdom[i] = i;
  }
  DFS(s);
  for (int i=ts; i>=2; i--) {
    int u = nfd[i];
    if(u == 0) continue ;
    for(int v:pred[u]) if(dfn[v]) {
      eval(v);
      if(cmp(sdom[mn[v]],sdom[u])) sdom[u] =
          sdom[mn[v]];
    }
    cov[sdom[u]].push_back(u);
    mom[u] = par[u];
    for(int w:cov[par[u]]) {
      eval(w);
      if(cmp(sdom[mn[w]],par[u])) idom[w] = mn[w];
      else idom[w] = par[u];
    }
    cov[par[u]].clear();
  }
  REP1(i,2,ts) {
    int u = nfd[i];
    if(u == 0) continue ;
    if(idom[u] != sdom[u]) idom[u] = idom[idom[u]];
  }
}
}dom;
```

3.6 Maximum Clique

```cpp
class MaxClique {
```

```cpp
public:
    static const int MV = 210;

    int V;
    int el[MV][MV/30+1];
    int dp[MV];
    int ans;
    int s[MV][MV/30+1];
    vector<int> sol;

    void init(int v) {
        V = v; ans = 0;
        FZ(el); FZ(dp);
    }

    /* Zero Base */
    void addEdge(int u, int v) {
        if(u > v) swap(u, v);
        if(u == v) return;
        el[u][v/32] |= (1<<(v%32));
    }

    bool dfs(int v, int k) {
        int c = 0, d = 0;
        for(int i=0; i<(V+31)/32; i++) {
            s[k][i] = el[v][i];
            if(k != 1) s[k][i] &= s[k-1][i];
            c += __builtin_popcount(s[k][i]);
        }
        if(c == 0) {
            if(k > ans) {
                ans = k;
                sol.clear();
                sol.push_back(v);
                return 1;
            }
            return 0;
        }
        for(int i=0; i<(V+31)/32; i++) {
            for(int a = s[k][i]; a ; d++) {
                if(k + (c-d) <= ans) return 0;
                int lb = a&(-a), lg = 0;
                a ^= lb;
                while(lb!=1) {
                    lb = (unsigned int)(lb) >> 1;
                    lg ++;
                }
                int u = i*32 + lg;
                if(k + dp[u] <= ans) return 0;
                if(dfs(u, k+1)) {
                    sol.push_back(v);
                    return 1;
                }
            }
        }
        return 0;
    }

    int solve() {
        for(int i=V-1; i>=0; i--) {
            dfs(i, 1);
            dp[i] = ans;
        }
        return ans;
    }
};
```

3.7 MinimumMeanCycle

```cpp
/* minimum mean cycle */
const int MAXE = 1805;
const int MAXN = 35;
const double inf = 1029384756;
const double eps = 1e-6;
struct Edge {
  int v,u;
  double c;
```

```
};
int n,m,prv[MAXN][MAXN], prve[MAXN][MAXN], vst[MAXN];
Edge e[MAXE];
vector<int> edgeID, cycle, rho;
double d[MAXN][MAXN];
inline void bellman_ford() {
  for(int i=0; i<n; i++) d[0][i]=0;
  for(int i=0; i<n; i++) {
    fill(d[i+1], d[i+1]+n, inf);
    for(int j=0; j<m; j++) {
      int v = e[j].v, u = e[j].u;
      if(d[i][v]<inf && d[i+1][u]>d[i][v]+e[j].c) {
        d[i+1][u] = d[i][v]+e[j].c;
        prv[i+1][u] = v;
        prve[i+1][u] = j;
      }
    }
  }
}
double karp_mmc() {
  // returns inf if no cycle, mmc otherwise
  double mmc=inf;
  int st = -1;
  bellman_ford();
  for(int i=0; i<n; i++) {
    double avg=-inf;
    for(int k=0; k<n; k++) {
      if(d[n][i]<inf-eps)
          avg=max(avg,(d[n][i]-d[k][i])/(n-k));
      else avg=max(avg,inf);
    }
    if (avg < mmc) tie(mmc, st) = tie(avg, i);
  }
  for(int i=0; i<n; i++) vst[i] = 0;
  edgeID.clear(); cycle.clear(); rho.clear();
  for (int i=n; !vst[st]; st=prv[i--][st]) {
    vst[st]++;
    edgeID.PB(prve[i][st]);
    rho.PB(st);
  }
  while (vst[st] != 2) {
    int v = rho.back(); rho.pop_back();
    cycle.PB(v);
    vst[v]++;
  }
  reverse(ALL(edgeID));
  edgeID.resize(SZ(cycle));
  return mmc;
}
```

4 Flow

4.1 Push-relabel

```
#include <algorithm>
#include <list>
constexpr int maxn = 604;
int c[maxn][maxn], f[maxn][maxn], h[maxn], e[maxn],
    g[2 * maxn + 1];
int max_flow(int s, int t, int n) {
  for (int i = 0; i < n; i++)
    fill_n(f[i], n, 0);
  fill_n(h, n, 0);
  fill_n(e, n, 0);
  fill_n(g, 2 * n + 1, 0);

  for (int i = 0; i < n; i++) {
    f[s][i] = e[i] = c[s][i];
    f[i][s] = -c[s][i];
  }
  h[s] = n;
  e[s]++, e[t]++;
  g[0] = n - 1;
  g[n] = 1;

  list<int> fifo;
```

```
  for (int i = 0; i < n; i++)
    if (i != s && i != t && e[i])
      fifo.push_back(i);
  while (!fifo.empty()) {
    int u = fifo.front();
    fifo.pop_front();
    while (e[u]) {
      for (int v = 0; e[u] && v < n; v++) {
        if (h[u] == h[v] + 1 && f[u][v] < c[u][v]) {
          if (e[v] == 0)
            fifo.push_back(v);
          int x = min(e[u], c[u][v] - f[u][v]);
          e[u] -= x;
          e[v] += x;
          f[u][v] += x;
          f[v][u] -= x;
        }
      }
      if (e[u]) {
        if (--g[h[u]] == 0 && h[u] < n)
          for (int i = 0; i < n; i++)
            if (h[i] > h[u] && h[i] < n)
              h[i] = n + 1;
        h[u] = 2 * n;
        for (int v = 0; v < n; v++)
          if (f[u][v] < c[u][v])
            h[u] = min(h[u], h[v] + 1);
        g[h[u]]++;
      }
    }
  }
  return e[t] - 1;
}
```

4.2 Dinic

```
struct Dinic{
  static const int MXN = 10000;
  struct Edge{ int v,f,re; };
  int n,s,t,level[MXN];
  vector<Edge> E[MXN];
  void init(int _n, int _s, int _t){
    n = _n; s = _s; t = _t;
    for (int i=0; i<n; i++) E[i].clear();
  }
  void add_edge(int u, int v, int f){
    E[u].PB({v,f,SZ(E[v])});
    E[v].PB({u,0,SZ(E[u])-1});
  }
  bool BFS(){
    for (int i=0; i<n; i++) level[i] = -1;
    queue<int> que;
    que.push(s);
    level[s] = 0;
    while (!que.empty()){
      int u = que.front(); que.pop();
      for (auto it : E[u]){
        if (it.f > 0 && level[it.v] == -1){
          level[it.v] = level[u]+1;
          que.push(it.v);
        }
      }
    }
    return level[t] != -1;
  }
  int DFS(int u, int nf){
    if (u == t) return nf;
    int res = 0;
    for (auto &it : E[u]){
      if (it.f > 0 && level[it.v] == level[u]+1){
        int tf = DFS(it.v, min(nf,it.f));
        res += tf; nf -= tf; it.f -= tf;
        E[it.v][it.re].f += tf;
        if (nf == 0) return res;
      }
    }
    if (!res) level[u] = -1;
```

```
      return res;
    }
    int flow(int res=0){
      while ( BFS() )
        res += DFS(s,2147483647);
      return res;
    }
}flow;
```

4.3 Cost Flow

```
typedef pair<long long, long long> pll;
struct CostFlow {
  static const int MXN = 205;
  static const long long INF = 102938475610293847LL;
  struct Edge {
    int v, r;
    long long f, c;
  };
  int n, s, t, prv[MXN], prvL[MXN], inq[MXN];
  long long dis[MXN], fl, cost;
  vector<Edge> E[MXN];
  void init(int _n, int _s, int _t) {
    n = _n; s = _s; t = _t;
    for (int i=0; i<n; i++) E[i].clear();
    fl = cost = 0;
  }
  void add_edge(int u, int v, long long f, long long
      c) {
    E[u].PB({v, SZ(E[v])  , f,  c});
    E[v].PB({u, SZ(E[u])-1, 0, -c});
  }
  pll flow() {
    while (true) {
      for (int i=0; i<n; i++) {
        dis[i] = INF;
        inq[i] = 0;
      }
      dis[s] = 0;
      queue<int> que;
      que.push(s);
      while (!que.empty()) {
        int u = que.front(); que.pop();
        inq[u] = 0;
        for (int i=0; i<SZ(E[u]); i++) {
          int v = E[u][i].v;
          long long w = E[u][i].c;
          if (E[u][i].f > 0 && dis[v] > dis[u] + w) {
            prv[v] = u; prvL[v] = i;
            dis[v] = dis[u] + w;
            if (!inq[v]) {
              inq[v] = 1;
              que.push(v);
            }
          }
        }
      }
      if (dis[t] == INF) break;
      long long tf = INF;
      for (int v=t, u, l; v!=s; v=u) {
        u=prv[v]; l=prvL[v];
        tf = min(tf, E[u][l].f);
      }
      for (int v=t, u, l; v!=s; v=u) {
        u=prv[v]; l=prvL[v];
        E[u][l].f -= tf;
        E[v][E[u][l].r].f += tf;
      }
      cost += tf * dis[t];
      fl += tf;
    }
    return {fl, cost};
  }
}flow;
```

4.4 Kuhn Munkres

