Template

GummyBear

October 10, 2019

目录

1 A

1.1 .vimrc

```
set nu ai ci si mouse=a ts=2 sts=2 sw=2
nmap<F2> : vs %<.in <CR>
nmap<F3> : !gedit % <CR>
nmap<F8> : !time ./%< < %<.in <CR>
nmap<F9> : :w <CR> :!g++ % -o %< -O2 -g -std=c++11 -Wall <CR>
nmap<F10> : :w <CR> :make %< <CR>
```

1.2 head

```
#include<bits/stdc++.h>
using namespace std;
#define fi first
#define se second
#define mp make_pair
#define pb push_back
#define rep(i, a, b) for(int i=(a); i<(b); i++)
#define per(i, a, b) for(int i=(b)-1; i>=(a); i--)
#define sz(a) (int)a.size()
#define de(a) cout << #a << " = " << a << endl
#define dd(a) cout << #a << " = " << a << " "
#define all(a) a.begin(), a.end()
#define pw(x) (1ll<<(x))
#define endl "\n"
typedef long long ll;
typedef pair<int, int> pii;
typedef vector<int> vi;
typedef double db;

int main() {
   std::ios::sync_with_stdio(false);
   std::cin.tie(0);
   // cout << setiosflags(ios::fixed);
   // cout << setprecision(3);
   return 0;
}
```

2 DP

2.1 DigDP

```
ll f[];
ll dfs(int pos, ..., bool lim) {
   if (pos == -1) return ?;
   if (!lim && ~f[...]) return f[...];
   ll res = 0;
   int up = lim ? dig[pos] : 9;
   rep(i, 0, up + 1) {
      if (..) res += dfs(pos - 1, ..., lim & (i == up));
```

```
   }
   if (!lim) f[] = res;
   return res;
}
ll solve(ll x) {
   int pos = 0;
   while(x) dig[pos++] = x % 10, x /= 10;
   return dfs(pos - 1, ..., 1);
}
void init() { memset(f, -1, sizeof(f)); }
void solve() {
   init();
   // 可调用 solve(x) 多次
}
```

3 DataStructure

3.1 2DST

```
namespace ST_2D{
   const int N = 1010, M = 11;
   int Log[N], p[M], dep1, dep2;
   short st[M][M][N][N];
   void build(int n, int m, short a[][N]){
      rep(i, 0, M) p[i] = 1 << i;
      rep(i, 2, N) Log[i] = Log[i >> 1] + 1;
      for(dep1 = 0; (1 << dep1) < n; dep1++);
      for(dep2 = 0; (1 << dep2) < m; dep2++);
      rep(i, 1, n + 1) rep(j, 1, m + 1)
         st[0][0][i][j] = a[i][j]; // modify
      rep(i, 1, n + 1) rep(j, 1, dep2 + 1) rep(k, p[j], m + 1)
         st[0][j][i][k] = max(st[0][j-1][i][k], st[0][j-1][i][k - p[j-1]]);
      //attention to range of k
      rep(i, 1, dep1+1) rep(j, p[i], n+1) rep(k, 0, dep2+1) rep(l, p[k], m+1)
         st[i][k][j][l]=max(st[i-1][k][j-p[i-1]][l], st[i-1][k][j][l]);
   }
   int qry(int x1, int y1, int x2, int y2){
      int l1 = Log[x2-x1+1], l2 = Log[y2-y1+1];
      int res1 = max(st[l1][l2][x1+p[l1]-1][y1+p[l2]-1], st[l1][l2][x2][y2]);
      int res2 = max(st[l1][l2][x1+p[l1]-1][y2], st[l1][l2][x2][y1+p[l2]-1]);
      return max(res1, res2);
   }
}
```

