Cornell University Team Reference Document

Contents

1 CopyFirst

1.1 template

```
typedef long long ll;
typedef vector<int> vi;
typedef pair<int,int> pii;
typedef pair<double, double> pdd;
#define pb push_back
#define mp make_pair
#define fs first
#define sc second
#define rep(i, from, to) for (int i = from; i < (to); ++i)
#define all(x) x.begin(), x.end()
#define sz(x) (int)(x).size()
#define FOR(i, to) for (int i = 0; i < (to); ++i)
typedef vector<vector<int> > vvi;
typedef vector<ll> vll;
typedef vector<vll> vvll;
typedef vector<pair<int, int> > vpi;
typedef pair<ll,ll> pll;
typedef vector<string> vs;
```

2 Data Structures

2.1 fenwick

```
int AIB[Nmax], N;
inline int zeros(int x) { return x & (-x); }
inline void add(int x, int q) {
  for (int i = x; i <= N; i += zeros(i)) AIB[i]+=q;
}
inline int comp(int x) {
  int ret = 0;
  for (int i = x; i > 0; i -= zeros(i)) ret += AIB[i];
  return ret;
}
```

2.2 kdTree

```
// number type for coordinates, and its maximum value
typedef long long ntype;
const ntype inf = numeric_limits<ntype>::max();
struct point {
  ntype x, y;
  point(ntype xx = 0, ntype yy = 0) : x(xx), y(yy) {}
};
bool operator==(const point &a, const point &b) {
  return a.x == b.x && a.y == b.y;
}
int cmpx(const point &a, const point &b) { return a.x < b.x; }
int cmpy(const point &a, const point &b) { return a.y < b.y; }
ntype dist(const point &a, const point &b) {
  ntype dx = a.x-b.x, dy = a.y-b.y;
  return dx*dx + dy*dy;
}
struct bbox { //bounding box
  ntype x0, x1, y0, y1;
  bbox() : x0(inf), x1(-inf), y0(inf), y1(-inf) {}
  void compute(const vector<point> &v) {
    for (int i = 0; i < v.size(); ++i) {
      x0 = min(x0, v[i].x); x1 = max(x1, v[i].x);
      y0 = min(y0, v[i].y); y1 = max(y1, v[i].y);
    }
  }
  //distance between a point and this box, 0 if inside
  ntype distance(const point &p) {
    if (p.x < x0) {
      if (p.y < y0) return dist(point(x0, y0), p);
      else if (p.y > y1) return dist(point(x0, y1), p);
      else return dist(point(x0, p.y), p);
    } else if (p.x > x1) {
      if (p.y < y0) return dist(point(x1, y0), p);
      else if (p.y > y1) return dist(point(x1, y1), p);
      else return dist(point(x1, p.y), p);
    } else {
      if (p.y < y0) return dist(point(p.x, y0), p);
      else if (p.y > y1) return dist(point(p.x, y1), p);
      else return 0;
    }
  }
};
struct kdnode {
  bool leaf; point pt; //if only has one point
```

```
  bbox bound;
  kdnode *left, *right; // two children of this kd-node
  kdnode() : leaf(false), left(0), right(0) {}
  ~kdnode() { if (left) delete left; if (right) delete right; }
  ntype intersect(const point &p) { return bound.distance(p); }
  void construct(vector<point> &vp) {
    bound.compute(vp);
    if (vp.size() == 1) {
      leaf = true; pt = vp[0];
    } else { // split on largest coord
      if (bound.x1-bound.x0 >= bound.y1-bound.y0) {
        sort(vp.begin(), vp.end(), cmpx);
      } else {
        sort(vp.begin(), vp.end(), cmpy);
      }
      int half = vp.size()/2;
      vector<point> vl(vp.begin(), vp.begin()+half);
      vector<point> vr(vp.begin()+half, vp.end());
      left = new kdnode(); left->construct(vl);
      right = new kdnode(); right->construct(vr);
    }
  }
};
struct kdtree {
  kdnode *root;
  kdtree(const vector<point> &vp) {
    vector<point> v(vp.begin(), vp.end());
    root = new kdnode(); root->construct(v);
  }
  ~kdtree() { delete root; }
  ntype search(kdnode *node, const point &p) {// nearest neighbour
    if (node->leaf) {
      if (p == node->pt) return inf; // if not want dupl
      else return dist(p, node->pt);
    }
    ntype bleft = node->left->intersect(p);
    ntype bright = node->right->intersect(p);
    if (bleft < bright) {
      ntype best = search(node->left, p);
      if (bright < best) best = min(best, search(node->right, p));
      return best;
    } else {
      ntype best = search(node->right, p);
      if (bleft < best) best = min(best, search(node->left, p));
      return best;
    }
  }
  ntype nearest(const point &p) { return search(root, p); }
};
```

2.3 rmq

```
int rmq[log][Nmax], v[Nmax], lg[Nmax], N;
void genRmq() {
  for(int i=2;i<=N;++i) lg[i] = lg[i/2]+1;
  for(int i=1;i<=N;++i) rmq[0][i] = v[i];
  for(int i=1;(1<<i)<=N;++i) {
    for(int j=1;j+(1<<i)-1<=N;++j) {
      rmq[i][j] = min(rmq[i-1][j],rmq[i-1][j+ (1<<(i-1))]);
    }
  }
}
int query(int x,int y) {
    int l = lg[y-x+1], sh = y-x+1-(1<<l);
    return min(rmq[l][x], rmq[l][x+sh]);
}
```

2.4 rmq2D

```
int A[Nmax][Nmax], N, M;
int rmq[lmax][lmax][Nmax][Nmax],lg[Nmax];
void genRmq() {
  for(int i=2;i<=max(N,M);++i) lg[i] = lg[i/2] + 1;
  for(int k=0;(1<<k)<=N;++k) {
    for(int l=0;(1<<l)<=M;++l) {
      for(int i=1;i+(1<<k)-1<=N;++i) {
        for(int j=1;j+(1<<l)-1<=M;++j) {
          if(!k && !l) rmq[k][l][i][j] = A[i][j];
          else if(!k) rmq[k][l][i][j] = min(rmq[k][l-1][i][j],
                                rmq[k][l-1][i] [j+(1<<(l-1))]);
          else rmq[k][l][i][j] = min(rmq[k-1][l][i][j],
                                rmq[k-1][l][i+(1<<(k-1))][j]);
        }
      }
    }
  }
```

```
}
int query(int xa, int ya, int xb, int yb) {//(xa,ya)topleft (xb,yb)
    bottomright
  int lx = lg[xb-xa+1], mx = xb+1-(1<<lx);
  int ly = lg[yb-ya+1], my = yb+1-(1<<ly);
  int ret1 = min(rmq[lx][ly][xa][ya], rmq[lx][ly][xa][my]);
  int ret2 = min(rmq[lx][ly][mx][ya], rmq[lx][ly][mx][my]);
  return min(ret1,ret2);
}
```

2.5 segTree

```
int t[4*Nmax];
void update(int nod, int st, int dr, int poz, int val) {
  if(st==dr) { t[nod]=val; return; }
  int mij = (st+dr)/2;
  if(poz<=mij) update(2*nod,st,mij,poz,val);
  else update(2*nod+1,mij+1,dr,poz,val);
  t[nod] = max(t[2*nod],t[2*nod+1]);
}
int getmax(int nod, int st, int dr, int l, int r) {
  if(l<=st && dr<=r) return t[nod];
  int ret = 0; int mij = (st+dr)/2;
  if(r>mij) ret = max(ret,getmax(nod*2+1,1+mij,dr,l,r));
  if(l<=mij) ret = max(ret,getmax(nod*2,st,mij,l,r)); return ret;
}
```

2.6 segTree2D

```
int N, val[4*Nmax];
vector<ll> AIB[4*Nmax];
vector<ll> v[4*Nmax];
inline int zeros(int x) { return x & (-x); }
inline void Add(int nod, int x, int q){
  for(int i=x ;i < AIB[nod].size(); i+= zeros(i)) AIB[nod][i] += q;
}
ll comp(int nod, int x){
  ll ret = 0;
  if(AIB[nod].size() == 0) return 0;
  for(int i=x;i>0;i-=zeros(i)) ret += AIB[nod][i];
  return ret;
}
void init(int nod, int st, int dr){
  if(dr - st == 0){
    v[nod].pb(val[st]);
    AIB[nod].resize(v[nod].size()+1);
    Add(nod,1,1);
  } else {
    int mij = (st+dr) / 2;
    init(nod*2, st, mij); init(nod*2+1, mij+1, dr);
    v[nod].resize(v[nod*2].size() + v[nod*2+1].size());
    merge(v[nod*2].begin(), v[nod*2].end(), v[nod*2 +1].begin(),
        v[nod*2+1].end(),v[nod].begin());
    AIB[nod].resize(v[nod].size()+1);
    for(int i=1;i<AIB[nod].size();++i) Add(nod, i, 1);
  }
}
int getInd(int nod, int x){
  int ret = -1, st = 0, dr = v[nod].size() -1;
  while(st <= dr ){
    int mij = (st+dr)/2;
    if(v[nod][mij] <= x) { st = mij + 1; ret = mij; }
    else dr = mij - 1;
  }
  return ret+1;
}
int K; ll retV = 0; // calc between ist,idr and k,l
int calc(int nod,int ist, int idr, int st, int dr, int k, int l) {
  if (K == 0 && k > l) return 0;
  if(ist <= st && idr >= dr){
    retV += comp(nod,getInd(nod,l)) - comp(nod,getInd(nod,k-1));
  } else {
    int mij = (st+dr)/2;
    if (ist <= mij) calc(nod*2,ist,idr,st,mij,k,l);
    if (idr > mij) calc(nod*2+1,ist,idr,mij+1,dr,k,l);
  }
}
```

2.7 segTreeLazy

```
int aint[Nmax*4], up[Nmax*4], v[Nmax];
inline void relax(int nod,int st,int dr) {
  if(!up[nod]) return; aint[nod] += up[nod];
  if(st!=dr) { up[2*nod]+=c; up[2*nod+1]+=c; }
```

```
  up[nod] = 0;
}
void update(int nod,int ist,int idr,int st,int dr,ll val) {
  relax(nod,st,dr);
  if(ist<=st&&idr>=dr) {
    up[nod] = val; relax(nod,st,dr);
  } else {
    int mij=(st+dr)/2;
    if(ist<=mij) update(2*nod,ist,idr,st,mij,val);
    if(idr>mij) update(2*nod+1,ist,idr,mij+1,dr,val);
    make(nod,st,dr);
  }
}
int calc(int nod,int ist,int idr,int st,int dr) {
  relax(nod,st,dr); int ret = 0;
  if(ist<=st && idr>=dr) ret += aint[nod];
  else {
    ll mij=(st+dr)/2;
    if(ist<=mij) ret += calc(2*nod,ist,idr,st,mij);
    if(idr>mij) ret += calc(2*nod+1,ist,idr,mij+1,dr);
    make(nod,st,dr);
  }
  return ret;
}
void init(int nod,int st,int dr){
  if(st != dr){
    int mij = (st+dr)/2;
    init(2*nod,st,mij); init(2*nod+1,mij+1,dr);
    make(nod,st,dr);
  } else {
    aint[nod] = v[st];
  }
}
```