```
struct KM{
// Maximum Bipartite Weighted Matching (Perfect
    Match)
  static const int MXN = 650;
  static const int INF = 2147483647; // long long
  int n,match[MXN],vx[MXN],vy[MXN];
  int edge[MXN][MXN],lx[MXN],ly[MXN],slack[MXN];
  // ^^^^ long long
  void init(int _n){
    n = _n;
    for (int i=0; i<n; i++)
      for (int j=0; j<n; j++)
        edge[i][j] = 0;
  }
  void add_edge(int x, int y, int w){ // long long
    edge[x][y] = w;
  }
  bool DFS(int x){
    vx[x] = 1;
    for (int y=0; y<n; y++){
      if (vy[y]) continue;
      if (lx[x]+ly[y] > edge[x][y]){
        slack[y] = min(slack[y],
            lx[x]+ly[y]-edge[x][y]);
      } else {
        vy[y] = 1;
        if (match[y] == -1 || DFS(match[y])){
          match[y] = x;
          return true;
        }
      }
    }
    return false;
  }
  int solve(){
    fill(match,match+n,-1);
    fill(lx,lx+n,-INF);
    fill(ly,ly+n,0);
    for (int i=0; i<n; i++)
      for (int j=0; j<n; j++)
        lx[i] = max(lx[i], edge[i][j]);
    for (int i=0; i<n; i++){
      fill(slack,slack+n,INF);
      while (true){
        fill(vx,vx+n,0);
        fill(vy,vy+n,0);
        if ( DFS(i) ) break;
        int d = INF; // long long
        for (int j=0; j<n; j++)
          if (!vy[j]) d = min(d, slack[j]);
        for (int j=0; j<n; j++){
          if (vx[j]) lx[j] -= d;
          if (vy[j]) ly[j] += d;
          else slack[j] -= d;
        }
      }
    }
    int res=0;
    for (int i=0; i<n; i++)
      res += edge[match[i]][i];
    return res;
  }
}graph;
```

4.5 SW-Mincut

```
struct SW{ // O(V^3) 0-base
  static const int MXN = 514;
  int n,vst[MXN],del[MXN];
  int edge[MXN][MXN],wei[MXN];
  void init(int _n){
    n = _n;
    for (int i=0; i<n; i++) {
      for (int j=0; j<n; j++)
        edge[i][j] = 0;
      del[i] = 0;
    }
```

```cpp
  }
  void add_edge(int u, int v, int w){
    edge[u][v] += w;
    edge[v][u] += w;
  }
  void search(int &s, int &t){
    for (int i=0; i<n; i++)
      vst[i] = wei[i] = 0;
    s = t = -1;
    while (true){
      int mx=-1, cur=0;
      for (int i=0; i<n; i++)
        if (!del[i] && !vst[i] && mx<wei[i])
          cur = i, mx = wei[i];
      if (mx == -1) break;
      vst[cur] = 1;
      s = t;
      t = cur;
      for (int i=0; i<n; i++)
        if (!vst[i] && !del[i]) wei[i] +=
            edge[cur][i];
    }
  }
  int solve(){
    int res = 2147483647;
    for (int i=0,x,y; i<n-1; i++){
      search(x,y);
      res = min(res,wei[y]);
      del[y] = 1;
      for (int j=0; j<n; j++)
        edge[x][j] = (edge[j][x] += edge[y][j]);
    }
    return res;
  }
}graph;
```

4.6 Maximum Simple Graph Matching

```cpp
struct GenMatch { // 1-base
  static const int MAXN = 514;
  int V;
  bool el[MAXN][MAXN];
  int pr[MAXN];
  bool inq[MAXN],inp[MAXN],inb[MAXN];
  queue<int> qe;
  int st,ed;
  int nb;
  int bk[MAXN],djs[MAXN];
  int ans;
  void init(int _V) {
    V = _V;
    for(int i = 0; i <= V; i++) {
      for(int j = 0; j <= V; j++) el[i][j] = 0;
      pr[i] = bk[i] = djs[i] = 0;
      inq[i] = inp[i] = inb[i] = 0;
    }
    ans = 0;
  }
  void add_edge(int u, int v) {
    el[u][v] = el[v][u] = 1;
  }
  int lca(int u,int v) {
    for(int i = 0; i <= V; i++) inp[i] = 0;
    while(1) {
      u = djs[u];
      inp[u] = true;
      if(u == st) break;
      u = bk[pr[u]];
    }
    while(1) {
      v = djs[v];
      if(inp[v]) return v;
      v = bk[pr[v]];
    }
    return v;
```

```cpp
  }
  void upd(int u) {
    int v;
    while(djs[u] != nb) {
      v = pr[u];
      inb[djs[u]] = inb[djs[v]] = true;
      u = bk[v];
      if(djs[u] != nb) bk[u] = v;
    }
  }
  void blo(int u,int v) {
    nb = lca(u,v);
    for (int i=0; i<=V; i++) inb[i] = 0;
    upd(u); upd(v);
    if(djs[u] != nb) bk[u] = v;
    if(djs[v] != nb) bk[v] = u;
    for(int tu = 1; tu <= V; tu++)
      if(inb[djs[tu]]) {
        djs[tu] = nb;
        if(!inq[tu]){
          qe.push(tu);
          inq[tu] = 1;
        }
      }
  }
  void flow() {
    for(int i = 1; i <= V; i++) {
      inq[i] = 0;
      bk[i] = 0;
      djs[i] = i;
    }

    while(qe.size()) qe.pop();
    qe.push(st);
    inq[st] = 1;
    ed = 0;
    while(qe.size()) {
      int u = qe.front(); qe.pop();
      for(int v = 1; v <= V; v++)
        if(el[u][v] && (djs[u] != djs[v]) && (pr[u]
            != v)) {
          if((v == st) || ((pr[v] > 0) && bk[pr[v]]
              > 0))
            blo(u,v);
          else if(bk[v] == 0) {
            bk[v] = u;
            if(pr[v] > 0) {
              if(!inq[pr[v]]) qe.push(pr[v]);
            } else {
              ed = v;
              return;
            }
          }
        }
    }
  }
  void aug() {
    int u,v,w;
    u = ed;
    while(u > 0) {
      v = bk[u];
      w = pr[v];
      pr[v] = u;
      pr[u] = v;
      u = w;
    }
  }
  int solve() {
    for(int i = 0; i <= V; i++) pr[i] = 0;
    for(int u = 1; u <= V; u++)
      if(pr[u] == 0) {
        st = u;
        flow();
        if(ed > 0) {
          aug();
          ans ++;
        }
```

```
      }
      return ans;
    }
}G;

int main() {
  G.init(V);
  for(int i=0; i<E; i++) {
    int u, v;
    cin >> u >> v;
    G.add_edge(u, v);
  }
  cout << G.solve() << endl;
}
```

4.7 Minimum Weight Matching (Clique version)

```
struct Graph {
  // Minimum General Weighted Matching (Perfect
      Match) 0-base
  static const int MXN = 105;

  int n, edge[MXN][MXN];
  int match[MXN],dis[MXN],onstk[MXN];
  vector<int> stk;

  void init(int _n) {
    n = _n;
    for (int i=0; i<n; i++)
      for (int j=0; j<n; j++)
        edge[i][j] = 0;
  }
  void add_edge(int u, int v, int w) {
    edge[u][v] = edge[v][u] = w;
  }
  bool SPFA(int u){
    if (onstk[u]) return true;
    stk.PB(u);
    onstk[u] = 1;
    for (int v=0; v<n; v++){
      if (u != v && match[u] != v && !onstk[v]){
        int m = match[v];
        if (dis[m] > dis[u] - edge[v][m] +
            edge[u][v]){
          dis[m] = dis[u] - edge[v][m] + edge[u][v];
          onstk[v] = 1;
          stk.PB(v);
          if (SPFA(m)) return true;
          stk.pop_back();
          onstk[v] = 0;
        }
      }
    }
    onstk[u] = 0;
    stk.pop_back();
    return false;
  }

  int solve() {
    // find a match
    for (int i=0; i<n; i+=2){
      match[i] = i+1;
      match[i+1] = i;
    }
    while (true){
      int found = 0;
      for (int i=0; i<n; i++)
        dis[i] = onstk[i] = 0;
      for (int i=0; i<n; i++){
        stk.clear();
        if (!onstk[i] && SPFA(i)){
          found = 1;
          while (SZ(stk)>=2){
            int u = stk.back(); stk.pop_back();
            int v = stk.back(); stk.pop_back();
```

```
            match[u] = v;
            match[v] = u;
          }
        }
      }
      if (!found) break;
    }
    int ret = 0;
    for (int i=0; i<n; i++)
      ret += edge[i][match[i]];
    ret /= 2;
    return ret;
  }
}graph;
```

4.8 (+1) SW-mincut $O(NM)$