3.2 2DSegTree

```
// 区域覆盖、标记永久化、标记单调
const int N=1010;
int n,m,q;
struct seg {
    int ma[N<<2], la[N<<2];
    void upd(int L,int R,int c,int l=0,int r=m,int rt=1) {
        ma[rt]=max(ma[rt], c);
        if(L<=l&&r<=R) return la[rt]=max(la[rt], c), void();
        int mid=l+r>>1;
        if(L<=mid) upd(L, R, c, l, mid, rt<<1);
        if(R>=mid+1) upd(L, R, c, mid+1, r, rt<<1|1);
    }
    int qry(int L,int R,int l=0,int r=m,int rt=1) {
        int ans=0;
        ans=max(ans, la[rt]);
        if(L<=l&&r<=R) return ans=max(ans, ma[rt]);
        int mid=l+r>>1;
        if(L<=mid) ans=max(ans, qry(L, R, l, mid, rt<<1));
        if(R>=mid+1) ans=max(ans, qry(L, R, mid+1, r, rt<<1|1));
        return ans;
    }
};
struct Seg {
    seg ma[N<<2], la[N<<2];
    void upd(int x1,int x2,int y1,int y2,int c,int l=0,int r=n,int rt=1) {
        ma[rt].upd(y1, y2, c);
        if(x1<=l&&r<=x2) return la[rt].upd(y1, y2, c), void();
        int mid=l+r>>1;
        if(x1<=mid) upd(x1, x2, y1, y2, c, l, mid, rt<<1);
        if(x2>=mid+1) upd(x1, x2, y1, y2, c, mid+1, r, rt<<1|1);
    }
    int qry(int x1,int x2,int y1,int y2,int l=0,int r=n,int rt=1) {
        int ans=0;
        ans=max(ans, la[rt].qry(y1, y2));
        if(x1<=l&&r<=x2) return ans=max(ans, ma[rt].qry(y1, y2));
        int mid=l+r>>1;
        if(x1<=mid) ans=max(ans, qry(x1, x2, y1, y2, l, mid, rt<<1));
        if(x2>=mid+1) ans=max(ans, qry(x1, x2, y1, y2, mid+1, r, rt<<1|1));
        return ans;
    }
}T;
```

3.3 CDQ

```
const int N = 200005;
int p1, p2, pos, n, k, nn, ans[N];
struct node{
   int x, y, z, num, ans;
   bool operator == (const node & b) const{
      return x == b.x && y == b.y && z == b.z;
   }
} a[N], tmp[N];
bool cmp(node a, node b){
   if (a.x != b.x) return a.x < b.x;
   if (a.y != b.y) return a.y < b.y;
   return a.z < b.z;
}
bool cmp2(node a, node b){
   //if (a.y != b.y) return a.y < b.y;
   //return a.z < b.z;
   return a.y < b.y;
}
void CDQ(int l, int r){
   if (l == r) {
      a[l].ans = a[l].num - 1;
      return;
   }
   int mid = l + r >> 1;
   CDQ(l, mid); CDQ(mid+1, r);
   pos = l;
   rep(i, mid+1, r+1) {
      while (pos <= mid && a[pos].y <= a[i].y) {
         fen.add(fen.a1, a[pos].z, a[pos].num);
         pos++;
      }
      a[i].ans += fen.sum(fen.a1, a[i].z);
   }
   rep(i, l, pos) fen.add(fen.a1, a[i].z, -a[i].num);
   p1 = l; p2 = mid+1;
   rep(i, l, r+1){
      if (p1 > mid) {tmp[i] = a[p2]; p2++;}
      else if (p2 > r) {tmp[i] = a[p1]; p1++;}
      else if (a[p1].y <= a[p2].y) {tmp[i] = a[p1]; p1++;}
      else  {tmp[i] = a[p2]; p2++;}
   }
   rep(i, l, r+1) a[i] = tmp[i];
}

int main(){
   cin >> n >> k;
   rep(i, 1, n+1) cin >> a[i].x >> a[i].y >> a[i].z;
   sort(a+1, a+n+1, cmp);
   nn = 0;
   rep(i, 1, n+1) {
      if (i > 1 && a[i] == a[i-1]) { a[nn].num++;
      } else {
         a[++nn] = a[i];
         a[nn].num = 1;
      }
   }
   fen.ini(N);
   CDQ(1, nn);
   rep(i, 1, nn+1) ans[a[i].ans] += a[i].num;
   rep(i, 0, n) cout << ans[i] << endl;
   return 0;
}
```