2.8 treap

```
typedef struct item * pitem;
struct item {
  int cnt, value, prior, key; bool rev; pitem l, r;
  item(int prior, int value) : cnt(1), rev(false), prior(prior),
      value(value), l(NULL), r(NULL) {}
};
int cnt(pitem t) { return t ? t->cnt : 0; }
void upd_cnt(pitem it) { if (it) it->cnt = cnt(it->l) + cnt(it->r) + 1; }
void push(pitem it) {
  if (it && it->rev) {
    it->rev = false; swap(it->l, it->r);
    if (it->l) it->l->rev ^= true;
    if (it->r) it->r->rev ^= true;
  }
}
void merge(pitem &t, pitem l, pitem r) {
  push(l); push(r); if (!l || !r) t = l ? l : r;
  else if (l->prior > r->prior) merge(l->r, l->r, r), t = l;
  else merge(r->l, l, r->l), t = r; upd_cnt(t);
}
void split(pitem t, pitem & l, pitem & r, int key, int add = 0) {
  if (!t) return void (l = r = 0);
  push (t); int cur_key = cnt(t->l) + add;
  if (key <= cur_key) split(t->l, l, t->l, key, add), r = t;
  else split(t->r, t->r, r, key, add + cnt(t->l) + 1), l = t;
  upd_cnt(t);
}
void split (pitem t, int key, pitem & l, pitem & r) {
  if (!t) l = r = NULL; push(t);
  else if (key < t->key) split (t->l, key, l, t->l), r = t;
  else split (t->r, key, t->r, r), l = t; upd_cnt(t);
}
void reverse(pitem t, int l, int r) {
  if(l > r) return; pitem t1, t2, t3;
  split(t, t1, t2, l); split(t2, t2, t3, r-l+1);
  t2->rev ^= true; merge(t, t1, t2); merge(t, t, t3);
}
void output (pitem t) {
  if (!t) return; push (t); output (t->l);
  printf ("%d", t->value); output (t->r);
}
void erase (pitem & t, int key) {
  push(t); if (t->key == key) merge (t, t->l, t->r);
  else erase (key < t->key ? t->l : t->r, key); upd_cnt(t);
}
void insert (pitem & t, pitem it) {
  push(t); if (!t) { t = it; return;};
  if (it->prior > t->prior) split (t, it->key, it->l, it->r), t = it;
  else insert (it->key < t->key ? t->l : t->r, it);
}
pitem cur = new item(rand(), s[i] - 'a'); if (root) merge(root, root,
    cur);
```

```
else root = cur; reverse(root, x,y);
```

2.9 unionFind

```
int par[Nmax],h[Nmax]
int findx(int x) {
  int R = x, y; while(par[R] != R) R = par[R];
  while(par[x] != x) { y = par[x]; par[x] = R; x = y;}
  return R;
}
void unite(int x, int y) {
  x = findx(x); y = findx(y); if (x == y) return;
  if(h[x] > h[y]) { par[y] = x; h[x] += h[y]; }
  else { par[x] = y; h[y] += h[x]; }
}
```

3 Graph

3.1 2sat

```
int N,s[2*Nmax],curr,c[2*Nmax],sol[2*Nmax]; //value of i = sol[2*i]
vector<int> g[2*Nmax],gt[2*Nmax],v[2*Nmax];
int viz[2*Nmax],vz[2*Nmax];
void dfs(int x) {
  viz[x] = 1; for(auto y: g[x]) if(!viz[y]) dfs(y); s[++curr] = x;
}
void dfs2(int x, int comp) {
  viz[x] = 0; c[x] = comp; v[comp].push_back(x);
  for(auto y: gt[x]) if(viz[y]) dfs2(y,comp);
}
inline int ng(int x) { if(x%2) return x-1; return x+1;}
bool f(int x, int val) {
  vz[x] = 1;
  for(auto y: v[x]) {
    if(sol[y] && sol[y]!=val) return false; sol[y] = val;
  }
  for(auto y: v[x]) {
    y = ng(y); if(sol[y] && sol[y]!=3-val) return false;
    if(!sol[y]) return f(c[y],3-val);
  }
  return true;
}
inline bool sat() {
  int comp = 0;
  for(int i=2;i<=2*N+1;++i) if(!viz[i]) dfs(i);
  for(int i=curr;i>=1;--i) if(viz[s[i]]) dfs2(s[i],++comp);
  for(int i=1;i<=comp;++i) if(!vz[i]) if(!f(i,1)) return false;
  return true;
}
inline void add_disj(int x, int sx, int y, int sy) {//s 0 normal,1
    negation
  g[2*x+(1-sx)].push_back(2*y+sy);
  g[2*y+(1-sy)].push_back(2*x+sx);
  gt[2*y+sy].push_back(2*x+(1-sx));
  gt[2*x+sx].push_back(2*y+(1-sy));
}
```

3.2 bellman

```
vector<pii> g[Nmax];
int N,cnt[Nmax],d[Nmax];
queue<int> q;
void bellman(int r) {
  for(int i=1;i<=N;++i) d[i] = inf; d[r] = 0; q.push(r);
  while(!q.empty()) {
    int x = q.front(); ++cnt[x];
    if(cnt[x] > N) return; q.pop();
    for(auto p: g[x]) {
      int y = p.fs, c = p.sc;
      if(d[y] > d[x] + c) { d[y] = d[x] + c; q.push(y); }
    }
  }
}
```

3.3 biconnected

```
int N,w[Nmax],low[Nmax],depth[Nmax],comp,viz[Nmax];
vector<pii> m; //edges stack
vector<vi> c; //result
vi g[Nmax], com; //adjancency list
void dfs(int x, int p, int dep) {
```

```
  viz[x] = 1; depth[x]=dep; low[x]=dep;
  for(auto y: g[x]) {
    if(!viz[y]) {
      m.pb(mp(x,y)); dfs(y,x,dep+1); low[x] = min(low[x],low[y]);
      if(low[y] >= depth[x]) {
        ++comp; com.clear();
        while(true) {
          int t = m.back().fs, u = m.back().sc;
          if(w[t] != comp) { w[t] = comp; com.pb(t); }
          if(w[u] != comp) { w[u] = comp; com.pb(u); }
          m.pop_back(); if(t==x && u==y) break;
        }
        c.pb(com);
      }
    } else if(y!=p) low[x]=min(low[x],depth[y]);
  }
}
void biconnect() {
  for(int i=1;i<=N;++i) {
    if(!viz[i]) dfs(i,0,0);
  }
}
```

3.4 dijkstra

```
int N, d[Nmax], viz[Nmax];
vector<pii> g[Nmax];
priority_queue<pii> pq;
void dijkstra(int r) {
  for(int i=1;i<=N;++i) d[i] = inf;
  d[r] = 0; pq.push(mp(0,r));
  while(!pq.empty()) {
    int x = pq.top().sc; pq.pop(); viz[x] = 1;
    for(auto a: g[x]) {
      int y = a.fs, c = a.sc;
      if(!viz[y]) {
        if(d[y] > c + d[x]) {
          d[y] = c + d[x]; pq.push(mp(-d[y],y));
        }
      }
    }
  }
}
```

3.5 dinic

```
struct Edge {
  int u, v; ll cap, flow; Edge() {}
  Edge(int u, int v, ll cap): u(u), v(v), cap(cap), flow(0) {}
};
int N; vector<Edge> E;
vi d, pt, g[Nmax]; //edge indices
inline void addEdge(int u, int v, int cap) {
  E.pb(Edge(u, v, cap)); g[u].pb(E.size() - 1);
  E.pb(Edge(v, u, 0)); g[v].pb(E.size() - 1);
}
inline int BFS(int S, int T) {
  queue<int> q; q.push(S);
  fill(d.begin(), d.end(), N + 1);
  d[S] = 0;
  while(!q.empty()) {
    int u = q.front(); q.pop();
    if (u == T) break;
    for (int k: g[u]) {
      Edge &e = E[k];
      if (e.flow < e.cap && d[e.v] > d[e.u] + 1) {
        d[e.v] = d[e.u] + 1; q.push(e.v);
      }
    }
  }
  return d[T] != N + 1;
}
ll DFS(int u, int T, ll flow = -1) {
  if (u == T || flow == 0) return flow;
  for (int &i = pt[u]; i < g[u].size(); ++i) {
    Edge &e = E[g[u][i]];
    Edge &oe = E[g[u][i]^1];
    if (d[e.v] == d[e.u] + 1) {
      ll amt = e.cap - e.flow;
      if (flow != -1 && amt > flow) amt = flow;
      if (ll pushed = DFS(e.v, T, amt)) {
        e.flow += pushed; oe.flow -= pushed; return pushed;
      }
    }
  }
  return 0;
}
ll MaxFlow(int S, int T) {
  ll total = 0; pt.resize(N); d.resize(N);
```

```
  while (BFS(S, T)) {
    fill(pt.begin(), pt.end(), 0);
    while (ll flow = DFS(S, T)) total += flow;
  }
  return total;
}
```

3.6 eulerCycle

```
void dfs(int x) { //isD = isDeleted
  for(auto n : g[x]) {
    if(isD[n.sc]==0) { isD[n.sc]=1; dfs(n.fs); }
  }
  ret.pb(x);
} // fs = node, sc = edge num
```

3.7 flowMinCost

```
int NRN,rez,d[Nmax],p[Nmax],viz[Nmax],inq[Nmax];
vector<pii> mc;
vi v,f,c, m[Nmax];
queue<int> q;
inline void add(int x) {
  if(inq[x]) return; inq[x] = viz[x] = 1; q.push(x);
}
inline int pop() {
  int x = q.front(); q.pop(); inq[x] = 0; // delete for bfs
  return x;
}
inline void reset_stuff(int S) {
  for(int i = 0; i <= NRN; ++i) {
    viz[i] = inq[i] = 0; d[i] = inf;
  } d[S] = 0;
}
inline void addEdge(int x,int y,int cap, int cost) {
  NRN = max(NRN,x); NRN = max(NRN,y);
  c.pb(cap); v.pb(cost); f.pb(0);
  m[x].pb(sz(mc)); mc.pb(mp(x,y));
  c.pb(0); f.pb(0); v.pb(-cost);
  m[y].pb(sz(mc)); mc.pb(mp(y,x));
}
```

```
inline int bfs(int S, int D) {
  reset_stuff(S); add(S);
  while(!q.empty()) {
    int x = pop(); if(x==D) continue;
    for(auto y : m[x]) {
      int ve = mc[y].sc;
      if(f[y] < c[y] && d[ve] > d[x] + v[y]) {
        add(ve); p[ve] = y; d[ve] = d[x] + v[y];
      }
    }
  }
  return viz[D];
}
pii update(int S, int D) {
  int ret = 0, retc = 0, flux = inf, curr = D;
  while(curr!=S) {
    int muc = p[curr], par = mc[p[curr]].fs;
    if(c[muc] - f[muc] < flux) flux = c[muc] - f[muc];
    if(!flux) break; curr = par;
  }
  curr = D;
  while(curr!=S) {
    int edg = p[curr], par = mc[p[curr]].fs;
    f[edg] += flux; f[edg^1] -= flux; curr = par;
  }
  ret += flux; retc += flux*d[D]; return mp(ret, retc);
}
pii flow(int S, int D) {
  int ret = 0, retc = 0;
  while(true) {
    if(!bfs(S, D)) break;
    pii u = update(S,D); ret += u.fs, retc += u.sc;
  }
  return mp(ret, retc);
}
```