```
// {{{ StoerWagner
const int inf=1000000000;
// should be larger than max.possible mincut
class StoerWagner {
  public:
    int n,mc; // node id in [0,n-1]
    vector<int> adj[MAXN];
    int cost[MAXN][MAXN];
    int cs[MAXN];
    bool merged[MAXN],sel[MAXN];
    // --8<-- include only if cut is explicitly
        needed
    DisjointSet djs;
    vector<int> cut;
    //--8<----------
    StoerWagner(int _n):n(_n),mc(inf),djs(_n) {
      for(int i=0;i<n;i++)
        merged[i]=0;
      for(int i=0;i<n;i++)
        for(int j=0;j<n;j++)
          cost[i][j]=cost[j][i]=0;
    }
    void append(int v,int u,int c) {
      if(v==u) return;
      if(!cost[v][u]&&c) {
        adj[v].PB(u);
        adj[u].PB(v);
      }
      cost[v][u]+=c;
      cost[u][v]+=c;
    }
    void merge(int v,int u) {
      merged[u]=1;
      for(int i=0;i<n;i++)
        append(v,i,cost[u][i]);
      // --8<-- include only if cut is explicitly
          needed
      djs.merge(v,u);
      //--8<----------
    }
    void phase() {
      priority_queue<pii> pq;
      for(int v=0;v<n;v++) {
        if(merged[v]) continue;
        cs[v]=0;
        sel[v]=0;
        pq.push({0,v});
      }
      int v,s,pv;
      while(pq.size()) {
        if(cs[pq.top().S]>pq.top().F) {
          pq.pop();
          continue;
        }
        pv=v;
        v=pq.top().S;
        s=pq.top().F;
        pq.pop();
        sel[v]=1;
        for(int i=0;i<adj[v].size();i++) {
```

```
                int u=adj[v][i];
                if(merged[u]||sel[u]) continue;
                cs[u]+=cost[v][u];
                pq.push({cs[u],u});
            }
        }
        if(s<mc) {
            mc=s;
            // --8<-- include only if cut is explicitly
                needed ------
            cut.clear();
            for(int i=0;i<n;i++)
                if(djs.getrep(i)==djs.getrep(v)) cut.PB(i);
            //--8<----------
        }
        merge(v,pv);
    }
    int mincut() {
        if(mc==inf) {
            for(int t=0;t<n-1;t++)
                phase();

            return mc;
        }
    }
    // --8<-- include only if cut is explicitly
        needed
    vector<int> getcut() { // return one side of
        the cut
        mincut();
        return cut;
    }
    //--8<----------
};
// }}}
```

5   Math

5.1   ax+by=gcd

```
typedef pair<int, int> pii;

pii gcd(int a, int b){
    if(b == 0) return make_pair(1, 0);
    else{
        int p = a / b;
        pii q = gcd(b, a % b);
        return make_pair(q.second, q.first - q.second *
            p);
    }
}
```

5.2   Fast Fourier Transform

```
// const int MAXN = 262144;
// (must be 2^k)

typedef long double ld;
typedef complex<ld> cplx;
const ld PI = acosl(-1);
const cplx I(0, 1);

cplx omega[MAXN+1];
void pre_fft()
{
    for(int i=0; i<=MAXN; i++)
        omega[i] = exp(i * 2 * PI / MAXN * I);
}
void fft(int n, cplx a[], bool inv=false)
{
    int basic = MAXN / n;
    int theta = basic;
    for (int m = n; m >= 2; m >>= 1) {
        int mh = m >> 1;
        for (int i = 0; i < mh; i++) {
```

```
            cplx w = omega[inv ? MAXN-(i*theta%MAXN) :
                i*theta%MAXN];
            for (int j = i; j < n; j += m) {
                int k = j + mh;
                cplx x = a[j] - a[k];
                a[j] += a[k];
                a[k] = w * x;
            }
        }
        theta = (theta * 2) % MAXN;
    }
    int i = 0;
    for (int j = 1; j < n - 1; j++) {
        for (int k = n >> 1; k > (i ^= k); k >>= 1);
        if (j < i) swap(a[i], a[j]);
    }
    if (inv)
        for (i = 0; i < n; i++)
            a[i] /= n;
}
```

5.3   Fast Linear Recurrence

```
ll n,m,dp[N+N];
void pre_dp(){
    dp[0]= 1;
    ll bdr = min(m+m,n);
    for(ll i=1; i<=bdr; i++)
        for(ll j=i-1; j>=max(0ll,i-m); j--)
            dp[i]= add(dp[i],dp[j]);
}
vector<ll> Mul(const vector<ll>& v1,const
    vector<ll>& v2){
    int sz1 = (int)v1.size();
    int sz2 = (int)v2.size();
    assert(sz1 == m and sz2 == m);
    vector<ll> _v(m+m);
    for(int i=0; i<m+m; i++) _v[i]= 0;
    // expand
    for(int i=0; i<sz1; i++)
        for(int j=0; j<sz2; j++)
            _v[i+j+1]= add(_v[i+j+1],mul(v1[i],v2[j]));
    // shrink
    for(int i=0; i<m; i++)
        for(int j=1; j<=m; j++)
            _v[i + j]= add(_v[i + j],_v[i]);
    for(int i=0; i<m; i++)
        _v[i]= _v[i + m];
    _v.resize(m);
    return _v;
}
vector<ll> I,A;
ll solve(){
    pre_dp();
    if(n <= m+m)return dp[n];
    I.resize(m);
    A.resize(m);
    for(int i=0; i<m; i++) I[i]=A[i]=1;
    // dp[n]= /Sum_{i=0}^{m-1} A_i * dp[n - i - 1]
    ll dlt = (n - m) / m;
    ll rdlt = dlt * m;
    while(dlt){
        if(dlt & 1ll) I = Mul(I,A);
        A = Mul(A,A);
        dlt >>= 1;
    }
    ll ans = 0;
    for(int i=0; i<m; i++)
        ans = add(ans,mul(I[i],dp[n-i-1-rdlt]));
    return ans;
}
```

5.4   (+1) ntt

```
int P=605028353,root=3,MAXNUM=262144;
// Remember coefficient are mod P
```

```
/*
p=a*2^n+1
n   2^n         p           a   root
5   32          97          3   5
6   64          193         3   5
7   128         257         2   3
8   256         257         1   3
9   512         7681        15  17
10  1024        12289       12  11
11  2048        12289       6   11
12  4096        12289       3   11
13  8192        40961       5   3
14  16384       65537       4   3
15  32768       65537       2   3
16  65536       65537       1   3
17  131072      786433      6   10
18  262144      786433      3   10 (605028353,
    2308, 3)
19  524288      5767169     11  3
20  1048576     7340033     7   3
21  2097152     23068673    11  3
22  4194304     104857601   25  3
23  8388608     167772161   20  3
24  16777216    167772161   10  3
25  33554432    167772161   5   3 (1107296257,
    33, 10)
26  67108864    469762049   7   3
27  134217728   2013265921  15  31
*/
int bigmod(long long a,int b){
  if(b==0)return 1;
  return (bigmod((a*a)%P,b/2)*(b%2?a:1ll))%P;
}
int inv(int a,int b){
  if(a==1)return 1;
  return (((long long)(a-inv(b%a,a))*b+1)/a)%b;
}
std::vector<long long> ps(MAXNUM);
std::vector<int> rev(MAXNUM);
struct poly{
  std::vector<unsigned int> co;
  int n;//polynomial degree = n
  poly(int d){n=d;co.resize(n+1,0);}
  void trans2(int NN){
    int r=0,st,N;
    unsigned int a,b;
    while((1<<r)<(NN>>1))++r;
    for(N=2;N<=NN;N<<=1,--r){
      for(st=0;st<NN;st+=N){
        int i,ss=st+(N>>1);
        for(i=(N>>1)-1;i>=0;--i){
          a=co[st+i]; b=(ps[i<<r]*co[ss+i])%P;
          co[st+i]=a+b; if(co[st+i]>=P)co[st+i]-=P;
          co[ss+i]=a+P-b; if(co[ss+i]>=P)co[ss+i]-=P;
        }
      }
    }
  }
  void trans1(int NN){
    int r=0,st,N;
    unsigned int a,b;
    for(N=NN;N>1;N>>=1,++r){
      for(st=0;st<NN;st+=N){
        int i,ss=st+(N>>1);
        for(i=(N>>1)-1;i>=0;--i){
          a=co[st+i]; b=co[ss+i];
          co[st+i]=a+b; if(co[st+i]>=P)co[st+i]-=P;
          co[ss+i]=((a+P-b)*ps[i<<r])%P;
        }
      }
    }
  }
  poly operator*(const poly& _b)const{
    poly a=*this,b=_b;
    int k=n+b.n,i,N=1;
    while(N<=k)N*=2;
    a.co.resize(N,0); b.co.resize(N,0);
```

```
    int r=bigmod(root,(P-1)/N),Ni=inv(N,P);
    ps[0]=1;
    for(i=1;i<N;++i)ps[i]=(ps[i-1]*r)%P;
    a.trans1(N);b.trans1(N);
    for(i=0;i<N;++i)a.co[i]=((long
      long)a.co[i]*b.co[i])%P
      ;
    r=inv(r,P);
    for(i=1;i<N/2;++i)std::swap(ps[i],ps[N-i]);
    a.trans2(N);
    for(i=0;i<N;++i)a.co[i]=((long
      long)a.co[i]*Ni)%P;
    a.n=n+_b.n; return a;
  }
};
```

5.5 Mod

```
/// _fd(a,b)  floor(a/b).
/// _rd(a,m)  a-floor(a/m)*m.
/// _pv(a,m,r) largest x s.t x<=a && x%m == r.
/// _nx(a,m,r) smallest x s.t x>=a && x%m == r.
/// _ct(a,b,m,r) |A| , A = { x : a<=x<=b && x%m == r
    }.