3.4 CartesianTree

```
// desc : bud a cartesion tree from a[0] .. a[n - 1]
// time : O(N)
// !!!! : return rt, a[n] will be rewrite
int ls[N], rs[N];
int cartesionTree(int a[], int n) {
   a[n] = INT_MAX; vi v(1, n);
   fill_n(ls, n, -1), fill_n(rs, n, -1);
   rep(i, 0, n) {
      while (a[v.back()] < a[i]) ls[i] = v.back(), v.pop_back();
      v.pb(rs[v.back()] = i);
   }
   return v[1];
}
```

3.5 Fenwick

```
// index : [1, n]
// time : nlogn
// support : segment add, sum
// !!!! : use before init()!
template<class T>
struct Fenwick{
   static const int N =2e5+7;
   int n;T a1[N],a2[N];
   void ini(int _n){ fill_n(a1+1,n=_n,0);fill_n(a2+1,n=_n,0); }
   void add(T *a,int p,T d) { for(; p<=n; p+=p & -p) a[p]+=d; }
   void add(int l,int r,T d) {
      add(a1, l, d), add(a1, r + 1, -d);
      add(a2, l, d * (l - 1)), add(a2, r + 1, -d * r);
   }
   T sum(T *a,int p) { T r=0; for(; p>=1; p-=p & -p) r+=a[p]; return r; }
   T pre(int p) { return !p ? 0 : sum(a1, p) * p - sum(a2, p);}
   T qry(int l,int r) {return pre(r)-pre(l-1); }
};
```

3.6 IntervalMaximumChangeTimes

```
inline int qry(int L, int R, ll &v, int o, bool spe, int l, int r, int rt) {
  if(L > R) return 0;
  if(!spe) {
    if(ma[rt] < v) return 0;
    if(L <= l && r <= R) {
      if(l == r) return v = ma[rt], 1;
      int mid = l + r >> 1;
      down(l, r, mid, rt);
      if(ma[ls | o] < v) return o ? qry(L, R, v, o, 0, l, mid, ls) : qry(L, R, v, o, 0,
      mid + 1, r, rs);
      int ans = cnt[rt][o] - cnt[ls | o][o] + (o == 0 ? qry(L, R, v, o, 0, l, mid, ls) :
       qry(L, R, v, o, 0, mid + 1, r, rs));
      return v = ma[rt], ans;
    }
  }
  int mid = l + r >> 1, ans = 0;
  down(l, r, mid, rt);
  if(o == 0 && L <= mid) ans += qry(L, R, v, o, 0, l, mid, ls);
  if(R > mid) ans += qry(L, R, v, o, 0, mid + 1, r, rs);
  if(o == 1 && L <= mid) ans += qry(L, R, v, o, 0, l, mid, ls);
  return ans;
}
void up(int l, int r, int rt) {
  ma[rt] = max(ma[ls], ma[rs]);
  rep(o, 0, 2) { ll v = 0; cnt[rt][o] = qry(l, r, v, o, 1, l, r, rt); }
```

```
}
```