3.8 heavyPath

```
int N,M,q,x,y,K;
int poz[Nmax],v[Nmax],nr[Nmax],l[Nmax],p[Nmax],cmp[Nmax],viz[Nmax];
vector<int> g[Nmax], c[Nmax];
//Segment tree stuff
```

```
vector<vector<int> > t;
void update(int comp, int nod, int st, int dr, int c, int d) {
  if(st==dr) { t[comp][nod]=d; return; }
  int mij = (st+dr)/2;
  if(c<=mij) update(comp,2*nod,st,mij,c,d);
  else update(comp,2*nod+1,mij+1,dr,c,d);
  t[comp][nod] = max(t[comp][2*nod],t[comp][2*nod+1]);
}
int getmax(int comp, int nod, int st, int dr, int c, int d) {
  int ret = 0;
  if(c<=st && dr<=d) return t[comp][nod];
  int mij = (st+dr)/2;
  if(d>mij) ret = max(ret,getmax(comp,nod*2+1,1+mij,dr,c,d));
  if(c<=mij) ret = max(ret,getmax(comp,nod*2,st,mij,c,d));
  return ret;
}
inline int query_val(int comp, int st, int dr) {
  return getmax(comp, 1, 1, c[comp].size(), st+1, dr+1);
}
//Available queries
inline void update_val(int comp, int poz, int val) {
  update(comp, 1, 1, c[comp].size(), poz+1, val);
  v[c[comp][poz]] = val;
}
int find_max(int x, int y) {
  if(cmp[x] == cmp[y]) return
      query_val(cmp[x],min(poz[x],poz[y]),max(poz[x],poz[y]));
  int px = p[c[cmp[x]][0]], py = p[c[cmp[y]][0]];
  if(l[px] < l[py]) { swap(x,y); swap(px,py); }
  int M = query_val(cmp[x],0,poz[x]);
  return max(M, find_max(px,y));
}
//Preprocessing
void dfs(int x) {
  viz[x] = 1; nr[x] = 1;
  int ind = -1, nrc = -1;
  for(auto y: g[x]) {
    if(viz[y]) continue;
    l[y] = l[x] + 1; p[y] = x; dfs(y);
    if(nr[y] > nrc) { ind = y; nrc = nr[y]; }
    nr[x] += nr[y];
  }
  if(nrc == -1) {
    vector<int> C; C.pb(x);
    ++K; c[K] = C; cmp[x] = K;
  } else {
    c[cmp[ind]].pb(x); cmp[x] = cmp[ind];
  }
}
void heavy_path(int r) {
  l[r] = 1; dfs(r);
  for(int i=1;i<=K;++i) {
    reverse(c[i].begin(),c[i].end());
    for(int j=0;j<c[i].size();++j) poz[c[i][j]] = j;
  }
  t.resize(K+10);
  for(int i=1;i<=K;++i) t[i].resize(4*c[i].size()+10,0);
  for(int i=1;i<=N;++i) update_val(cmp[i],poz[i],v[i]);
}
```

3.9 hungarian

```
int hungarian() { //given a[n][m] matrix of costs
  vi u (n+1), v (m+1), p (m+1), way (m+1), ans(n+1);
  for (int i=1; i<=n; ++i) {
    p[0] = i; int j0 = 0;
    vi minv (m+1, INF), used (m+1, 0);
    do {
      used[j0] = 1; int i0 = p[j0], delta = INF, j1;
      for (int j=1; j<=m; ++j) {
        if (!used[j]) {
          int cur = a[i0][j]-u[i0]-v[j];
          if (cur < minv[j]) minv[j] = cur, way[j] = j0;
          if (minv[j] < delta) delta = minv[j], j1 = j;
        }
      }
      for (int j=0; j<=m; ++j) {
        if (used[j]) u[p[j]] += delta, v[j] -= delta;
        else minv[j] -= delta;
      }
      j0 = j1;
    } while (p[j0] != 0);
    do {
      int j1 = way[j0]; p[j0] = p[j1]; j0 = j1;
    } while (j0);
  }
  for (int j=1; j<=m; ++j) ans[p[j]] = j;
  return -v[0];
```

```
}
```

3.10 lca

```
void dfs(int x, int lev) {
  e[++K] = x; L[K] = lev; poz[x] = K;
  for(auto y: g[x]) {
    dfs(y,lev+1); e[++K] = x; L[K] = lev;
  }
}
void preprocess_lca() {
  dfs(1,0);
  for(int i=2;i<=K;++i) Lg[i] = Lg[i/2]+1;
  for(int i=1;i<=K;++i) rmq[0][i]=i;
  for(int i=1;(1<<i) < K; ++i) {
    for(int j=1;j<=K-(1<<i);++j) {
      rmq[i][j] = rmq[i-1][j];
      if(L[rmq[i-1][j + (1<<(i-1))]] < L[rmq[i][j]]) {
        rmq[i][j] = rmq[i-1][j + (1<<(i-1))];
      }
    }
  }
}
int lca(int x, int y) {
  int a = poz[x], b = poz[y];
  if(a>b) swap(a,b);
  int l = Lg[b-a+1], sol = rmq[l][a];
  if(L[sol] > L[rmq[l][b - (1<<l) + 1]]) {
    sol = rmq[l][b - (1<<l) + 1];
  }
  return e[sol];
}
```

3.11 lcaLog

```
int par[Nmax][Lmax],N,M, lg[Nmax], lvl[Nmax];
vi g[Nmax];
void dfs(int nod, int lev){
  lvl[nod] = lev;
  for(auto x: g[nod])
  if(!lvl[x]) { par[x][0] = nod; dfs(x, lev+1); }
}
int lca(int x,int y){
  if(lvl[x] < lvl[y]) swap(x,y);
  int log1=1, log2=1;
  for(;(1<<log1) < lvl[x]; ++log1);
  for(;(1<<log2) < lvl[y]; ++log2);
  for(int k = log1; k >= 0; --k){
    if(lvl[x] - (1 << k) >= lvl[y]) x = par[x][k];
  }
  if (x == y) return x;
  for(int k=log2; k>=0 ;--k) {
    if(par[x][k] && par[x][k] != par[y][k]){
      x = par[x][k]; y = par[y][k];
    }
  }
  return par[x][0];
}
void preprocessLca() {
  dfs(1,1);
  for(int k=1; (1<<k) <= N; ++k){
    for(int i=1;i<=N;++i){
      par[i][k] = par[par[i][k-1]][k-1];
    }
  }
}
```

3.12 matching

```
int N,M,L,R,K,v[Nmax],p;
vi g[Nmax];
int l[Nmax],r[Nmax],u[Nmax],was[Nmax],S;
int match(int q) {
  if(was[q]) return 0; was[q]=1;
  for(auto x : g[q]) {
    if(!r[x]) { l[q]=x; r[x]=q; return 1;}
  }
  for(auto x: g[q]) {
    if(match(r[x])) { l[q]=x; r[x]=q; return 1; }
  }
  return 0;
}
void matching() { //edges i,l[i] if l[i]>0
  int ok = 1; while(ok) {
```

```
    ok = 0; for(int i=0;i<=L;++i) was[i]=0;
    for(int i=1;i<=L;++i) { if(!l[i]) ok|= match(i); }
  }
}
```

3.13 royFloyd

```
FOR(k,N) FOR(i,N) FOR(j,N) {
  if(best[i][k] && best[k][j] && i!=j &&
      (best[i][k]+best[k][j]<best[i][j] || !best[i][j])) {
     best[i][j]=best[i][k]+best[k][j];
  }
}
```

3.14 strongConnected

```
vi g[Nmax],stack,viz,low,iss,aux;
vector<vi> comp;
int k,index=1,N,M,x,y;
void df(int x){
  viz[x] = index; low[x] = index;
  stack[++k] = x; iss[x] = 1; ++index;
  for(auto n : g[x]){
    if(viz[n] == 0){
      df(n); low[x] = min(low[x],low[n]);
    } else if(iss[n]) low[x] = min(low[x],low[n]);
  }
  if(low[x] == viz[x]){
    aux.clear();
    do {
      aux.pb(stack[k]); iss[stack[k]] = 0; --k;
    } while(stack[k+1] != x);
    comp.pb(aux);
  }
}
void init(){
  stack.resize(N+10); viz.resize(N+10);
  iss.resize(N+10); low.resize(N+10);
  for(int i=1;i<=N;++i) {
    if(!viz[i]) df(i);
  }
}
```

4 Math

4.1 comb

```
ll MOD, inv[Nmax], fact[Nmax], ifact[Nmax];
ll c[nmax][nmax];
void make_comb(int N) { // N ~ 10^3
  for(int i=0;i<=N;++i) {
    c[i][0] = 1;
    for(int j=1;j<=i;++j) c[i][j] = (c[i-1][j] + c[i-1][j-1]) % MOD;
  }
}
void make_fact(int N) { // N ~ 10^6
  inv[1] = fact[0] = fact[1] = ifact[0] = ifact[1] = 1;
  for(int i=2;i<=N;++i) {
    inv[i] = (MOD - (MOD/i) * inv[MOD%i] % MOD) % MOD;
    fact[i] = (fact[i-1]*i) % MOD;
    ifact[i] = (ifact[i-1]*inv[i]) % MOD;
  }
}
ll comb(ll a, ll b) { // a,b ~ 10^6
  ll ret = (fact[a] * ifact[b]) % MOD;
  return (ret * ifact[a-b]) % MOD;
}
ll vp(ll x) { //exponent of MOD in x!
  ll z = MOD, ret = 0;
  while(z <= x) { ret += x/z; z *= MOD;}
  return ret;
}
ll f(ll x) { // x! % MOD if we ignore the MOD factors
  if(x < MOD) return fact[x];
  ll z = 1, k = 0;
  do { z *= MOD; ++k; } while (z <= x/MOD);
  ll ret = (fact[x/z] * f(x%z)) % MOD;
  if(k%2 && t%2) return (MOD-ret)%MOD;
  else return ret;
}
ll getComb(ll A, ll B) { // A, B ~ 10^18
  if(vp(A) > vp(B) + vp(A-B)) return 0;
  if(MOD==2) return 1;
```

```
  return (((f(A)*inv[f(B)])%MOD)*inv[f(A-B)])%MOD;
}
```