int _fd(int a,int b){ return a<0?(-~a/b-1):a/b; }
int _rd(int a,int m){ return a-_fd(a,m)*m; }
int _pv(int a,int m,int r)
{
    r=(r%m+m)%m;
    return _fd(a-r,m)*m+r;
}
int _nt(int a,int m,int r)
{
    m=abs(m);
    r=(r%m+m)%m;
    return _fd(a-r-1,m)*m+r+m;
}
int _ct(int a,int b,int m,int r)
{
    m=abs(m);
    a=_nt(a,m,r);
    b=_pv(b,m,r);
    return (a>b)?0:((b-a+m)/m);
}
```

5.6 (+1) Miller Rabin

```
// n < 4,759,123,141       3 :  2, 7, 61
// n < 1,122,004,669,633   4 :  2, 13, 23, 1662803
// n < 3,474,749,660,383     6 :  pirmes <= 13
// n < 2^64                  7 :
// 2, 325, 9375, 28178, 450775, 9780504, 1795265022
// Make sure testing integer is in range [2, n  2]
    if
// you want to use magic.
long long power(long long x,long long p,long long
    mod){
  long long s=1,m=x;
  while(p) {
    if(p&1) s=mult(s,m,mod);
    p>>=1;
    m=mult(m,m,mod);
  }
  return s;
}
bool witness(long long a,long long n,long long u,int
    t){
  long long x=power(a,u,n);
  for(int i=0;i<t;i++) {
    long long nx=mult(x,x,n);
    if(nx==1&&x!=1&&x!=n-1) return 1;
    x=nx;
  }
  return x!=1;
}
```

```cpp
bool miller_rabin(long long n,int s=100) {
  // iterate s times of witness on n
  // return 1 if prime, 0 otherwise
  if(n<2) return 0;
  if(!(n&1)) return n==2;
  long long u=n-1;
  int t=0;
  // n-1 = u*2^t
  while(!(u&1)) {
    u>>=1;
    t++;
  }
  while(s--) {
    long long a=randll()%(n-1)+1;
    if(witness(a,n,u,t)) return 0;
  }
  return 1;
}
```

5.7 Pollard Rho

```cpp
// does not work when n is prime
long long modit(long long x,long long mod) {
  if(x>=mod) x-=mod;
  //if(x<0) x+=mod;
  return x;
}
long long mult(long long x,long long y,long long
    mod) {
  long long s=0,m=x%mod;
  while(y) {
    if(y&1) s=modit(s+m,mod);
    y>>=1;
    m=modit(m+m,mod);
  }
  return s;
}
long long f(long long x,long long mod) {
  return modit(mult(x,x,mod)+1,mod);
}
long long pollard_rho(long long n) {
  if(!(n&1)) return 2;
  while (true) {
    long long y=2, x=rand()%(n-1)+1, res=1;
    for (int sz=2; res==1; sz*=2) {
      for (int i=0; i<sz && res<=1; i++) {
        x = f(x, n);
        res = __gcd(abs(x-y), n);
      }
      y = x;
    }
    if (res!=0 && res!=n) return res;
  }
}
```

5.8 Algorithms about Primes

```
/*
 * 12721
 * 13331
 * 14341
 * 75577
 * 123457
 * 222557
 * 556679
 * 999983
 * 1097774749
 * 1076767633
 * 100102021
 * 999997771
 * 1001010013
 * 1000512343
 * 987654361
 * 999991231
 * 999888733
 * 98789101
```
```
 * 987777733
 * 999991921
 * 1010101333
 * 1010102101
 * 1000000000039
 * 1000000000000037
 * 2305843009213693951
 * 4611686018427387847
 * 9223372036854775783
 * 18446744073709551557
 */
```

```cpp
int mu[MX],p_tbl[MX];
vector<int> primes;
void sieve() {
  mu[1] = p_tbl[1] = 1;
  for (int i=2; i<MX; i++) {
    if (!p_tbl[i]) {
      p_tbl[i] = i;
      primes.PB(i);
      mu[i] = -1;
    }
    for (auto p : primes) {
      int x = i*p;
      if (x >= M) break;
      p_tbl[x] = p;
      mu[x] = -mu[i];
      if (i%p==0) {
        mu[x] = 0;
        break;
      }
    }
  }
}

vector<int> factor(int x) {
  vector<int> fac{1};
  while (x > 1) {
    int fn=SZ(fac), p=p_tbl[x], pos=0;
    while (x%p == 0) {
      x /= p;
      for (int i=0; i<fn; i++)
        fac.PB(fac[pos++]*p);
    }
  }
  return fac;
}
```

5.9 (+1) PolynomialGenerator

```cpp
class PolynomialGenerator {
  /* for a nth-order polynomial f(x), *
   * given f(0), f(1), ..., f(n) *
   * express f(x) as sigma_i{c_i*C(x,i)} */
  public:
    int n;
    vector<long long> coef;
    // initialize and calculate f(x), vector _fx
        should be
    // filled with f(0) to f(n)
    PolynomialGenerator(int _n,vector<long long>
        _fx):n(_n
        ),coef(_fx) {
      for(int i=0;i<n;i++)
        for(int j=n;j>i;j--)
          coef[j]-=coef[j-1];
    }
    // evaluate f(x), runs in O(n)
    long long eval(int x) {
      long long m=1,ret=0;
      for(int i=0;i<=n;i++) {
        ret+=coef[i]*m;
        m=m*(x-i)/(i+1);
      }
      return ret;
    }
};
```

13

5.10   Pseudoinverse of Square matrix

```
Mat pinv(Mat m)
{
  Mat res = I;

  FZ(used);
  for(int i=0; i<W; i++)
  {
    int piv = -1;
    for(int j=0; j<W; j++)
    {
      if(used[j]) continue;
      if(abs(m.v[j][i]) > EPS)
      {
        piv = j;
        break;
      }
    }
    if(piv == -1)
      continue;
    used[i] = true;
    swap(m.v[piv], m.v[i]);
    swap(res.v[piv], res.v[i]);

    ld rat = m.v[i][i];
    for(int j=0; j<W; j++)
    {
      m.v[i][j] /= rat;
      res.v[i][j] /= rat;
    }

    for(int j=0; j<W; j++)
    {
      if(j == i) continue;
      rat = m.v[j][i];
      for(int k=0; k<W; k++)
      {
        m.v[j][k] -= rat * m.v[i][k];
        res.v[j][k] -= rat * res.v[i][k];
      }
    }
  }

  for(int i=0; i<W; i++)
  {
    if(used[i]) continue;
    for(int j=0; j<W; j++)
      res.v[i][j] = 0;
  }

  return res;
}
```

5.11   Simplex

```
const int maxn = 111;
const int maxm = 111;
const double eps = 1E-10;

double a[maxn][maxm], b[maxn], c[maxm],
    d[maxn][maxm];
double x[maxm];
int ix[maxn + maxm]; // !!! array all indexed from 0
// max{cx} subject to {Ax<=b,x>=0}
// n: constraints, m: vars !!!
// x[] is the optimal solution vector
//
// usage :
// value = simplex(a, b, c, N, M);
double simplex(double a[maxn][maxm], double b[maxn],
    double c[maxm], int n, int m) {
    ++m;
    int r = n, s = m - 1;
    memset(d, 0, sizeof(d));
    for (int i = 0; i < n + m; ++i) ix[i] = i;
    for (int i = 0; i < n; ++i) {
        for (int j = 0; j < m - 1; ++j)
            d[i][j] = -a[i][j];
        d[i][m - 1] = 1;
        d[i][m] = b[i];
        if (d[r][m] > d[i][m]) r = i;
    }
    for (int j = 0; j < m - 1; ++j) d[n][j] = c[j];
    d[n + 1][m - 1] = -1;
    for (double dd;; ) {
        if (r < n) {
            int t = ix[s];
            ix[s] = ix[r + m]; ix[r + m] = t;
            d[r][s] = 1.0 / d[r][s];
            for (int j = 0; j <= m; ++j)
                if (j != s) d[r][j] *= -d[r][s];
            for (int i = 0; i <= n + 1; ++i)
                if (i != r) {
                    for (int j = 0; j <= m; ++j)
                        if (j != s)
                            d[i][j] +=
                                d[r][j]*d[i][s];
                    d[i][s] *= d[r][s];
                }
        }
        r = -1; s = -1;
        for (int j = 0; j < m; ++j)
            if (s < 0 || ix[s] > ix[j]) {
                if (d[n + 1][j] > eps || (d[n +
                    1][j] > -eps && d[n][j] > eps))
                    s = j;
            }
        if (s < 0) break;
        for (int i=0; i<n; ++i) if (d[i][s] < -eps) {
            if (r < 0 || (dd = d[r][m] / d[r][s] -
                d[i][m] / d[i][s]) < -eps || (dd <
                eps && ix[r + m] > ix[i + m])) r = i;
        }
        if (r < 0) return -1; // not bounded
    }
    if (d[n + 1][m] < -eps) return -1; // not
        executable
    double ans = 0;
    for(int i=0; i<m; i++) x[i] = 0;
    for (int i = m; i < n + m; ++i) { // the missing
        enumerated x[i] = 0
        if (ix[i] < m - 1)
        {
            ans += d[i - m][m] * c[ix[i]];
            x[ix[i]] = d[i-m][m];
        }
    }
    return ans;
}
```

6   Geometry

6.1   Point operators