3.7 KDT

```
// init
typedef int T; // modify
namespace KDT {
   const int N = 1e6 + 7, D = 2;
   const T INF = 1e9 + 7;
   const db al = 0.75;

   int rt,L,top,W,sta[N];
   struct P{ T x[D]; bool operator < (const P &c) const { return x[W] < c.x[W]; } }p[N];
   struct Node{ T mi[D],ma[D]; int son[2],sz; P val; }nd[N];

   void init() { rt = L = top = 0; }
   int newnode() { return top ? sta[top--] : ++L; }
   void up(int k) {
      rep(i, 0, D) {
         nd[k].mi[i]=nd[k].ma[i]=nd[k].val.x[i];
         rep(o, 0, 2) if(nd[k].son[o]) {
            int s = nd[k].son[o];
            nd[k].mi[i] = min(nd[k].mi[i], nd[s].mi[i]);
            nd[k].ma[i] = max(nd[k].ma[i], nd[s].ma[i]);
         }
      }
      nd[k].sz = 1; rep(i, 0, 2) nd[k].sz += nd[nd[k].son[i]].sz;
   }
   int build(int l,int r,int w) {
      if(l > r) return 0;
      int mid = l + r >> 1, k = newnode();
      W=w,nth_element(p+l,p+mid,p+r+1),nd[k].val=p[mid];
      nd[k].son[0] = build(l,mid-1, (w + 1) % D);
      nd[k].son[1] = build(mid+1,r, (w + 1) % D);
      up(k); return k;
   }
   void pia(int k,int &cnt) {
      if(nd[k].son[0]) pia(nd[k].son[0],cnt);
      p[++cnt]=nd[k].val,sta[++top]=k;
      if(nd[k].son[1]) pia(nd[k].son[1],cnt);
   }
   void check(int &k,int w) {
      bool o = 0;
      rep(i, 0, 2) if(al * nd[k].sz < nd[nd[k].son[i]].sz) o = 1;
      if(o) { int cnt = 0; pia(k,cnt),k = build(1,cnt,w); }
   }
   void ins(P p,int &k,int w) {
      if(!k) {k=newnode(),nd[k].val=p,nd[k].son[0]=nd[k].son[1]=0,up(k);return;}
      ins(p, nd[k].son[nd[k].val.x[w] < p.x[w]], (w + 1) % D);
      up(k),check(k,w);
   }
   // 抄上面这部分就好了，下面部分是视具体题目定的
   // 最近点（曼哈顿距离）
   // O(nsqrt(n))
   T dis(P p,int k) {
        T ans = 0;
        rep(i, 0, D) ans += max(0, p.x[i] - nd[k].ma[i]) + max(0, nd[k].mi[i] - p.x[i]);
        // modify
        return ans;
    }
    T dis(P a, P b) {
        T ans = 0; rep(i, 0, D) ans += abs(a.x[i] - b.x[i]);
        return ans;
    }
    void qry(P p, int k, T &ans) {
        ans=min(ans,dis(p,nd[k].val));
        int ls = nd[k].son[0], rs = nd[k].son[1];
        T dl = ls ? dis(p, ls) : INF;
        T dr = rs ? dis(p, rs) : INF;
        if(dl > dr) swap(dl, dr), swap(ls, rs);
        if(dl<ans) qry(p,ls,ans);
        if(dr<ans) qry(p,rs,ans);
    }
    // 矩形区域的最大值（伪代码）
    // O(n ^ (2 - 1 / D))
    void qry(int u, int &ans) {
        if(no_in || ma < ans) return ;
        if(all_in) { ans = max(ans, ma); return ; }
        if(u_in) ans = max(ans, u_val);
        rep(i, 0, 2) if(nd[u].son[i]) qry(nd[u].son[i], ans);
    }
    // 距离点 u 第 k 远
    priority_queue<ll> ans;
    void init() {
        while(!ans.empty()) ans.pop();
        rep(i, 0, k) ans.push(1);
    }
    ll sqr(int x) { return 1ll * x * x; }
    ll Dis(P p, int u) {
        ll ans = 0;
        rep(d, 0, D) ans += max(sqr(nd[u].mi[d] - p.x[d]), sqr(nd[u].ma[d] - p.x[d]));
        return ans;
    }
    void qry(P p, int u) {
        ll dis = 0; rep(d, 0, D) dis += sqr(nd[u].val.x[d] - p.x[d]);
        ans.push(-dis), ans.pop();
        int ls = nd[u].son[0], rs = nd[u].son[1];
        ll dl = ls ? Dis(p, ls) : -1;
        ll dr = rs ? Dis(p, rs) : -1;
        if(dl > dr) swap(dl, dr), swap(ls, rs);
        if(dr > -ans.top()) qry(p, rs);
        if(dl > -ans.top()) qry(p, ls);
    }
}
```