4.2 fft

```
const int MAX = 1<<20;
typedef int value;
typedef complex<double> comp;
int N, p[MAX];
comp omega[MAX], omega_tmp[MAX], a1[MAX], a2[MAX], z1[MAX], z2[MAX];
void fft(comp *a, comp *z) {
  FOR(i, N) z[i] = a[p[i]]; int t = N;
  for (int size = 1; size < N; size *= 2) {
    t /= 2; FOR(j,size) omega_tmp[j] = omega[j*t];
    for (int i = 0; i < N; i += 2*size) {
      FOR(j,size) {
        comp c = omega_tmp[j] * z[i+j+size];
        z[i+j+size] = z[i+j] - c; z[i+j] += c;
      }
    }
  }
}
void mult(value *a, value *b, value *c, int len) {
  N = 2*len; while (N & (N-1)) ++N;
  p[0] = 0; int k = 0, t = -1;
  while ((1 << k) < N) ++k;
  for(int i = 1; i < N; ++i) {
    if ((i&(i-1)) == 0) ++t;
    p[i] = p[i^(1<<t)]; p[i] |= 1 << (k-t-1);
  }
  FOR (i, N) { a1[i] = 0; a2[i] = 0; }
  FOR (i, len) { a1[i] = double(a[i]); a2[i] = double(b[i]); }
  FOR (i, N/2) {
    omega[i] = polar(1.0, 2*M_PI/N*i);
    omega[N-i-1] = conj(omega[i]);
  }
  fft(a1, z1); fft(a2, z2);
  FOR (i, N) omega[i] = conj(omega[i]);
  FOR (i, N) a1[i] = z1[i] * z2[i] / comp(N, 0);
  fft(a1, z1);
  FOR (i, 2*len) c[i] = value(round(z1[i].real()));
}
```

4.3 fftMOD

```
#define Pmax 19
#define MOD 5767169 //magic: MOD-1 must be multiple of 2^Pmax
#define GEN1 177147 //magic: root of order 2^Pmax mod MOD
#define GEN2 5087924 //magic: inverse of GEN1
#define Nmax 530000
vi v1,v2,B; int k=1, z[2][Nmax], b[Nmax];
int powy(int x, int y) {
  if(!y) return 1; int z = powy(x,y/2);
  z = (1LL*z*z) % MOD; if(y%2) z = (1LL*z*x) % MOD;
  return z;
}
//start: index, inc: divisions, rev0 fft , rev1 reverse fft
void fft(vector<int> &v, int start, int inc, int rev) {
  if(inc==k) return; //we done
  fft(v,start,inc*2,rev); //compute first half
  fft(v,start+inc,inc*2,rev); //compute second half
  int nr = k/inc, Z = 1, zN = z[rev][nr];
  for(int i=0;i<nr/2;++i) {
    int x = (1LL*Z*v[start + (2*i+1)*inc]) % MOD;
    b[start+i*inc] = (v[start + 2*i*inc] + x) % MOD;
    b[start+(i+nr/2)*inc] = (v[start + 2*i*inc] - x + MOD) % MOD;
    Z = (1LL*Z*zN)%MOD; //Z is current root
  }
  for(int i=0;i<nr;++i) {
    v[start+i*inc] = b[start+i*inc];
  }
}
void preprocess_fft(vector<int> &v1, vector<int> &v2) {
  int pw = 0; //smallest power of 2 greater than degree
  int N = v1.size(), M = v2.size(), deg = M+N;
  while(k<deg) { k*=2; ++pw; } //smallest pw2 >= final degree
  for(int i=N;i<k;++i) v1.pb(0);
  for(int i=M;i<k;++i) v2.pb(0);
  int r1 = GEN1, r2 = GEN2; //square until roots of order 2^pw
  for(int i=pw;i<Pmax;++i) {
    r1 = (1LL*r1*r1) % MOD; r2 = (1LL*r2*r2) % MOD;
  }
  for(int nr=k;nr>=1;nr/=2) { //primitives 2^nr
    z[0][nr] = r1; z[1][nr] = r2; // 1 / z0
    r1 = (1LL*r1*r1) % MOD; r2 = (1LL*r2*r2) % MOD;
  }
}
vector<int> multiply(vector<int> &v1, vector<int> &v2) {
```

```
  preprocess_fft(v1, v2);
  vi ret;
  fft(v1,0,1,0); fft(v2,0,1,0);
  for(int i=0;i<k;++i) ret.pb((1LL*v1[i]*v2[i])%MOD);
  fft(ret,0,1,1);
  int inv = powy(k,MOD-2);
  for(int i=0;i<k;++i) ret[i] = (1LL*ret[i]*inv) % MOD;
  return ret;
}
```

4.4 fraction

```
pll contToFrac(vector<ll> &a, int start) {
  if(start >= a.size()-1) return mp(a[start],1);
  pll p = contToFrac(a,start+1);
  return reduce(a[start] * p.fs + p.sc, p.fs);
}
vector<ll> fracToCont(pll frac) {
  vector<ll> ret; ll A = frac.fs, B = frac.sc;
  while(true) {
    if(B == 0 || (A < B && ret.size()!= 0) ) break; ret.pb(A/B);
    ll newA = B, newB = A%B; A = newA; B = newB;
  }
  return ret;
}
```