```
#define x first
#define y second

#define cpdd const pdd
struct pdd : pair<double, double> {
    using pair<double, double>::pair;

    pdd operator + (cpdd &p) const {
        return {x+p.x, y+p.y};
    }

    pdd operator - () const {
        return {-x, -y};
    }

    pdd operator - (cpdd &p) const {
        return (*this) + (-p);
```

```cpp
    }

    pdd operator * (double f) const {
        return {f*x, f*y};
    }

    double operator * (cpdd &p) const {
        return x*p.x + y*p.y;
    }
};

double abs(cpdd &p) { return hypot(p.x, p.y); }
double arg(cpdd &p) { return atan2(p.y, p.x); }
double cross(cpdd &p, cpdd &q) { return p.x*q.y -
    p.y*q.x; }
double cross(cpdd &p, cpdd &q, cpdd &o) { return
    cross(p-o, q-o); }
pdd operator * (double f, cpdd &p) { return p*f; }
    // !! Not f*p !!
```

6.2   Intersection of two circles

```cpp
using ld = double;
vector<pdd> interCircle(pdd o1, double r1, pdd o2,
    double r2) {
  ld d2 = (o1 - o2) * (o1 - o2);
  ld d = sqrt(d2);
  if (d < abs(r1-r2)) return {};
  if (d > r1+r2) return {};
  pdd u = 0.5*(o1+o2) +
    ((r2*r2-r1*r1)/(2*d2))*(o1-o2);
  double A = sqrt((r1+r2+d) * (r1-r2+d) * (r1+r2-d)
    * (-r1+r2+d));
  pdd v = A / (2*d2) * pdd(o1.S-o2.S, -o1.F+o2.F);
  return {u+v, u-v};
}
```

6.3   Intersection of two lines

```cpp
const double EPS = 1e-9;

pdd interPnt(pdd p1, pdd p2, pdd q1, pdd q2, bool
    &res){
  double f1 = cross(p2, q1, p1);
  double f2 = -cross(p2, q2, p1);
  double f = (f1 + f2);

  if(fabs(f) < EPS) {
    res = false;
    return {};
  }

  res = true;
  return (f2 / f) * q1 + (f1 / f) * q2;
}
```

6.4   Half Plane Intersection

```cpp
const double EPS = 1e-9;

pdd interPnt(Line l1, Line l2, bool &res){
    pdd p1, p2, q1, q2;
    tie(p1, p2) = l1;
    tie(q1, q2) = l2;
  double f1 = cross(p2, q1, p1);
    double f2 = -cross(p2, q2, p1);
  double f = (f1 + f2);

    if(fabs(f) < EPS) {
        res = false;
        return {0, 0};
    }

    res = true;
  return (f2 / f) * q1 + (f1 / f) * q2;
}
```

```cpp
bool isin(Line l0, Line l1, Line l2) {
    // Check inter(l1, l2) in l0
    bool res;
    pdd p = interPnt(l1, l2, res);
    return cross(l0.S, p, l0.F) > EPS;
}

/* If no solution, check: 1. ret.size() < 3
 * Or more precisely, 2. interPnt(ret[0], ret[1])
 * in all the lines. (use (l.S - l.F).cross(p - l.F)
    > 0
 */
vector<Line> halfPlaneInter(vector<Line> lines) {
    int sz = lines.size();
    vector<double> ata(sz), ord(sz);
    for (int i=0; i<sz; i++) {
        ord[i] = i;
        pdd d = lines[i].S - lines[i].F;
        ata[i] = atan2(d.y, d.x);
    }
    sort(ALL(ord), [&](int i, int j) {
        if (abs(ata[i] - ata[j]) < EPS) {
            return cross(lines[i].S, lines[j].S,
                lines[i].F) < 0;
        }
        return ata[i] < ata[j];
    });
    vector<Line> fin;
    for (int i=0; i<sz; i++) {
        if (!i or fabs(ata[ord[i]] - ata[ord[i-1]])
            > EPS) {
            fin.PB(lines[ord[i]]);
        }
    }

    deque<Line> dq;
    for (int i=0; i<SZ(fin); i++) {
        while(SZ(dq) >= 2 and
            not isin(fin[i], dq[SZ(dq)-2],
                dq[SZ(dq)-1])) {
            dq.pop_back();
        }
        while(SZ(dq) >= 2 and
            not isin(fin[i], dq[0], dq[1])) {
            dq.pop_front();
        }
        dq.push_back(fin[i]);
    }

    while (SZ(dq) >= 3 and
        not isin(dq[0], dq[SZ(dq)-2],
            dq[SZ(dq)-1])) {
        dq.pop_back();
    }

    while (SZ(dq) >= 3 and
        not isin(dq[SZ(dq)-1], dq[0], dq[1])) {
        dq.pop_front();
    }
    vector<Line> res(ALL(dq));
    return res;
}
```

6.5   2D Convex Hull

```cpp
vector<pdd> convex_hull(vector<pdd> pt){
  sort(pt.begin(),pt.end());
  int top=0;
  vector<pdd> stk(2*pt.size());
  for (int i=0; i<(int)pt.size(); i++){
    while (top >= 2 &&
        cross(stk[top-1],pt[i],stk[top-2]) <= 0)
      top--;
    stk[top++] = pt[i];
  }
  for (int i=pt.size()-2, t=top+1; i>=0; i--){
```

```
    while (top >= t && cross(stk[top-1],pt[i],
        stk[top-2]) <= 0)
      top--;
    stk[top++] = pt[i];
  }
  stk.resize(top-1);
  return stk;
}
```

6.6 3D Convex Hull

```
// return the faces with pt indexes
int flag[MXN][MXN];
struct Point{
  ld x,y,z;
  Point operator - (const Point &b) const {
    return (Point){x-b.x,y-b.y,z-b.z};
  }
  Point operator * (const ld &b) const {
    return (Point){x*b,y*b,z*b};
  }
  ld len() const { return sqrtl(x*x+y*y+z*z); }
  ld dot(const Point &a) const {
    return x*a.x+y*a.y+z*a.z;
  }
  Point operator * (const Point &b) const {
    return
        (Point){y*b.z-b.y*z,z*b.x-b.z*x,x*b.y-b.x*y};
  }
};
Point ver(Point a, Point b, Point c) {
  return (b - a) * (c - a);
}
vector<Face> convex_hull_3D(const vector<Point> pt) {
  int n = SZ(pt);
  REP(i,n) REP(j,n)
    flag[i][j] = 0;

  vector<Face> now;
  now.push_back((Face){0,1,2});
  now.push_back((Face){2,1,0});
  int ftop = 0;
  for (int i=3; i<n; i++){
    ftop++;
    vector<Face> next;
    REP(j, SZ(now)) {
      Face& f=now[j];
      ld d=(pt[i]-pt[f.a]).dot(ver(pt[f.a], pt[f.b],
          pt[f.c]));
      if (d <= 0) next.push_back(f);
      int ff = 0;
      if (d > 0) ff=ftop;
      else if (d < 0) ff=-ftop;
      flag[f.a][f.b] = flag[f.b][f.c] =
          flag[f.c][f.a] = ff;
    }
    REP(j, SZ(now)) {
      Face& f=now[j];
      if (flag[f.a][f.b] > 0 and flag[f.a][f.b] !=
          flag[f.b][f.a])
        next.push_back((Face){f.a,f.b,i});
      if (flag[f.b][f.c] > 0 and flag[f.b][f.c] !=
          flag[f.c][f.b])
        next.push_back((Face){f.b,f.c,i});
      if (flag[f.c][f.a] > 0 and flag[f.c][f.a] !=
          flag[f.a][f.c])
        next.push_back((Face){f.c,f.a,i});
    }
    now=next;
  }
  return now;
}
```

6.7 Minimum Covering Circle

```
struct Mcc{
```

```
// return pair of center and r^2
static const int MAXN = 1000100;
int n;
pdd p[MAXN],cen;
double r2;

void init(int _n, pdd _p[]){
  n = _n;
  memcpy(p,_p,sizeof(pdd)*n);
}
double sqr(double a){ return a*a; }
double abs2(pdd a){ return a*a; }
pdd center(pdd p0, pdd p1, pdd p2) {
  pdd a = p1-p0;
  pdd b = p2-p0;
  double c1=abs2(a)*0.5;
  double c2=abs2(b)*0.5;
  double d = a % b;
  double x = p0.x + (c1 * b.y - c2 * a.y) / d;
  double y = p0.y + (a.x * c2 - b.x * c1) / d;
  return pdd(x,y);
}

pair<pdd,double> solve(){
  random_shuffle(p,p+n);
  r2=0;
  for (int i=0; i<n; i++){
    if (abs2(cen-p[i]) <= r2) continue;
    cen = p[i];
    r2 = 0;
    for (int j=0; j<i; j++){
      if (abs2(cen-p[j]) <= r2) continue;
      cen = 0.5 * (p[i]+p[j]);
      r2 = abs2(cen-p[j]);
      for (int k=0; k<j; k++){
        if (abs2(cen-p[k]) <= r2) continue;
        cen = center(p[i],p[j],p[k]);
        r2 = abs2(cen-p[k]);
      }
    }
  }
  return {cen,r2};
}
}mcc;
```

6.8 KDTree (Nearest Point)

```
const int MXN = 100005;

struct KDTree {
  struct Node {
    int x,y,x1,y1,x2,y2;
    int id,f;
    Node *L, *R;
  }tree[MXN];
  int n;
  Node *root;

  long long dis2(int x1, int y1, int x2, int y2) {
    long long dx = x1-x2;
    long long dy = y1-y2;
    return dx*dx+dy*dy;
  }
  static bool cmpx(Node& a, Node& b){ return
      a.x<b.x; }
  static bool cmpy(Node& a, Node& b){ return
      a.y<b.y; }
  void init(vector<pair<int,int>> ip) {
    n = ip.size();
    for (int i=0; i<n; i++) {
      tree[i].id = i;
      tree[i].x = ip[i].first;
      tree[i].y = ip[i].second;
    }
    root = build_tree(0, n-1, 0);
  }
  Node* build_tree(int L, int R, int dep) {
```