3.8 LCT

```
struct Node { int val, sum, fa, son[2]; bool rev; };
struct LCT {
    static const int N = ::N;
    Node nd[N]; int sta[N];
    // if(no root) return 1
    bool nrt(int x) {
        int fa = nd[x].fa;
        return nd[fa].son[0] == x || nd[fa].son[1] == x;
    }
    void up(int x) {
        if(!x) return ;
        int ls = nd[x].son[0], rs = nd[x].son[1];
        nd[x].sum = nd[ls].sum + nd[rs].sum + nd[x].val;
    }
    void gao(int x) {
        if(!x) return ;
        nd[x].rev ^= 1, swap(nd[x].son[0], nd[x].son[1]);
    }
    void down(int x) { if(nd[x].rev) gao(nd[x].son[0]), gao(nd[x].son[1]), nd[x].rev =
    0; }
    int id(int u) { return nd[nd[u].fa].son[1] == u; }
    void rot(int x) {
        int y = nd[x].fa, z = nd[y].fa;
        int l = id(x), r = (l ^ 1), s = nd[x].son[r];
        if(nrt(y)) nd[z].son[id(y)] = x; nd[x].son[r] = y; nd[y].son[l] = s;
        if(s) nd[s].fa = y; nd[y].fa = x; nd[x].fa = z;
        up(y), up(x);
    }
    void splay(int x) {
        int top = 0;
        for(int i = x; ; i = nd[i].fa) {
            sta[++top] = i;
            if(!nrt(i)) break;
        }
        while(top) down(sta[top--]);
        while(nrt(x)) {
            int y = nd[x].fa;
            if(nrt(y)) (id(x) ^ id(y)) ? rot(x) : rot(y);
            rot(x);
        }
    }
    void access(int x) { for(int y = 0; x ; y = x, x = nd[x].fa) splay(x), nd[x].son[1]
    = y, up(x); }
    // 换根
    void makeroot(int x) { access(x); splay(x); gao(x); }
    // 找根
    int findroot(int x) {
        access(x); splay(x);
        while(nd[x].son[0]) down(x), x = nd[x].son[0];
        splay(x);
        return x;
    }
    // 加边
    void link(int x, int y) {
        makeroot(x);
        if(findroot(y) != x) nd[x].fa = y;
    }
    // 删边
    void cut(int x, int y) {
        makeroot(x);
        if(findroot(y) == x && nd[y].fa == x && !nd[y].son[0]) nd[y].fa = nd[x].son[1] =
         0, up(x);
    }
    // nd[y]: 路径信息
    void path(int x, int y) { makeroot(x); access(y); splay(y); }
    // 单点修改
    void upd(int x, int c) { splay(x); nd[x].val = c; up(x); }
}T;
```