4.5 geom2D

```
#define PI 3.141592653589793
#define eps 0.00000001
int cmp(double a, double b) {
  if(a + eps <= b) return -1; if(a < b + eps) return 0; return 1;
}
int cmp(pdd a, pdd b) {
  int t = cmp(a.fs, b.fs); if(t) return t; return cmp(a.sc,b.sc);
}
pdd in = mp(INFINITY, INFINITY); //no solution
pdd operator+(pdd a, pdd b) { return mp(a.fs+b.fs, a.sc+b.sc); }
pdd operator-(pdd a, pdd b) { return mp(a.fs-b.fs, a.sc-b.sc); }
pdd operator*(pdd a, double t) { return mp(a.fs*t, a.sc*t); }
pdd operator/(pdd a, double t) { return mp(a.fs/t, a.sc/t); }
double operator*(pdd a, pdd b) { return a.fs*b.fs + a.sc*b.sc; }
double operator%(pdd a, pdd b) { return a.fs*b.sc - a.sc*b.fs; }
bool operator<(pdd a, pdd b) { return cmp(a,b) < 0; }
bool operator==(pdd a, pdd b) { return cmp(a,b) == 0; }
double norm(pdd a) { return sqrt(a*a); }
double arg(pdd a) { return atan2(a.sc,a.fs); }
inline double dist(pdd p, pdd q) { return norm(p-q);}
inline int ccw(pdd a, pdd b, pdd c) { return cmp((a-c)%(b-c),0);}
inline double angle(pdd a, pdd b, pdd c) { //[-PI/2, PI/2]
  pdd u = a - b, v = c - b; return atan2(u % v, u * v);
}
inline int between(pdd p, pdd q, pdd r) {
  return ccw(p,q,r) == 0 && cmp((p-q)*(r-q),0) <= 0;
}
int segInters(pdd p, pdd q, pdd r, pdd s) {
  pdd A = q-p, B = s-r, C = r-p, D = s-q;
  int a = cmp(A%C,0) + 2*cmp(A%D,0);
  int b = cmp(B%C,0) + 2*cmp(B%D,0);
  if(a==3 || a==-3 || b==3 || b==-3) return false;
  if (a || b || p == r || p == s || q == r || q == s) return true;
  int t = (p<r) + (p<s) + (q<r) + (q<s);
  return t!=0 && t!=4;
}
double distToSeg(pdd p, pdd q, pdd r) { // from r to pq
  pdd A = r - q, B = r - p, C = q - p;
  double a = A * A, b = B * B, c = C * C;
  if (cmp(b, a + c) >= 0) return sqrt(a);
  else if (cmp(a, b + c) >= 0) return sqrt(b);
  else return abs(A % B) / sqrt(c); //ONLY THIS IF WHOLE LINE
}
int inPoly(pdd p, vector<pdd> &T) { // -1 border, 0 outside, 1 inside
  double a = 0; int N = T.size();
  for (int i = 0; i < N; i++) {
    if (between(T[i], p, T[(i+1) % N])) return -1;
    a += angle(T[i], p, T[(i+1) % N]);
  }
  return cmp(a,0) != 0;
}
pdd lineInters(pdd p, pdd q, pdd r, pdd s) {
  if(cmp(p-q,s-r) == 0 || cmp(p-q,r-s) == 0) return mp(INFINITY,INFINITY);
  pdd a = q - p, b = s - r, c = mp(p % q, r % s);
  return mp(mp(a.fs, b.fs) % c, mp(a.sc, b.sc) % c) / (a % b);
}
pdd foot(pdd P, pdd A, pdd B) {
  pdd dir = B-A; pdd x = P-A;
```

```
  return (dir*((P-A)*dir))/(dir*dir) + A;
}
// Perp in Q on PQ at distance d
pair<pdd,pdd> disppt(pdd P, pdd Q, double d) {
  pdd dir = P - Q; pdd r = pdd(dir.sc, -dir.fs);
  pdd k = r * (d/norm(r)); return mp(Q + k, Q - k);
}
double area(vector<pdd> &v) {
  double ret = 0; int N = v.size();
  for(int i=0;i<N;++i) { ret += v[i]%v[(j+1)%N]; }
  return abs(ret)/2;
}
double getX(pdd v1, pdd v2, double t) { // find (x,t) on (v1,v2)
  return (t - v1.sc)*(v1.fs - v2.fs) / (v1.sc - v2.sc) + v1.fs;
}
double getY(pdd v1, pdd v2, double t) { // find (t,y) on (v1,v2)
  return (t - v1.fs)*(v1.sc - v2.sc) / (v1.fs - v2.fs) + v1.sc;
}
vector<pdd> polygIntersect(vector<pdd> &P, vector<pdd> &Q) {
  vector<pdd> R, nope;
  int m = Q.size(), n = P.size(); if (m == 0 || n == 0) return nope;
  int a = 0, b = 0, aa = 0, ba = 0, inflag = 0;
  while ((aa < n || ba < m) && aa < 2*n && ba < 2*m) {
    pdd p1 = P[a], p2 = P[(a+1) % n], q1 = Q[b], q2 = Q[(b+1) % m];
    pdd A = p2 - p1, B = q2 - q1;
    int cross = cmp(A % B, 0), ha = ccw(p2, q2, p1), hb = ccw(q2, p2, q1);
    if (cross == 0 && ccw(p1, q1, p2) == 0 && cmp(A * B,0) < 0) {
      if(between(p1, q1, p2)) R.pb(q1); if(between(p1, q2, p2)) R.pb(q2);
      if(between(q1, p1, q2)) R.pb(p1); if(between(q1, p2, q2)) R.pb(p2);
      if (R.size() < 2) return nope; inflag = 1; break;
    } else if (cross != 0 && segInters(p1, p2, q1, q2)) {
      if (inflag == 0) aa = ba = 0;
      R.pb(lineInters(p1, p2, q1, q2));
      inflag = (hb > 0) ? 1 : -1;
    }
    if (cross == 0 && hb < 0 && ha < 0) return R;
    bool t = cross == 0 && hb == 0 && ha == 0;
    if (t ? (inflag == 1) : (cross >= 0) ? (ha <= 0) : (hb > 0)) {
      if (inflag == -1) R.pb(q2); ba++; b++; b %= m;
    } else {
      if (inflag == 1) R.pb(p2); aa++; a++; a %= n;
    }
  }
  if (inflag == 0) {
    if (inPoly(P[0], Q)) return P; if (inPoly(Q[0], P)) return Q;
  }
  R.erase(unique(R.begin(),R.end()),R.end());
  if (R.size() > 1 && R.front() == R.back()) R.pop_back();
  return R;
}
/********** CONVEX HULL CCW *********/
pdd V; int cmpv(pdd a, pdd b) { return ccw(V,a,b) > 0;}
vector<pdd> hull(vector<pdd> &a) { //graham
  vector<pdd> b; sort(a.begin(),a.end()); V = a[0];
  sort(a.begin()+1,a.end(),cmpv);
  for(int i = 0; i < a.size(); ++i) {
    while(b.size() >= 2 && ccw(b[b.size()-2],b[b.size()-1],a[i]) <= 0) {
      b.pop_back();
    }
    b.pb(a[i]);
  }
  return b;
}
vector<pdd> hull2 (vector<pdd> & a) {
  if (a.size() == 1) return a;
  sort (a.begin(), a.end()); pdd p1 = a[0], p2 = a.back();
  vector<pdd> up, down, b; up.pb (p1); down.pb (p1);
  for (int i=1; i<a.size(); ++i) {
    if (i==a.size()-1 || ccw (p1, a[i], p2) < 0) { //<= > for colin
      while (up.size()>=2 && ccw(up[up.size()-2], up[up.size()-1], a[i])
          >= 0) {
        up.pop_back(); }
      up.pb(a[i]);
    }
    if(i==a.size()-1 || ccw (p1, a[i], p2) > 0) { //>= < for colin
      while (down.size()>=2 && ccw(down[down.size()-2],
          down[down.size()-1], a[i]) <= 0) {
        down.pop_back();
      }
      down.pb (a[i]);
    }
  }
  for (size_t i=0; i<down.size(); ++i) b.pb(down[i]);
  for (size_t i=up.size()-2; i>0; --i) b.pb(up[i]);
  return b;
}
/*************** CIRCLES *****************/
int between(pdd o, double r, pdd A, pdd B, pdd C) { //is B on arc AC (ccw)
  double a = arg(A - o), b = arg(B - o), c = arg(C - o);
  if(cmp(a,c) == -1) return cmp(a,b) <= 0 && cmp(b,c) <= 0;
```

```
  else return cmp(a,b) <= 0 || cmp(b,c) <= 0;
}
pair<pdd,pdd> circInters(pdd o1, double r1, pdd o2, double r2) {
  pdd dir = o2 - o1; double d = norm(dir);
  if(cmp(r1 + r2, d) == 0 || cmp(d + r2, r1) == 0 || cmp(d + r1, r2) ==
      0) {
    return mp(o1 + dir * r1/(r1 + r2), in); //tangent
  }
  if(cmp(r1 + r2, d) == -1 || cmp(d + r2, r1) == -1 || cmp(d + r1, r2) ==
      -1) {
    return mp(in, in); // too far, 2nd inside 1st, 1st inside 2nd
  }
  double x = (d*d - r2*r2 + r1*r1)/(2*d); //2 intersections
  return disppt(o1, o1 + dir * x/d, sqrt(r1*r1 - x*x));
}
pair<pdd,pdd> circLine(pdd o, double r, pdd a, pdd b){
  pdd h = foot(o, a, b); double d = norm(h - o);
  if(cmp(d,0) == 0) { //line through center
    return mp( (a - o)*r/norm(a-o) + o, (b - o)*r/norm(b - o) + o);
  }
  if(cmp(d*d, r*r) == 0) return mp(h, in); //tangent
  if(cmp(d*d, r*r) == 1) return mp(in, in); //no inters
  return disppt(o, h, sqrt(r*r - d*d));
}
pdd cinv(pdd o, double r, pdd p) { //inversion of p
  pdd p0 = p - o; return o + p0 * (r*r/(p0 * p0));
}
bool inCircle(pdd o, double r, pdd p) {
  return cmp((p - o)*(p - o), r*r) <= 0;
}
pdd circumcenter(pdd p, pdd q, pdd r) {
  pdd a = p - r, b = q - r, c = mp(a * (p + r) / 2, b * (q + r) / 2);
  return mp(c % pdd(a.sc, b.sc), pdd(a.fs, b.fs) % c) / (a % b);
}
pair<pdd,double> SpanningCircle(vector<pdd>& T) {
  int N = T.size(); random_shuffle(T.begin(),T.end());
  pair<pdd,double> C = mp(mp(0,0), 0);
  for (int i = 0; i < N; ++i) {
    if(inCircle(C.fs,C.sc,T[i])) continue;
    C = mp(T[i], 0);
    for (int j = 0; j < i; ++j) {
      if (inCircle(C.fs,C.sc, T[j])) continue;
      C = mp((T[i] + T[j]) / 2, norm(T[i] - T[j]) / 2);
      for (int k = 0; k < j; k++) {
        if (inCircle(C.fs,C.sc, T[k])) continue;
        pdd o = circumcenter(T[i], T[j], T[k]);
        C = mp(o, norm(o - T[k]));
      }
    }
  }
  return C;
}
pair<pdd, pdd> getTangents(pdd o, double r, pdd p) {
  if(inCircle(o,r,p)) return mp(in,in);
  double d = sqrt( (p-o)*(p-o) - r*r );
  double ang = arg(o-p); double ofs = atan(r/d);
  pdd dir1 = mp(cos(ang+ofs),sin(ang+ofs));
  pdd dir2 = mp(cos(ang-ofs),sin(ang-ofs));
  return mp(p + dir1*d, p + dir2*d);
}
/************************ ax + by + c = 0 **********************/
pair<pdd,double> getLine(pdd p1, pdd p2) { //points to ax+by+c=0
  double a = (p2.sc - p1.sc), b = (p1.fs - p2.fs);
  double n = sqrt(a*a+ b*b); a /= n; b /= n; //only if normalize
  double c = -(a*p1.fs + b*p1.sc); return mp(mp(a,b),c);
}
double dist(pdd p, pair<pdd,double> line) {
  double ret = abs(line.fs.fs*p.fs + line.fs.sc*p.sc + line.sc);
  ret /= norm(ret); return ret;
}
pdd lineInters(pair<pdd,double> l1, pair<pdd,double> l2) {
  double a1=l1.fs.fs, b1=l1.fs.sc, c1 = l1.sc;
  double a2=l2.fs.fs, b2=l2.fs.sc, c2 = l2.sc;
  double x, y;
  if(cmp(a1,0) == 0) {
    y = -c1 / b1;
    if(cmp(a2,0) != 0) x = (- c2 - b2*y) / a2;
    else return mp(INFINITY,INFINITY); //both vertical
  } else if(cmp(a2,0) == 0) {
    y = -c2 / b2; x = ( - c1 - b1*y) / a1;
  } else {
    b1 /= a1; b2 /= a2; c1 /= a1; c2 /= a2;
    if(cmp(b1,b2)==0) return mp(INFINITY,INFINITY); //parallel
    else y = (c2 - c1) / (b1 - b2); x = -c1 - b1*y;
  }
  return mp(x,y);
}
```

4.6 geom3D

```
struct point {
  double x, y, z; point(){};
  point(double _x, double _y, double _z){ x=_x; y=_y; z=_z; }
  point operator+ (point p) { return point(x+p.x, y+p.y, z+p.z); }
  point operator- (point p) { return point(x-p.x, y-p.y, z-p.z); }
  point operator* (double c) { return point(x*c, y*c, z*c); }
  point operator/ (double c) { return point(x/c, y/c, z/c); }
  point operator-() const { return point(-x, -y, -z);}
  point operator%(point p) {
    return point(y*p.z-z*p.y, z*p.x-x*p.z, x*p.y-y*p.x);
  }
  double operator* (point p) { return x*p.x + y*p.y + z*p.z; }
};
inline double dist(point a, point b) { return (a-b)*(a-b); }
void makePlane(point p1, point p2, point p3, double& a, double& b,
    double& c, double& d) {
  point normal = cross(p2-p1, p3-p1);
  a = normal.x; b = normal.y; c = normal.z; d = -a*p1.x-b*p1.y-c*p1.z;
}
point planeProj(point p, double a, double b, double c, double d) {
  double l = (a*p.x+b*p.y+c*p.z+d)/(a*a+b*b+c*c);
  return point(p.x-a*l, p.y-b*l, p.z-c*l);
}
double planeDist(point p, double a, double b, double c, double d){
  return fabs(a*p.x + b*p.y + c*p.z + d) / sqrt(a*a + b*b + c*c);
}
double planePlaneDist(double a, double b, double c, double d1, double d2){
  return fabs(d1 - d2) / sqrt(a*a + b*b + c*c); //only if parallel
}
// square distance between point and line(0), ray(2) or segment(1)
double pointLineDist(point s1, point s2, point p, int type){
  double pd2 = dist(s1, s2); point r; if(pd2 == 0) r = s1;
  else {
    double u = (p-s1)*(s2-s1) / pd2;
    r = s1 + (s2 - s1)*u;
    if(type != 0 && u < 0.0) r = s1;
    if(type == 1 && u > 1.0) r = s2;
  }
  return dist(r, p);
}
// Squared distance between lines ab and cd
double lineLineDistance(point a, point b, point c, point d) {
  point v1 = b-a, v2 = d-c, cr = cross(v1, v2);
  if (cr * cr < eps) return pointLineDist(a,b, c, 0); //parallel
  else {
    point n = cr / sqrt(cr * cr); //direction
    return n * (c - a);
  }
}
point lineLineInters(point a, point b, point c, point d) {
  if(lineLineDistance(a,b,c,d) > eps) return point(); //don't intersect
  point v1 = b-a, v2 = d-c, v3 = c-a, cr = cross(v1, v2);
  double s = (cross(v3,v2) * cr) / (cr*cr);
  return a + (b-a)*s;
}
double signedTetrahedronVol(point A, point B, point C, point D) {
  double A11 = A.x - B.x, A12 = A.x - C.x, A13 = A.x - D.x;
  double A21 = A.y - B.y, A22 = A.y - C.y, A23 = A.y - D.y;
  double A31 = A.z - B.z, A32 = A.z - C.z, A33 = A.z - D.z;
  double det = A11*A22*A33 + A12*A23*A31 +
    A13*A21*A32 - A11*A23*A32 - A12*A21*A33 - A13*A22*A31;
  return det / 6;
}

/********* HULL ******************/
struct twoset {
  void insert(int x) { (a == -1 ? a : b) = x; }
  bool contains(int x) { return a == x || b == x; }
  void erase(int x) { (a == x ? a : b) = -1; }
  int size() { return (a != -1) + (b != -1); } int a, b;
} E[MAXN][MAXN];
struct face { point norm; double disc; int I[3];};
face make_face(int i, int j, int k, int inside_i) {
  E[i][j].insert(k); E[i][k].insert(j); E[j][k].insert(i);
  face f; f.I[0] = i; f.I[1] = j; f.I[2] = k;
  f.norm = (A[j] - A[i]) % (A[k] - A[i]); f.disc = f.norm * A[i];
  if(f.norm * A[inside_i] > f.disc) {
    f.norm = -f.norm; f.disc = -f.disc;
  }
  return f;
}
vector<face> hull3(vector<point> &A) {
  vector<face> faces; face f; memset(E, -1, sizeof(E));
  FOR(i,4) for(int j=i+1;j<4;j++) for(int k=j+1;k<4;k++) {
    faces.pb(make_face(i, j, k, 6 - i - j - k));
  }
  for(int i = 4; i < N; i++) {
    for(int j = 0; j < faces.size(); j++) {
```

```
      f = faces[j];
      if(f.norm * A[i] > f.disc) {
        E[f.I[0]][f.I[1]].erase(f.I[2]);
        E[f.I[0]][f.I[2]].erase(f.I[1]);
        E[f.I[1]][f.I[2]].erase(f.I[0]);
        faces[j--] = faces.back();
        faces.resize(faces.size() - 1);
      }
    }
    int nfaces = faces.size();
    for(int j = 0; j < nfaces; j++) {
      f = faces[j];
      for(int a = 0; a < 3; a++) for(int b = a + 1; b < 3; b++) {
        int c = 3 - a - b;
        if(E[f.I[a]][f.I[b]].size() == 2) continue;
        faces.pub(make_face(f.I[a], f.I[b], i, f.I[c]));
      }
    }
  }
  return faces;
}
```