```
      if (L>R) return nullptr;
      int M = (L+R)/2;
      tree[M].f = dep%2;
      nth_element(tree+L, tree+M, tree+R+1, tree[M].f
          ? cmpy : cmpx);
      tree[M].x1 = tree[M].x2 = tree[M].x;
      tree[M].y1 = tree[M].y2 = tree[M].y;

      tree[M].L = build_tree(L, M-1, dep+1);
      if (tree[M].L) {
        tree[M].x1 = min(tree[M].x1, tree[M].L->x1);
        tree[M].x2 = max(tree[M].x2, tree[M].L->x2);
        tree[M].y1 = min(tree[M].y1, tree[M].L->y1);
        tree[M].y2 = max(tree[M].y2, tree[M].L->y2);
      }

      tree[M].R = build_tree(M+1, R, dep+1);
      if (tree[M].R) {
        tree[M].x1 = min(tree[M].x1, tree[M].R->x1);
        tree[M].x2 = max(tree[M].x2, tree[M].R->x2);
        tree[M].y1 = min(tree[M].y1, tree[M].R->y1);
        tree[M].y2 = max(tree[M].y2, tree[M].R->y2);
      }

      return tree+M;
  }
  int touch(Node* r, int x, int y, long long d2){
    long long dis = sqrt(d2)+1;
    if (x<r->x1-dis || x>r->x2+dis || y<r->y1-dis ||
        y>r->y2+dis)
      return 0;
    return 1;
  }
  void nearest(Node* r, int x, int y, int &mID, long
      long &md2) {
    if (!r || !touch(r, x, y, md2)) return;
    long long d2 = dis2(r->x, r->y, x, y);
    if (d2 < md2 || (d2 == md2 && mID < r->id)) {
      mID = r->id;
      md2 = d2;
    }
    // search order depends on split dim
    if ((r->f == 0 && x < r->x) ||
        (r->f == 1 && y < r->y)) {
      nearest(r->L, x, y, mID, md2);
      nearest(r->R, x, y, mID, md2);
    } else {
      nearest(r->R, x, y, mID, md2);
      nearest(r->L, x, y, mID, md2);
    }
  }
  int query(int x, int y) {
    int id = 1029384756;
    long long d2 = 102938475612345678LL;
    nearest(root, x, y, id, d2);
    return id;
  }
}tree;
```

6.9 Triangulation

```
bool inCircle(pdd a, pdd b, pdd c, pdd d) {
    b = b - a;
    c = c - a;
    d = d - a;
    if (cross(b, c) < 0) swap(b, c);
    double m[3][3] = {
        {b.x, b.y, b*b},
        {c.x, c.y, c*c},
        {d.x, d.y, d*d}
    };

    double det = m[0][0] * (m[1][1]*m[2][2] -
        m[1][2]*m[2][1])
              + m[0][1] * (m[1][2]*m[2][0] -
                  m[1][0]*m[2][2])
```
```
              + m[0][2] * (m[1][0]*m[2][1] -
                  m[1][1]*m[2][0]);
    return det < 0;
}

bool intersect(pdd a, pdd b, pdd c, pdd d) {
    return cross(b, c, a) * cross(b, d, a) < 0 and
        cross(d, a, c) * cross(d, b, c) < 0;
}


const double EPS = 1e-12;
struct Triangulation {
    static const int MXN = 1e5+5;
    int N;
    vector<int> ord;
    vector<pdd> pts;
    set<int> E[MXN];
    vector<vector<int>> solve(vector<pdd> p) {
        N = SZ(p);
        ord.resize(N);
        for (int i=0; i<N; i++) {
            E[i].clear();
            ord[i] = i;
        }
        sort(ALL(ord), [&p](int i, int j) {
            return p[i] < p[j];
        });

        pts.resize(N);
        for (int i=0; i<N; i++) pts[i] = p[ord[i]];

        go(0, N);

        vector<vector<int>> res(N);
        for (int i=0; i<N; i++) {
            int o = ord[i];
            for (auto x: E[i]) {
                res[o].PB(ord[x]);
            }
        }
        return res;
    }

    void add_edge(int u, int v) {
        E[u].insert(v);
        E[v].insert(u);
    }

    void remove_edge(int u, int v) {
        E[u].erase(v);
        E[v].erase(u);
    }

    void go(int l, int r) {
        int n = r - l;

        if (n <= 3) {
            for (int i=l; i<r; i++)
                for (int j=i+1; j<r; j++)
                    add_edge(i, j);
            return;
        }
        int md = (l+r)/2;

        go(l, md);
        go(md, r);

        int il = l, ir = r-1;
        while (1) {
            int nx = -1;
            for (auto i: E[il]) {
                double cs = cross(pts[il], pts[i],
                    pts[ir]);
                if (cs > EPS ||
                    (abs(cs) < EPS and
                        abs(pts[i]-pts[ir]) <
```

```
                    abs(pts[il]-pts[ir]))) {
                nx = i;
                break;
            }
        }
        if (nx != -1) {
            il = nx;
            continue;
        }
        for (auto i: E[ir]) {
            double cs = cross(pts[ir], pts[i],
                pts[il]);
            if (cs < -EPS ||
                (abs(cs) < EPS and
                    abs(pts[i]-pts[il]) <
                    abs(pts[ir]-pts[il]))) {
                nx = i;
                break;
            }
        }

        if (nx != -1) {
            ir = nx;
        } else break;
    }

    add_edge(il, ir);

    while (1) {
        int nx = -1;
        bool is2 = false;

        for (int i: E[il]) {
            if (cross(pts[il], pts[i], pts[ir])
                < -EPS and
                (nx == -1 or inCircle(pts[il],
                    pts[ir], pts[nx], pts[i])))
                    nx = i;
        }

        for (int i: E[ir]) {
            if (cross(pts[ir], pts[i], pts[il])
                > EPS and
                (nx == -1 or inCircle(pts[il],
                    pts[ir], pts[nx], pts[i])))
                    nx = i, is2 = 1;
        }

        if (nx == -1) break;

        int a = il, b = ir;
        if (is2) swap(a, b);

        for (auto i: E[a]) {
            if (intersect(pts[a], pts[i],
                pts[b], pts[nx])) {
                remove_edge(a, i);
            }
        }
        if (is2) {
            add_edge(il, nx);
            ir = nx;
        } else {
            add_edge(ir, nx);
            il = nx;
        }
    }
  }
}
} tri;
```

7 Stringology

7.1 Suffix Array

```
const int MAX = 1020304;
```

```
int ct[MAX], he[MAX], rk[MAX], sa[MAX], tsa[MAX],
    tp[MAX][2];

void suffix_array(char *ip){

  int len = strlen(ip);
  int alp = 256;

  memset(ct, 0, sizeof(ct));
  for(int i=0;i<len;i++) ct[ip[i]+1]++;
  for(int i=1;i<alp;i++) ct[i]+=ct[i-1];
  for(int i=0;i<len;i++) rk[i]=ct[ip[i]];

  for(int i=1;i<len;i*=2){
    for(int j=0;j<len;j++){
      if(j+i>=len) tp[j][1]=0;
      else tp[j][1]=rk[j+i]+1;

      tp[j][0]=rk[j];
    }
    memset(ct, 0, sizeof(ct));
    for(int j=0;j<len;j++) ct[tp[j][1]+1]++;
    for(int j=1;j<len+2;j++) ct[j]+=ct[j-1];
    for(int j=0;j<len;j++) tsa[ct[tp[j][1]]++]=j;

    memset(ct, 0, sizeof(ct));
    for(int j=0;j<len;j++) ct[tp[j][0]+1]++;
    for(int j=1;j<len+1;j++) ct[j]+=ct[j-1];
    for(int j=0;j<len;j++)
        sa[ct[tp[tsa[j]][0]]++]=tsa[j];

    rk[sa[0]]=0;
    for(int j=1;j<len;j++){
      if( tp[sa[j]][0] == tp[sa[j-1]][0] &&
        tp[sa[j]][1] == tp[sa[j-1]][1] )
        rk[sa[j]] = rk[sa[j-1]];
      else
        rk[sa[j]] = j;
    }
  }

  for(int i=0,h=0;i<len;i++){
    if(rk[i]==0) h=0;
    else{
      int j=sa[rk[i]-1];
      h=max(0,h-1);
      for(;ip[i+h]==ip[j+h];h++);
    }
    he[rk[i]]=h;
  }
}
```

7.2 Suffix Array (SAIS TWT514)

```
struct SA{
#define REP(i,n) for ( int i=0; i<int(n); i++ )
#define REP1(i,a,b) for ( int i=(a); i<=int(b); i++ )
static const int MXN = 300010;
bool _t[MXN*2];
int _s[MXN*2], _sa[MXN*2], _c[MXN*2], x[MXN],
    _p[MXN], _q[MXN*2], hei[MXN], r[MXN];
int operator [] (int i){ return _sa[i]; }
void build(int *s, int n, int m){
  memcpy(_s, s, sizeof(int) * n);
  sais(_s, _sa, _p, _q, _t, _c, n, m);
  mkhei(n);
}
void mkhei(int n){
  REP(i,n) r[_sa[i]] = i;
  hei[0] = 0;
  REP(i,n) if(r[i]) {
    int ans = i>0 ? max(hei[r[i-1]] - 1, 0) : 0;
    while(_s[i+ans] == _s[_sa[r[i]-1]+ans]) ans++;
    hei[r[i]] = ans;
  }
}
```