3.9 LCT_diameter

```
int fir(multiset<int> st) { return sz(st) ? *(st.rbegin()) : 0; }
int sec(multiset<int> st) { return sz(st) > 1 ? *(++st.rbegin()) : 0; }
void Era(multiset<int> &s, int x) { s.erase(s.find(x)); }
struct Node { int fa, son[2], lmx, rmx, mxs, sum; bool rev; multiset<int> chain, path;
};
struct LCT {
   static const int N = 30303;
   Node nd[N]; int sta[N];
   void del(int x, int y) { Era(nd[x].chain, nd[y].lmx), Era(nd[x].path, nd[y].mxs); }
   void add(int x, int y) { nd[x].chain.insert(nd[y].lmx), nd[x].path.insert(nd[y].mxs);
    }
   void init() {
      rep(i, 1, n + m + 1) {
         nd[i].fa = nd[i].son[0] = nd[i].son[1] = nd[i].rev = 0;
         nd[i].lmx = nd[i].rmx = nd[i].mxs = nd[i].sum = 0;
         nd[i].chain.clear(), nd[i].path.clear();
      }
   }
   void up(int x) {
      if(!x) return ;
      int p = x, ls = nd[x].son[0], rs = nd[x].son[1];
      // 以下考虑的都是链 p 与链  p 的所有虚子树
      nd[p].sum = nd[ls].sum + nd[rs].sum + (p > n); // 当前链的长度
      int cha = fir(nd[p].chain); // 从 p 沿虚儿子走的最远距离
      int L = max(cha, nd[ls].rmx) + (p > n); // 从 p 沿父亲走的最远距离
      int R = max(cha, nd[rs].lmx) + (p > n); // 从 p 沿实儿子走的最远距离
      nd[p].lmx = max(nd[ls].lmx, nd[ls].sum + R); // 从链顶出发的最远距离
      nd[p].rmx = max(nd[rs].rmx, nd[rs].sum + L); // 从链底出发的最远距离
      nd[p].mxs = max(nd[ls].mxs, nd[rs].mxs); // mxs[p] 表示当前范围的直径
      Ma(nd[p].mxs, fir(nd[p].path)); // 虚子树的直径
      Ma(nd[p].mxs, nd[ls].rmx + R); // 经过 p 父边的答案
      Ma(nd[p].mxs, nd[rs].lmx + L); // 经过 u 向下实边的答案
      Ma(nd[p].mxs, cha + sec(nd[p].chain) + (p > n)); // 虚子树中到根路径最长的两条拼起来
   }
   void gao(int x) {
      if(!x) return ;
      nd[x].rev ^= 1;
      swap(nd[x].son[0], nd[x].son[1]);
      swap(nd[x].lmx, nd[x].rmx);
   }
   void access(int x) {
      for(int y = 0; x ; y = x, x = nd[x].fa) {
         splay(x);
         int &rs = nd[x].son[1];
         if(y) del(x, y);
         if(rs) add(x, rs);
         rs = y, up(x);
      }
   }
   void link(int x, int y) {
      makeroot(x);
      if(findroot(y) != x) makeroot(y), nd[x].fa = y, add(y, x), up(y);
   }
   int getAns() { access(1); splay(1); return nd[1].mxs; }
}lct;
```

3.10 Rope

```
#include <ext/rope>
using namespace __gnu_cxx;
rope<char> rp;
rp.push_back(ch);               // 在末尾插入字符
rp.insert(cur, 字符数组 );     // 在 cur 处插入字符数组
rp.erase(cur, len);             // 删除 cur 开始的 len 个字符
rp.replace(cur, 字符数组 );    // 删除 cur 处的字符，换成字符数组
rp.copy(cur, len, 字符数组 ); // 复制 cur 处开始的 len 个字符到字符数组
rp.at(cur);                     // 取第 cur 个字符
rp[cur];                        // 取第 cur 个字符
rp.substr(cur, len);            // 提取从 cur 处开始的 len 个字符
rp[i] = rp[i - 1];              // 可持久化, O(1), 直接拷贝根节点
/*
 * 一）翻转操作
 *  1. 维护一正一反两个 rope
 *  2. 翻转等价于交换两个子串
 * 二）区间循环位移
 *  1. 拆成多个子串，重新安排它们的位置
 * 三）区间 a -> b, b -> c, c -> d ... z -> a
 *  1. 维护 26 个 rope
 */
```