4.7 geomMisc

```
/****************** CLOSEST POINTS ********************/
vector<pair<pll,int> > v,x,y;
ll INF = 4e18;
ll dist(pll a, pll b) {
  return (a.fs - b.fs) * (a.fs - b.fs) + (a.sc - b.sc) * (a.sc - b.sc);
}
pair<ll,pii> solve(int st, int dr) {
  if (st >= dr - 1) return mp(INF,mp(0,0));
  if (dr - st == 2) {
    if (y[st] > y[st + 1]) swap(y[st], y[st + 1]);
    return mp(dist(x[st].fs, x[st + 1].fs),mp(x[st].sc, x[st+1].sc));
  }
  int mij = (st + dr) / 2;
  pair<ll,pii > ret = min(solve(st,mij),solve(mij,dr));
  merge(y.begin() + st, y.begin() + mij, y.begin() + mij, y.begin() + dr,
      v.begin());
  copy(v.begin(), v.begin() + (dr - st), y.begin() + st);
  int nr = 0;
  for (int i=st; i<dr; ++i) {
    if (abs(y[i].fs.sc - x[mij].fs.fs) < ret.fs) v[nr++] = y[i];
  }
  for(int i=0;i<nr;++i) {
    for (int j=i+1; j<nr && j<=i+7; ++j) {
      ll d = dist(v[i].fs,v[j].fs);
      if(d < ret.fs) ret = mp(d,mp(v[i].sc,v[j].sc));
    }
  }
  return ret;
}
pair<ll,pii> closest_points() {
  int N = x.size(); sort(x.begin(), x.end()); v.resize(N);
  for(ll i=0;i<N;++i) y.pb( mp(mp(x[i].fs.sc, x[i].fs.fs), x[i].sc));
  return solve(0,N);
}
/****************** MAX AREA ********************/
pair<pdd,pair<pdd,pdd> > getMaxArr(vector<pdd> &v) {
  v = convex_hull(v);
  int A = 0, B = 1, C = 2, N = v.size();
  double S = 0; pdd a,b,c;
  while(true) {
    double T = arr(v[A],v[B],v[C]);
    if(T > S) {a = v[A], b = v[B], c = v[C]; S = T;}
    while(true) {
      while(T <= arr(v[A], v[B], v[(C+1) %N])) {
        C = (C+1) %N; T = arr(v[A],v[B],v[C]);
        if(T > S) {a = v[A], b = v[B], c = v[C]; S = T;}
      }
      if(T < arr(v[A], v[(B+1) %N], v[C])) {
        B = (B+1) %N; T = arr(v[A],v[B],v[C]);
        if(T > S) {a = v[A], b = v[B], c = v[C]; S = T;}
      } else break;
    }
    A = (A+1) % N; if(A == 0) break;
    if(A == B) B = (B+1) %N; if(B == C) C = (C+1) %N;
  }
  return mp(a,mp(b,c));
}
/****************** FARTHEST POINTS ********************/
pair<pdd,pdd> farthestPoints(vector<pdd> &v) {
  v = convex_hull(v); vector<pair<pdd,pdd> > A; //antipodal
  int k = 1, M = v.size();
  while(area(v[0],v[k+1],v[M-1]) > area(v[0],v[k],v[M-1])) ++k;
  int i = 0, j = k;
  //If want farthest point from each edge, do j<2*M and double array
```

```
  while(i<=k && j<M) {
    A.pb(mp(v[i],v[j]));
    while(j<M-1 && area(v[i+1],v[j+1],v[i]) > area(v[i+1],v[j],v[i])) {
      A.pb(mp(v[i],v[j])); ++j;
    }
    ++i;
  }
  pair<pdd,pdd> ret = mp(v[0],v[0]);
  for(auto p: A) {
    if(dist(p.fs,p.sc) > dist(ret.fs,ret.sc)) ret = p;
  }
  return ret;
}
/*************** TRANSFORMATIONS AND ELLIPSE ***************/
double rot[3][3] = {{cos(a),-sin(a),0},{sin(a),cos(a)},0},{0,0,1}};
double rotx[4][4] = {{1,0,0,0}, {0,cos(a),-sin(a),0},
{0,sin(a),cos(a)},0}, {0,0,0,1}};
double roty[4][4] = {{cos(a),0,sin(a),0}, {0,1,0,0},
{-sin(a),0,cos(a)},0},{0,0,0,1}};
double rotz[4][4] = {{cos(a),-sin(a),0,0},{sin(a),cos(a),0,0},
{0,0,1,0},{0,0,0,1}};
long double ellipseInt(long double x) {
  return b*(x*sqrt(a*a - x*x) + a*a*atan(x / sqrt(a*a - x*x))) / (2*a);
}
```

4.8 geomSphere

```
#define PI 3.14159265358979
double R = 6370.0; //earth
double rad(double x) { return x * PI / 180.0;}
double ang(double x) { return x * 180.0 / PI;}
double dist(pdd a, pdd b) {
  if(a == b) return 0;
  double v = sin(a.fs)*sin(b.fs) + cos(a.fs) * cos(b.fs) * cos(a.sc -
      b.sc);
  return acos(v);
}
struct cel {
  double x,y,z;
  cel(double lat, double lon) {
    x = cos(lat) * cos(lon); y = cos(lat) * sin(lon); z = sin(lat);
  }
  cel(double a1, double a2, double a3) { x = a1, y = a2, z = a3; }
  inline cel operator^(cel c1) { //cross
    return cel(y*c1.z - c1.y * z, c1.x * z - c1.z * x, c1.y * x - c1.x *
        y);
  }
  inline cel operator*(double cx) { return cel(cx*x,cx*y,cx*z); }
  double mag() { return sqrt(x*x + y*y + z*z); }
  pdd rev() {
    double lat = asin(z); double tmp = cos(lat);
    double sgn = asin(y/tmp); double lon = acos(x / tmp);
    if(sgn < 0) lon = 2 * PI - lon; return mp(lat,lon);
  }
};
int inside(pdd a, pdd b, pdd c) {
  return abs(dist(a,b) + dist(a,c) - dist(b,c)) <= 0.000001;
}
pdd inter(pdd a, pdd b, pdd c, pdd d) {
  cel p1(a.fs,a.sc); cel p2(b.fs,b.sc);
  cel p3(c.fs,c.sc); cel p4(d.fs,d.sc);
  cel v1 = p1 ^ p2; cel v2 = p3 ^ p4;
  v1 = v1 * (1.0 / v1.mag()); v2 = v2 * (1.0 / v2.mag());
  cel d1 = v1 ^ v2; d1 = d1 * (1.0 / d1.mag());
  cel d2 = d1 * (-1);
  pdd x1 = d1.rev(), x2 = d2.rev();
  if(inside(x1,a,b) && inside(x1,c,d)) return x1;
  if(inside(x2,a,b) && inside(x2,c,d)) return x2;
  return mp(inf,inf);
}
```

4.9 numberTheory

```
int lcm(int a, int b) { return a / __gcd(a,b) * b; }
int gcd(int a, int b, int &x, int &y) {
  if(!b) { x=1; y=0; return a; }
  else {
    int x0, y0, d = gcd(b,a%b,x0,y0);
    x = y0; y = x0 - a/b *y0; return d;
  }
}
pair<int,int> euclid(int a, int b, int c) { //ax - by = c;
  int x, y, sol1, sol2; int d = gcd(a,b,x,y);
  if(c%d) return mp(0,0); //no sol
  } else { sol1 = (c/d)*x; sol2 = -(c/d)*y; }
  //only if want minimal
```

```
  while(sol1 < 0 || sol2 < 0) { sol1 += b/d; sol2 += a/d;}
  while(sol1 >= b/d || sol2 >= a/d) { sol1 -= b/d; sol2 -= a/d;}
  return mp(sol1,sol2);
}
int inversmod(int a, int b) { //inverse of a mod b
  int x,y; gcd(a,b,x,y);
  if(x<0) { int k = (-x-1)/b + 1; x += k*b;}
  return x%b;
}
ll mulmod(ll a,ll b,ll c) {// a*b mod c
  ll x = 0, y=a%c;
  while(b > 0){ if(b%2 == 1) x = (x+y)%c; y = (y*2)%c; b /= 2; }
  return x%c;
}
int fi[Nmax],sp[Nmax]; //sp = smallest prime
void makeSieve(int lim) {
  for(int i=2;i<=lim;++i) {
    if(!sp[i]) {
      for(int j=i;i*j<=lim;++j) { if(!sp[i*j]) sp[i*j] = i;}
      sp[i] = i;
    }
  }
}
void makeFi(int lim) {
  fi[1] = 1; for(int i=2;i<=lim;++i) fi[i] = i-1;
  for(int i=2;i<=lim;++i) {
    for(int j=2;i*j<=lim;++j) fi[i*j] -= fi[i];
  }
}
vector<int> linSolver(int a, int b, int c) { //ax = b (mod c)
  vector<int> sol; int d = __gcd(a,c);
  pair<int, int> e = euclid(a,c,d);
  int x = e.fs, y = e.sc;
  if(b%d == 0) {
    x = ((b/d * x) % c + c) % c;
    for(int i=0;i<d;++i) {
      sol.pb(((x + c/d * i)%c + c)%c );
    }
  }
  return sol;
}
// x = a1 mod m1, x = a2 mod m2
pair<int,int> crt(int a1, int m1, int a2, int m2) {
  int d = gcd(m1,m2);
  pair<int,int> p = euclid(m1,m2,d);
  int s = p.fs, t = -p.sc;
  if (a1 % d != a2 % d) return make_pair(0, -1);
  int x = (s * a2 * m1 + t * a1 * m2) % (m1*m2);
  x = (x + m1*m2) % (m1*m2);
  return mp(x/d, m1*m2/d);
}
// x = a[i] (mod m[i]) also returns period (lcm)
pair<int,int> chinese(vector<int> &a, vector<int> &m) {
  pair<int,int> ret = mp(a[0], m[0]);
  for (int i = 1; i < a.size(); ++i) {
    ret = crt(ret.fs, ret.sc, a[i], m[i]); if (ret.sc == -1) break;
  }
  return ret;
}
```