```cpp
void sais(int *s, int *sa, int *p, int *q, bool *t,
    int *c, int n, int z){
  bool uniq = t[n-1] = true, neq;
  int nn = 0, nmxz = -1, *nsa = sa + n, *ns = s + n,
      lst = -1;
#define MS0(x,n) memset((x),0,n*sizeof(*(x)))
#define MAGIC(XD) MS0(sa, n); \
  memcpy(x, c, sizeof(int) * z); \
  XD; \
  memcpy(x + 1, c, sizeof(int) * (z - 1)); \
  REP(i,n) if(sa[i] && !t[sa[i]-1]) \
      sa[x[s[sa[i]-1]]++] = sa[i]-1; \
  memcpy(x, c, sizeof(int) * z); \
  for(int i = n - 1; i >= 0; i--) if(sa[i] && \
      t[sa[i]-1]) sa[--x[s[sa[i]-1]]] = sa[i]-1;
  MS0(c, z);
  REP(i,n) uniq &= ++c[s[i]] < 2;
  REP(i,z-1) c[i+1] += c[i];
  if (uniq) { REP(i,n) sa[--c[s[i]]] = i; return; }
  for(int i = n - 2; i >= 0; i--) t[i] =
      (s[i]==s[i+1] ? t[i+1] : s[i]<s[i+1]);
  MAGIC(REP1(i,1,n-1) if(t[i] && !t[i-1])
      sa[--x[s[i]]]=p[q[i]=nn++]=i);
  REP(i, n) if (sa[i] && t[sa[i]] && !t[sa[i]-1]) {
    neq = lst < 0 || memcmp(s + sa[i], s + lst,
        (p[q[sa[i]] + 1] - sa[i]) * sizeof(int));
    ns[q[lst=sa[i]]]=nmxz+=neq;
  }
  sais(ns, nsa, p + nn, q + n, t + n, c + z, nn,
      nmxz + 1);
  MAGIC(for(int i = nn - 1; i >= 0; i--)
      sa[--x[s[p[nsa[i]]]]] = p[nsa[i]]);
}
}sa;

void suffix_array(int* ip, int len) {
// should padding a zero in the back
// s is int array, n is array length
// s[0..n-1] != 0, and s[n] = 0
// resulting SA will be length n+1
ip[len++] = 0;
sa.build(ip, len, 128);
// original 1-base
for (int i=0; i<l; i++) {
  hei[i] = sa.hei[i + 1];
  sa[i] = sa._sa[i + 1];
}
}
```

7.3 Aho-Corasick Algorithm

```cpp
struct ACautomata{
  struct Node{
    int cnt,dp;
    Node *go[26], *fail;
    Node (){
      cnt = 0;
      dp = -1;
      memset(go,0,sizeof(go));
      fail = 0;
    }
  };

  Node *root, pool[1048576];
  int nMem;

  Node* new_Node(){
    pool[nMem] = Node();
    return &pool[nMem++];
  }
  void init(){
    nMem = 0;
    root = new_Node();
  }
  void add(const string &str){
    insert(root,str,0);
  }
```

```cpp
  void insert(Node *cur, const string &str, int pos){
    if (pos >= (int)str.size()){
      cur->cnt++;
      return;
    }
    int c = str[pos]-'a';
    if (cur->go[c] == 0){
      cur->go[c] = new_Node();
    }
    insert(cur->go[c],str,pos+1);
  }
  void make_fail(){
    queue<Node*> que;
    que.push(root);
    while (!que.empty()){
      Node* fr=que.front();
      que.pop();
      for (int i=0; i<26; i++){
        if (fr->go[i]){
          Node *ptr = fr->fail;
          while (ptr && !ptr->go[i]) ptr = ptr->fail;
          if (!ptr) fr->go[i]->fail = root;
          else fr->go[i]->fail = ptr->go[i];
          que.push(fr->go[i]);
        }
      }
    }
  }
};
```

7.4 KMP

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

void build_fail_function(string B, int *fail) {
    int len = B.length(), pos;
    pos = fail[0] = -1;
    for (int i = 1; i < len; i ++) {
        while (pos != -1 and B[pos + 1] != B[i])
            pos = fail[pos];
        if (B[pos + 1] == B[i]) pos ++;
        fail[i] = pos;
    }
}

void match(string A, string B, int *fail) {
    int lenA = A.length(), lenB = B.length();
    int pos = -1;
    for (int i = 0; i < lenA; i ++) {
        while (pos != -1 and B[pos + 1] != A[i])
            pos = fail[pos];

        if (B[pos + 1] == A[i]) pos ++;

        if (pos == lenB - 1) {
            // Match ! A[i - lenB + 1, i] = B
            pos = fail[pos];
        }
    }
}
```

7.5 Z value

```cpp
void Zval(const char *s, int len, int *z) {
    z[0] = 0;
    for (int b=0, i=1; i<len; i++) {
        z[i] = max(min(z[i-b], z[b] + b - i), 0);
        while (s[i + z[i]] == s[z[i]]) z[i] ++;
        if (i+z[i] > b+z[b]) b=i;
    }
}
```

7.6 Z value (palindrome ver.)

```cpp
void Zpal(const char *s, int len, int *z) {
    // Only odd palindrome len is considered
    // z[i] means that the longest odd palindrom
        centered at
    // i is [i-z[i] .. i+z[i]]
    z[0] = 0;
    for (int b=0, i=1; i<len; i++) {
        if (z[b] + b >= i) z[i] = min(z[2*b-i],
            b+z[b]-i);
        else z[i] = 0;
        while (i+z[i]+1 < len and i-z[i]-1 >= 0 and
                s[i+z[i]+1] == s[i-z[i]-1]) z[i] ++;
        if (z[i] + i > z[b] + b) b = i;
    }
}
```

7.7 palindromic tree

```cpp
//bcw0x1bd2 {{{
#include<bits/stdc++.h>
#include<unistd.h>
using namespace std;
#define F first
#define S second
#define MP make_pair
#define PB push_back
#define IOS ios_base::sync_with_stdio(0); cin.tie(0);
#define SZ(x) ((int)((x).size()))
#define ALL(x) begin(x),end(x)
#define REP(i,x) for (int i=0; i<(x); i++)
#define REP1(i,a,b) for (int i=(a); i<=(b); i++)

typedef long long ll;
typedef pair<int,int> pii;
typedef pair<ll,ll> pll;
typedef long double ld;

#ifdef DARKHH
#define FILEIO(name)
#else
#define FILEIO(name) \
  freopen(name".in", "r", stdin); \
  freopen(name".out", "w", stdout);
#endif

#ifdef DARKHH
template<typename T>
void _dump( const char* s, T&& head ) {
    cerr<<s<<"="<<head<<endl; }

template<typename T, typename... Args>
void _dump( const char* s, T&& head, Args&&... tail
    ) {
  int c=0;
  while ( *s!=',' || c!=0 ) {
    if ( *s=='(' || *s=='[' || *s=='{' ) c++;
    if ( *s==')' || *s==']' || *s=='}' ) c--;
    cerr<<*s++;
  }
  cerr<<"="<<head<<",␣";
  _dump(s+1,tail...);
}

#define dump(...) do { \
  fprintf(stderr, "%s:%d␣-␣", __PRETTY_FUNCTION__,
      __LINE__); \
  _dump(#__VA_ARGS__, __VA_ARGS__); \
} while (0)

template<typename Iter>
ostream& _out( ostream &s, Iter b, Iter e ) {
  s<<"[";
  for ( auto it=b; it!=e; it++ ) s<<(it==b?"":":␣
      ")<<*it;
  s<<"]";
  return s;
}
```

```cpp
template<typename A, typename B>
ostream& operator <<( ostream &s, const pair<A,B> &p
    ) { return s<<"("<<p.first<<","<<p.second<<")"; }
template<typename T>
ostream& operator <<( ostream &s, const vector<T> &c
    ) { return _out(s,ALL(c)); }
template<typename T, size_t N>
ostream& operator <<( ostream &s, const array<T,N>
    &c ) { return _out(s,ALL(c)); }
template<typename T>
ostream& operator <<( ostream &s, const set<T> &c )
    { return _out(s,ALL(c)); }
template<typename A, typename B>
ostream& operator <<( ostream &s, const map<A,B> &c
    ) { return _out(s,ALL(c)); }
#else
#define dump(...)
#endif
// }}}

struct palindromic_tree{
  struct node{
    int next[26],fail,len;
    int cnt,num,st,ed;
    node(int l=0):fail(0),len(l),cnt(0),num(0){
      for(int i=0;i<26;++i)next[i]=0;
    }
  };
  vector<node> state;
  vector<char> s;
  int last,n;

  void init(){
    state.clear();
    s.clear();
    last=1;
    n=0;
    state.push_back(0);
    state.push_back(-1);
    state[0].fail=1;
    s.push_back(-1);
  }
  int get_fail(int x){
    while(s[n-state[x].len-1]!=s[n])x=state[x].fail;
    return x;
  }
  void add(int c){
    s.push_back(c-='a');
    ++n;
    int cur=get_fail(last);
    if(!state[cur].next[c]){
      int now=state.size();
      state.push_back(state[cur].len+2);
      state[now].fail =
          state[get_fail(state[cur].fail)].next[c];
      state[cur].next[c]=now;
      state[now].num=state[state[now].fail].num+1;
    }
    last=state[cur].next[c];
    ++state[last].cnt;
  }
  int size(){
    return state.size()-2;
  }
}pt;

int main() {
  string s;
  cin >> s;
  pt.init();
  for (int i=0; i<SZ(s); i++) {
    int prvsz = pt.size();
    pt.add(s[i]);
    if (prvsz != pt.size()) {
      int r = i;
      int l = r - pt.state[pt.last].len + 1;
```

```cpp
        cout << "Find␣pal␣@␣[" << l << "␣" << r << "]␣
            :␣" << s.substr(l,r-l+1) << endl;
    }
  }

  return 0;
}
```