3.11 ST

```
// [0,n)
struct ST{
   static const int N = 101010;
   int a[20][N], lg[N];
   void build(int *v, int n){
      rep(i, 2, n + 1) lg[i] = lg[i >> 1] + 1;
      rep(i, 0, n) a[0][i] = v[i];
      rep(i, 1, lg[n] + 1) rep(j, 0, n - (1 << i) + 1) {
         a[i][j] = max(a[i - 1][j], a[i - 1][j + (1 << i >> 1)]);
      }
   }
   int qry(int l, int r){
      if(l > r) swap(l, r);
      int i = lg[r - l + 1];
      return max(a[i][l] , a[i][r + 1 – (1 << i)]);
   }
};
```

3.12 SegIntervalMax

```
// O(nlogn)
// 区间取 max, 区间求和
struct Seg {
   static const int N = ::N << 2;
   ll sum[N]; int mi[N][2], cnt[N];
   void up(int rt) {
      sum[rt] = sum[ls] + sum[rs];
      rep(i, 0, 2) mi[rt][i] = min(mi[ls][i], mi[rs][i]);
      cnt[rt] = 0;
      rep(i, 0, 2) {
         if(mi[rt][0] == mi[ls | i][0]) cnt[rt] += cnt[ls | i];
         else mi[rt][1] = min(mi[rt][1], mi[ls | i][0]);
      }
   }
   void build(int l, int r, int rt) {
      if(l == r) {
         sum[rt] = mi[rt][0] = a[l]; //modify
         mi[rt][1] = inf; cnt[rt] = 1;
         return ;
      }
      int mid = l + r >> 1;
      build(l, mid, ls); build(mid + 1, r, rs); up(rt);
   }
   void gao(int rt, int c) {
      if(c <= mi[rt][0]) return ;
      sum[rt] += 1ll * cnt[rt] * (c – mi[rt][0]);
      mi[rt][0] = c;
   }
   void down(int rt) { gao(ls, mi[rt][0]); gao(rs, mi[rt][0]); }
   void upd(int L, int R, int c, int l, int r, int rt) {
      if(L > R) return ;
      if(L <= l && r <= R && c < mi[rt][1]) return gao(rt, c), void();
      int mid = l + r >> 1; down(rt);
      if(L <= mid) upd(L, R, c, l, mid, ls);
      if(R > mid) upd(L, R, c, mid + 1, r, rs);
      up(rt);
   }
}seg;
```

3.13 Splay

```
// init
// id starts from 1
// if go to vertex p, must splay(p)
struct Node { int val, fa, son[2], cnt, sz; bool rev; };
struct Splay {
   static const int N = ::N;
   int rt, L; Node nd[N];
   int newnode(int c, int fa = 0, int o = 0) {
      nd[++L].fa = fa;
      nd[fa].son[o] = L;
      nd[L].val = c;
      nd[L].son[0] = nd[L].son[1] = nd[L].rev = 0;
      nd[L].cnt = nd[L].sz = 1;
      return L;
   }
   void init(int n) { rt = L = 0; }
   void gao(int u) {
      if(!u) return ;
      nd[u].rev ^= 1, swap(nd[u].son[0], nd[u].son[1]);
   }
   void down(int u) {
      if(nd[u].rev) gao(nd[u].son[0]), gao(nd[u].son[1]), nd[u].rev = 0;
   }
   void up(int u) {
      if(!u) return ;
      int ls = nd[u].son[0], rs = nd[u].son[1];
      nd[u].sz = nd[ls].sz + nd[rs].sz + nd[u].cnt;
   }
   int id(int u) { return nd[nd[u].fa].son[1] == u; }
   void rot(int x) {
      int y = nd[x].fa, z = nd[y].fa;
      int l = id(x), r = (l ^ 1), s = nd[x].son[r];
      if(z) nd[z].son[id(y)] = x; nd[x].son[r] = y; nd[y].son[l] = s;
      if(s) nd[s].fa = y; nd[y].fa = x; nd[x].fa = z;
      up(y), up(x);
   }
   void splay(int x, int g = 0) {
      while(nd[x].fa != g) {
         int y = nd[x].fa, z = nd[y].fa;
         if(z != g) (id(x) ^ id(y)) ? rot(x) : rot(y);
         rot(x);
      }
      if(!g) rt = x;
   }
}T;
```