4.10 polynom

```
typedef complex<double> cdouble;
typedef vector<cdouble> pol;
int cmp(double a, double b) { //same as geom2d
  if(a + eps <= b) return -1; if(a <= b + eps) return 0; return 1;
}
int cmp(cdouble x, cdouble y = 0) { return cmp(abs(x), abs(y));}
pol deriv(pol &a) {
  int N = a.size(); pol ret(N-1);
  for(int i=1;i<N;++i) ret[i-1] = a[i]*cdouble(i); return ret;
}
pair<pol, cdouble> ruffini(pol &a, cdouble z) {// divide by (x-z)
  int N = a.size(); pol ret;
  if (N == 0) return mp(ret, 0); if (N == 1) return mp(ret, a[0]);
  ret.resize(N-1); ret[N-2] = a[N-1];
  for(int i=N-2;i>0;--i) ret[i-1] = ret[i]*z + a[i];
  return mp(ret, ret[0]*z + a[0]);
}
cdouble getVal(pol &a, cdouble z) { return ruffini(a,z).sc; }
cdouble find_one_root(pol &a, cdouble x) {
  pol a1 = deriv(a); pol a2 = deriv(a1);
  int N = a.size(), m = 1000;
  while (m--) {
    cdouble y0 = getVal(a,x);
    if(cmp(y0) == 0) break;
    cdouble G = getVal(a1,x) / y0;
    cdouble H = G * G - getVal(a2,x) / y0;
```

```
    cdouble R = sqrt(cdouble(N-1) * (H * cdouble(N) - G * G));
    cdouble D1 = G + R, D2 = G - R;
    cdouble A = cdouble(N) / (cmp(D1, D2) > 0 ? D1 : D2);
    x -= A; if (cmp(A) == 0) break;
  }
  return x;
}
```

5 Misc

5.1 binSearch

```
int binary_search(int val) { //array A of size N
  int i, step;
  for (step = 1; step < N; step <<= 1);
  for (i = 0; step; step >>= 1) {
    if (i + step < N && A[i + step] <= val) i += step;
  }
  return i;
}
```

5.2 dpOptim

```
/* 1. best[i] = max(best[i] + a[i]*b[j]) a increasing, b decreasing
max:q(a[i]),add(b[i],best[i]); min: q(-a[i]),add(b[i],-best[i])
2. best[i][j] = max(best[i-1][k] + b[k]*a[j]), a increasing, b decreasing
Do independently for each i
max:q(a[j]),add(b[j],best[i-1][j]);min:q(-a[j]),add(b[j],-best[i-1][j]) */
int cnt; pll h[Nmax]; //stores the lines
pll operator-(pll a, pll b) { return mp(a.fs-b.fs, a.sc-b.sc); }
inline int cw(pll a, pll b, pll c) {
  pll x = a-c, y = b-c; return x.fs*y.sc <= x.sc*y.fs; //Beware overflow
}
void add(ll a, ll b) { // Add ax+b (decreasing order of a)
  h[cnt++] = mp(a,b);
  while(cnt > 2 && cw(h[cnt-3],h[cnt-2],h[cnt-1])) {
    h[cnt-2] = h[cnt-1]; --cnt;
  }
}
long long query(long long x) { //return max ax+b
  int st = 0, dr = cnt - 2, ret = cnt-1;
  while(st <= dr) {
    int mij = (st+dr)/2;
    if(h[mij].fs * x + h[mij].sc > h[mij+1].fs*x + h[mij+1].sc) {
       ret = mij; dr = mij-1;
    } else st = mij+1;
  }
  return h[ret].fs * x + h[ret].sc;
}
/*KNUTH: d[i][j] = min(d[i][k] + d[k][j] + c[i][j])
DIVCONQ: d[i][j] = min(d[i-1][k] + c[k][j]), c[i][j] <= c[i][j+1]
compute(j, l, r, optL, optR) m = (l+r)/2; brut(m,optL,optR);
compute(j,l,m-1,optL, opt[m][j]); compute(j,m+1,r, opt[m][j], optR);
*/
```

5.3 gauss

```
double A[Nmax][Nmax], X[Nmax];
int k, N, M; // A[i][1] * X[1] + ... + A[i][M] * X[M] = A[i][M+1]
void gauss() {
  int i=1,j=1;
  while(i<=N && j<=M) {
    int k;
    for(k=i;k<=N;++k) if( A[k][j]<-EPS || A[k][j]>EPS) break;
    if(k==N+1) { ++j; continue; }
    if(k!=i) for(int q=1;q<=M+1;++q) swap(A[i][q],A[k][q]);
    for(int q=j+1;q<=M+1;++q) A[i][q] /= A[i][j];
    A[i][j] = 1;
    for(int u=i+1;u<=N;++u) {
      for(int q=j+1;q<=M+1;++q) A[u][q] -= A[u][j]*A[i][q];
      A[u][j]=0;
    }
    ++i; ++j;
  }
  for(int i=N;i>0;--i) {
    for(int j=1;j<=M+1;++j) {
      if(A[i][j]>EPS || A[i][j]<-EPS) {
        if(j==M+1) {} //HANDLE NO SOLUTION
        X[j] = A[i][M+1];
        for(int k=j+1;k<=M;++k) X[j] -= X[k]*A[i][k];
        break;
      }
    }
```

```
  }
}
```

5.4 matpow

```
vvll mul(vvll &a, vvll &b) {
  int N = sz(a), M = sz(a[0]), P = sz(b[0]);
  vvll ret(N, vll (P,0));
  FOR(i,N) FOR(j,P) FOR(k,M)
    ret[i][j] += a[i][k]*b[k][j]; ret[i][j] %= MOD;
  return ret;
}
vvll matpw(vvll &v, long long k) {
  int N = sz(v); vvll ret(N, vll(N,0));
  for(int i=0;i<N;++i) ret[i][i] = 1;
  for(int i=0;(1LL<<i) <= k; ++i) {
    if((1LL<<i)&k) ret = mul(ret, v); v = mul(v,v);
  }
  return ret;
}
```

5.5 parse

```
#define BUFFER_SIZE 1234
char buff[BUFFER_SIZE];
int buffIt;
inline int getNumber() {
  int ret = 0;
  while (buff[buffIt] < '0' || buff[buffIt] > '9')
    if (++buffIt == BUFFER_SIZE)
      fread(buff, BUFFER_SIZE, 1, stdin), buffIt = 0;
  while (buff[buffIt] >= '0' && buff[buffIt] <= '9') {
    ret = ret * 10 + buff[buffIt] - '0';
    if (++buffIt == BUFFER_SIZE) {
      buffIt = 0; fread(buff, BUFFER_SIZE, 1, stdin);
    }
  }
  return ret;
}
for(int i=1;i<=M;++i) {
  if(!getline(cin, s)) break;
  if (s == "") { --i; continue; }
  stringstream ss; ss << s;
  int x; while(ss >> x) { g[i].push_back(x+M+1); }
}
```

5.6 simplex

```
// maximize cx, Ax <= b and x >= 0, A = Mxy, b size M, c size y
typedef long double DOUBLE;
typedef vector<DOUBLE> VD;
typedef vector<VD> VVD;
typedef vector<int> VI;
DOUBLE inf = -numeric_limits<DOUBLE>::infinity();
const DOUBLE EPS = 1e-9;

struct LPSolver {
  int m, n;
  VI B, y; //slack variables, actual variables
  VVD D; //canonical form
  LPSolver(const VVD &A, const VD &b, const VD &c) {
    m = b.size(); n = c.size(); y = VI(n+1);
    B = VI(m); D = VVD(m+2,VD(n+2));
    for (int i = 0; i < m; i++) {
      for (int j = 0; j < n; j++) D[i][j] = A[i][j];
    }
    for (int i = 0; i < m; i++) {
      B[i] = n + i; D[i][n] = -1; D[i][n + 1] = b[i];
    }
    for (int j = 0; j < n; j++) {
      y[j] = j; D[m][j] = -c[j];
    }
    y[n] = -1; D[m + 1][n] = 1;
  }
  void Pivot(int r, int s) {
    double inv = 1.0 / D[r][s];
    for (int i = 0; i < m + 2; i++) {
      if (i == r) continue;
      for (int j = 0; j < n + 2; j++) {
        if (j == s) continue;
        D[i][j] -= D[r][j] * D[i][s] * inv;
      }
    }
    for (int j = 0; j < n + 2; j++) {
```

```
    if (j != s) D[r][j] *= inv;
  }
  for (int i = 0; i < m + 2; i++) {
    if (i != r) D[i][s] *= -inv;
  }
  D[r][s] = inv; swap(B[r], y[s]);
}
bool Simplex(int phase) {
  int x = phase == 1 ? m + 1 : m;
  while (true) {
    int s = -1;
    for (int j = 0; j <= n; j++) {
      if (phase == 2 && y[j] == -1) continue;
      if (s == -1 || D[x][j] < D[x][s] || D[x][j] == D[x][s] && y[j] <
          y[s]) s = j;
    }
    if (D[x][s] > -EPS) return true;
    int r = -1;
    for (int i = 0; i < m; i++) {
      if (D[i][s] < EPS) continue;
      if (r == -1 || D[i][n + 1] / D[i][s] < D[r][n + 1] / D[r][s] ||
        (D[i][n + 1] / D[i][s]) == (D[r][n + 1] / D[r][s]) && B[i] <
            B[r]) r = i;
    }
    if (r == -1) return false;
    Pivot(r, s);
  }
}
DOUBLE Solve(VD &x) {
  int r = 0;
  for (int i = 1; i < m; i++) {
    if (D[i][n + 1] < D[r][n + 1]) r = i;
  }
  if (D[r][n + 1] < -EPS) {
    Pivot(r, n);
    if (!Simplex(1) || D[m + 1][n + 1] < -EPS) return inf;
    for (int i = 0; i < m; i++) if (B[i] == -1) {
      int s = -1;
      for (int j = 0; j <= n; j++) {
        if (s == -1 || D[i][j] < D[i][s] || D[i][j] == D[i][s] && y[j] <
            y[s]) {
          s = j;
        }
      }
      Pivot(i, s);
    }
    if (!Simplex(2)) return inf;
    x = VD(n);
    for (int i = 0; i < m; i++) if (B[i] < n) x[B[i]] = D[i][n + 1];
    return D[m][n + 1];
  }
};
```