7.8 Lexicographically Smallest Rotation

```cpp
string mcp(string s){
  int n = s.length();
  s += s;
  int i=0, j=1;
  while (i<n && j<n){
    int k = 0;
    while (k < n && s[i+k] == s[j+k]) k++;
    if (s[i+k] <= s[j+k]) j += k+1;
    else i += k+1;
    if (i == j) j++;
  }
  int ans = i < n ? i : j;
  return s.substr(ans, n);
}
```

7.9 Suffix Automaton

```cpp
// par : fail link
// val : a topological order ( useful for DP )
// go[x] : automata edge ( x is integer in [0,26) )

struct SAM{
  struct State{
    int par, go[26], val;
    State () : par(0), val(0){ FZ(go); }
    State (int _val) : par(0), val(_val){ FZ(go); }
  };
  vector<State> vec;
  int root, tail;

  void init(int arr[], int len){
    vec.resize(2);
    vec[0] = vec[1] = State(0);
    root = tail = 1;
    for (int i=0; i<len; i++)
      extend(arr[i]);
  }
  void extend(int w){
    int p = tail, np = vec.size();
    vec.PB(State(vec[p].val+1));
    for ( ; p && vec[p].go[w]==0; p=vec[p].par)
      vec[p].go[w] = np;
    if (p == 0){
      vec[np].par = root;
    } else {
      if (vec[vec[p].go[w]].val == vec[p].val+1){
        vec[np].par = vec[p].go[w];
      } else {
        int q = vec[p].go[w], r = vec.size();
        vec.PB(vec[q]);
        vec[r].val = vec[p].val+1;
        vec[q].par = vec[np].par = r;
        for ( ; p && vec[p].go[w] == q; p=vec[p].par)
          vec[p].go[w] = r;
      }
    }
    tail = np;
  }
};
```

8 Problems

8.1 Mo-Algorithm on Tree

```cpp
#include<bits/stdc++.h>
using namespace std;
#define IOS ios_base::sync_with_stdio(0); cin.tie(0);
#define SZ(x) ((int)((x).size()))

const int MX = 500005;
const int SQ = 1400;
const int LOG = 17;

struct BIT {
  int bit[MX];
  int lb(int x) { return x & -x; }
  void add(int p, int v) {
    p++;
    for (int i=p; i<MX; i+=lb(i)) bit[i] += v;
  }
  int qry() {
    int v = 0;
    for (int i=1<<LOG; i>0; i>>=1) {
      if ((v|i) < MX and bit[v|i]==i) v |= i;
    }
    return v;
  }
}bit;

struct Query {
  int l,r,qid;
}qry[MX];
struct Edge {
  int v,x;
};

int N,Q,timestamp[MX],ans[MX];
int in[MX],cnt[MX];
vector<Edge> E[MX];
vector<Edge> seq;

void DFS(int u, int f) {
  timestamp[u] = SZ(seq);
  for (auto it:E[u]) {
    if (it.v == f) continue;
    seq.push_back(it);
    DFS(it.v,u);
    seq.push_back(it);
  }
}
void poke(int id) {
  int v = seq[id].v;
  int x = seq[id].x;
  in[v] ^= 1;
  cnt[x] += in[v] ? 1 : -1;
  if (in[v] and cnt[x] == 1) bit.add(x, 1);
  if (!in[v] and cnt[x] == 0) bit.add(x, -1);
}
int main() {
  IOS;
  cin >> N >> Q;
  for (int i=0; i<N-1; i++) {
    int u,v,x;
    cin >> u >> v >> x;
    x = min(x,N);
    E[u].push_back({v,x});
    E[v].push_back({u,x});
  }
  DFS(1,1);
  for (int i=1; i<=Q; i++) {
    int u,v;
    cin >> u >> v;
    int l = timestamp[u], r = timestamp[v];
    if (l > r) swap(l,r);
    r--;
    qry[i] = {l,r,i};
  }
  sort(qry+1,qry+1+Q, [](Query a, Query b) {
      return make_pair(a.l/SQ,a.r) <
        make_pair(b.l/SQ,b.r);
      });
```

```cpp
  int curL = 1, curR = 0;
  for (int i=1; i<=Q; i++) {
    int ql=qry[i].l,qr=qry[i].r;
    while (curL > ql) poke(--curL);
    while (curR < qr) poke(++curR);
    while (curL < ql) poke(curL++);
    while (curR > qr) poke(curR--);
    ans[qry[i].qid] = bit.qry();
  }

  for (int i=1; i<=Q; i++) cout << ans[i] << "\n";

  return 0;
}
```

8.2 Manhattan MST

```cpp
#include<bits/stdc++.h>
#define REP(i,n) for(int i=0;i<n;i++)
using namespace std;
typedef long long LL;
const int N=200100;
int n,m;
struct PT {int x,y,z,w,id;}p[N];
inline int dis(const PT &a,const PT &b){return
    abs(a.x-b.x)+abs(a.y-b.y);}
inline bool cpx(const PT &a,const PT &b){return
    a.x!=b.x? a.x>b.x:a.y>b.y;}
inline bool cpz(const PT &a,const PT &b){return
    a.z<b.z;}
struct E{int a,b,c;}e[8*N];
bool operator<(const E&a,const E&b){return a.c<b.c;}
struct Node{
  int L,R,key;
}node[4*N];
int s[N];
int F(int x){return s[x]==x?x:s[x]=F(s[x]);}
void U(int a,int b){s[F(b)]=F(a);}
void init(int id,int L,int R) {
  node[id]=(Node){L,R,-1};
  if(L==R)return;
  init(id*2,L,(L+R)/2);
  init(id*2+1,(L+R)/2+1,R);
}
void ins(int id,int x) {
  if(node[id].key==-1 ||
      p[node[id].key].w>p[x].w)node[id].key=x;
  if(node[id].L==node[id].R)return;
  if(p[x].z<=(node[id].L+node[id].R)/2)ins(id*2,x);
  else ins(id*2+1,x);
}
int Q(int id,int L,int R){
  if(R<node[id].L || L>node[id].R)return -1;
  if(L<=node[id].L && node[id].R<=R)return
      node[id].key;
  int a=Q(id*2,L,R),b=Q(id*2+1,L,R);
  if(b==-1 || (a!=-1 && p[a].w<p[b].w)) return a;
  else return b;
}
void calc() {
  REP(i,n) {
    p[i].z=p[i].y-p[i].x;
    p[i].w=p[i].x+p[i].y;
  }
  sort(p,p+n,cpz);
  int cnt=0,j,k;
  for(int i=0;i<n;i=j){
    for(j=i+1;p[j].z==p[i].z && j<n;j++);
    for(k=i,cnt++;k<j;k++)p[k].z=cnt;
  }
  init(1,1,cnt);
  sort(p,p+n,cpx);
  REP(i,n) {
    j=Q(1,p[i].z,cnt);
    if (j != -1)
```

```cpp
      e[m++] = (E){p[i].id, p[j].id, dis(p[i],
          p[j])};
      ins(1,i);
    }
  }
}
LL MST() {
  LL r=0;
  sort(e,e+m);
  REP(i,m) {
    if(F(e[i].a)==F(e[i].b))continue;
    U(e[i].a,e[i].b);
    r+=e[i].c;
  }
  return r;
}
int main(){
  int ts;
  scanf("%d", &ts);
  while (ts--) {
    m = 0;
    scanf("%d",&n);
    REP(i,n) {
      scanf("%d%d",&p[i].x,&p[i].y);
      p[i].id=s[i]=i;
    }
    calc();
    REP(i,n)p[i].y= -p[i].y;
    calc();
    REP(i,n)swap(p[i].x,p[i].y);
    calc();
    REP(i,n)p[i].x=-p[i].x;
    calc();
    printf("%lld\n",MST()*2);
  }
  return 0;
}
```

9 Miscellany

9.1 tabi no hidarite saihate no migite

tabi no hidarite　　saihate　no migite
旅の左手、最果ての右手

sora ni　　ukannderu　　hikaru nami o
空に浮かんでる光る波を
tabanete niji no hasi wo　　kakeyou
束ねて虹の橋をかけよう
ayaui　　asiba　　suberu　　suro-pu
危うい足場　滑るスロープ
kako　to　mirai　no mitisirube
過去と未来の道標

kimi no hidarite　　boku no migite wo
君の左手　僕の右手を
tunaide　tunagete　　hazimeyou
繋いで繋げてはじめよう
itumo　　soba ni　　iru　yo
いつも　そばに　いるよ
kako　mo　ima mo　　mirai　mo
過去も　今も　未来も
dakara　mayowazu ni　aruite　ikou
だから迷わずに歩いていこう

irotoridori　　no yume　　kasanete
色とりどりの夢　重ねて
asita　ha　doko　ni　mukau
明日はどこに向かう
kimi to issyo ni　iretara　　sekai　ha　kagayakidasu　yo
君と一緒にいれたら　世界は輝きだすよ
saihate　no ti ni　saku hana wo　sagasou
最果ての地に咲く花を探そう

9.2 Made in Abyss