5.7 ternary

```
// F is some concave function
double ternarySearch(double left, double right) {//finds max
  while(abs(right - left) > EPS) {
    long double lt = left + (right - left) / 3.0;
    long double rt = right - (right - left) / 3.0;
    if (F(lt) < F(rt)) left = lt;
    else right = rt; //reverse if convex and min
  }
  return (left + right) / 2.0;
}
```

5.8 util

```
struct cmp {
  bool operator()(piii a, piii b) { return a.fs > b.fs;}
};
priority_queue<piii, vector<piii>, cmp> pq;
namespace std {
  template <>
  struct hash<pii> {
  public:
    size_t operator()(pair<int, int> x) const throw() {
      size_t h = x.fs + x.sc * 1145; return h;
    }
  };
}
for (int news=state; news; news=state&(news-1))//all subsets of state
```

6 String

6.1 ahoCorasick

```
int tr[Nmax], nw[Nmax], NR, term[Nmax], len[Nmax], to[Nmax][sigma],
    link[Nmax], sz = 1;
void add(string s) { // doesn't handle dups
  int cur = 0;
  for(auto c: s) {
    if(!to[cur][c - 'a']) {
       to[cur][c - 'a'] = sz++;
       len[to[cur][c - 'a']] = len[cur] + 1;
    }
    cur = to[cur][c - 'a'];
  }
  term[cur] = cur; tr[cur] = ++NR;
}

void push_links() {
  queue<int> Q; Q.push(0);
  while(!Q.empty()) {
    int V = Q.front(); Q.pop(); int U = link[V];
    if(!term[V]) term[V] = term[U];
    for(int c = 0; c < sigma; c++) {
      if(to[V][c]) {
         link[to[V][c]] = V ? to[U][c] : 0;
         Q.push(to[V][c]);
      } else to[V][c] = to[U][c];
    }
  }
}

void match(string s) {
  int cur = 0;
  for(auto c : s) {
    cur = to[cur][c-'a']; int f = cur;
    while(f) {
      if(tr[f]) tr[f]; // match on tr[f]
      if(f == term[f]) f = term[link[f]];
      else f = term[f];
    }
  }
}
```

6.2 expresioneval

```
string gb; vector<vector<string>> ops;
// is operation op, at location loc, when the string is st,dr
int is(string op, int loc,int st,int dr) {
  if(loc + op.size() - 1 > dr) return 0;
  int ok = 1;
  for(int i=0;i<op.size() && ok;++i) {
    if(gb[i+loc] != op[i]) ok = 0;
  }
  return ok;
}
int make(string op,int x,int y) { // make the operations
  if(op == "+") return x+y; return 0;
}
// evals substring(st,dr), when we should only check lvl forward
int eval(int st,int dr, int lvl) {
  for(int t = lvl;t<ops.size();++t) {
    int par = 0;
    for(int i = dr;i>=st;--i) {
      if(gb[i] == '(') ++ par; if(gb[i] == ')') -- par;
      if(par) continue;
      for(auto op : ops[t]) {
        if(is(op,i,st,dr)) {
          return make(op,eval(st,i-1,t),eval(i+op.size(),dr,t));
        }
      }
    }
  }
  if(gb[st] == '(') return eval(st+1,dr-1,0);
  return getnum(st,dr);
}
void init() { //add operations in order, call at main start
  ops.pb(vector<string>({"+","-"}));
  ops.pb(vector<string>({"/","%","*"}));
}
int eval(string &s) {
  gb = "";
  for(auto c : s) if(c != ' ') gb.pb(c);
  int ret = eval(0,gb.size()-1,0);
  return ret;
}
```

6.3 kmp

```
void make() {
  int k = -1; nx[0] = -1;
  for(int i=1;i<sz(s1);++i) {
    while(k >= 0 && s1[k+1] != s1[i]) k = nx[k];
    if(s1[k+1] == s1[i]) ++k; nx[i] = k;
  }
}
void match() {
  int k=-1;
  for(int i=0;i<sz(s2);++i) {
    while(k >= 0 && s1[k+1] != s2[i]) k = nx[k];
    if(s1[k+1] == s2[i]) ++k;
    if(k==sz(s1)-1) k = nx[k]; // match on i - sz(s1)
  }
}
```

6.4 pscpld

```
char s[2010000], s1[2010000];
int val[2020201],maxind,maxVal,N;
void make_sir(){
  s1[0]='*';
  for(int i=1;i<=N;++i) { s1[i*2-1]=s[i]; s1[i*2]='*';}
}
ll pscpld (char *s) {
  ll S = 0; N = strlen(s+1); make_sir();
  for(int i=1;i<2*N;++i) {
    if(maxVal >= i) {
      int loc = maxind - (i-maxind);
      val[i] = min(val[loc],maxVal-i);
    }
    while((i - val[i] >= 0) && (i + val[i] <= 2*N) &&
          (s1[i-val[i]] == s1[i+val[i]])) {
      ++val[i];
      if(i + val[i] > maxVal){
        maxVal = i + val[i]; maxind = i;
      }
    }
  }
  for(int i=1;i<2*N;++i){
    S += (val[i]+1)/2; if(s1[i]=='*') --S;
  }
  return S;
}
```

6.5 rabinquery

```
#define Nmax 101010
#define p1 47
#define p2 149
#define MOD1 666013
#define MOD2 991777
int nr1[Nmax],pow1[Nmax],pow2[Nmax],nr2[Nmax],nrfin;
char car[Nmax]; // string we want to hash, STARTING FROM 1
int N,M;
void make() {
  pow1[0]=1,pow2[0]=1;
  for(int i=1;i<=N;++i) {
    pow1[i]=(1LL*pow1[i-1]*p1)%MOD1;
    pow2[i]=(1LL*pow2[i-1]*p2)%MOD2;
    nr1[i]=((1LL*nr1[i-1]*p1)%MOD1 + car[i])%MOD1;
    nr2[i]=((1LL*nr2[i-1]*p2)%MOD2 + car[i])%MOD2;
  }
}
int querry(int x,int y,int x1,int y1,int debug) {
  int sol1,sol2,sol12,sol22;
  sol1=1LL*(nr1[y]-(1LL*pow1[y-x+1]*nr1[x-1])%MOD1+MOD1)%MOD1;
  sol2=1LL*(nr1[y1]-(1LL*pow1[y1-x1+1]*nr1[x1-1])%MOD1+MOD1)%MOD1;
  sol12=(nr2[y]-(1LL*pow2[y-x+1]*nr2[x-1])%MOD2+MOD2)%MOD2;
  sol22=(nr2[y1]-(1LL*pow2[y1-x1+1]*nr2[x1-1])%MOD2+MOD2)%MOD2;
  if(sol1==sol2 && sol12 == sol22) return 1; return 0;
}
```

6.6 suffixAuto

```
unordered_map<char, int> h[200100];
int len[200100],lnk[200100],last,curr,nr;
long long add_char(char c) {
  long long ret = 0;
  curr = ++nr; len[curr] = len[last] + 1; int p = last;
  while(p !=-1 && !h[p][c]){h[p][c] = curr; p = lnk[p];}
  if(p==-1) {
```

```
    lnk[curr] = 0; ret += len[curr];
  } else {
    int q = h[p][c];
    if(len[q] == len[p]+1) {
      lnk[curr] = q; ret += (len[curr] - len[q]);
    } else {
      int clone = ++nr;
      len[clone] = len[p] + 1; lnk[clone] = lnk[q];
      ret += (len[clone] - len[lnk[q]]); h[clone] = h[q];
      while(p!=-1 && h[p][c] == q) {
        h[p][c] = clone; p = lnk[p];
      }
      ret -= (len[q] - len[lnk[q]]); lnk[q] = clone;
      ret += (len[q] - len[lnk[q]]); lnk[curr] = clone;
      ret += (len[curr] - len[clone]);
    }
  }
  last = curr; return ret;
}
void suffix_automaton(string &s) {
  last = 0, curr = 0, nr = 0; len[0] = 0; lnk[0] = -1;
  for(auto c: s) add_char(c); //also counts new suffixes
}
```

6.7 suffixarray

```
struct entry {
  int nr[2], p;
} L[Nmax];
int P[LMax][Nmax], N, i, stp, cnt;
bool cmp(const entry &a, const entry &b) {
  return a.nr[0] == b.nr[0] ? (a.nr[1] < b.nr[1]) : (a.nr[0] < b.nr[0]);
}
int lcp(int x, int y) {
  int k, ret = 0; if (x == y) return N - x;
  for (k = stp - 1; k >= 0 && x < N && y < N; --k) {
    if (P[k][x] == P[k][y]) {
      x += 1 << k; y += 1 << k; ret += 1 << k;
    }
  }
  return ret;
}
void suffArr(string A) {
  int N = a.size();
  for(int i=0;i<N;++i) P[0][i] = A[i] - 'a';
  for (stp = 1, cnt = 1; cnt >> 1 < N; ++stp, cnt <<= 1) {
    for (i = 0; i < N; ++i) {
      L[i].p = i; L[i].nr[0] = P[stp - 1][i];
      L[i].nr[1] = i + cnt < N ? P[stp - 1][i + cnt] : -1;
    }
    sort(L, L + N, cmp);
    for (i = 0; i < N; ++i)
      P[stp][L[i].p] = i > 0 && L[i].nr[0] == L[i - 1].nr[0] &&
          L[i].nr[1] == L[i - 1].nr[1] ? P[stp][L[i - 1].p] : i;
  }
}
```

6.8 zalgo

```
// z -> int array, s2 -> char array, N, it's length. At the end have z[i]
int left=0, right=0;
for(int i=1;i<N;++i) {
  if( i > right) {
    left = i, right = i;
    while (right < N && s2[right-left] == s2[right]) ++right;
    z[i] = right - left; --right;
  } else {
    int k = i - left; if(z[k] < right-i+1) z[i] = z[k];
    else {
      left=i;
      while (right < N && s2[right - left] == s2[right]) ++right;
      z[i] = right-left; --right;
}}}
```