3.14 fhqTreap

```
// init
// id starts from 1
// 不要修改 0 节点的值
struct Node { int val, cnt, sz, ls, rs; ll r; bool rev;};
struct fhqTreap {
   static const int N = ::N;
   int rt, L; Node nd[N];
   void init() { rt = L = 0; srand(time(0)); }
   ll Rand() { return ((rand() * 1ll << 32) ^ (rand() * 1ll << 16) ^ rand()); }
   int newnode(int c) {
      nd[++L].r = Rand();
      nd[L].val = c;
      nd[L].cnt = nd[L].sz = 1;
      nd[L].ls = nd[L].rs = nd[L].rev = 0;
      return L;
   }
   void up(int x) {
      if(!x) return ;
      int ls = nd[x].ls, rs = nd[x].rs;
      nd[x].sz = nd[ls].sz + nd[rs].sz + nd[x].cnt;
   }
   void gao(int x) {
      if(!x) return ;
      nd[x].rev ^= 1, swap(nd[x].ls, nd[x].rs);
   }
   void down(int x) { if(nd[x].rev) gao(nd[x].ls), gao(nd[x].rs), nd[x].rev = 0; }
   // u -> (<= c) (> c)
   void splitc(int u, int c, int &x, int &y) {
      if(u) {
         down(u);
         if(nd[u].val <= c) x = u, splitc(nd[u].rs, c, nd[u].rs, y);
         else y = u, splitc(nd[u].ls, c, x, nd[u].ls);
         up(u);
      } else x = y = 0;
   }
   // u -> (1 ~ k) (k+1 ~ L)
   // !!!: nd[].cnt == 1
   void splitk(int u, int k, int &x, int &y) {
      if(u) {
         down(u);
         int sz = nd[nd[u].ls].sz;
         if(sz < k) x = u, splitk(nd[u].rs, k - sz - 1, nd[u].rs, y);
         else y = u, splitk(nd[u].ls, k, x, nd[u].ls);
         up(u);
      } else x = y = 0;
   }
   int merge(int x, int y) {
      if(x && y) {
         if(nd[x].r < nd[y].r) { down(x), nd[x].rs = merge(nd[x].rs, y), up(x); return x
         ; }
         else { down(y), nd[y].ls = merge(x, nd[y].ls), up(y); return y; }
      } else return x + y;
   }
}T;
```

3.15 lcSegTree

```
// init
struct Node {
    ll k, b;
    Node() : k(0), b(0) {}
    Node(ll k, ll b) : k(k), b(b) {}
    ll getf(int x) const { return k * x + b; }
};
struct Seg {
    static const int N = ::N << 2;
    Node nd[N], mi[N]; // nd: max val; mi: min val;
    void _upd(Node k, int l, int r, int rt) {
        int mid = l + r >> 1;
        if(k.getf(v[mid]) > nd[rt].getf(v[mid])) swap(k, nd[rt]);
        if(l == r) return ;
        if(min(nd[rt].getf(v[l]), nd[rt].getf(v[r])) >= max(k.getf(v[l]), k.getf(v[r])))
         return ;
        if(nd[rt].k > k.k) _upd(k, l, mid, ls);
        else _upd(k, mid + 1, r, rs);
    }
    void _min(Node k, int l, int r, int rt) {
        int mid = l + r >> 1;
        if(k.getf(v[mid]) < mi[rt].getf(v[mid])) swap(k, mi[rt]);
        if(l == r) return ;
        if(max(mi[rt].getf(v[l]), mi[rt].getf(v[r])) <= min(k.getf(v[l]), k.getf(v[r])))
         return ;
        if(mi[rt].k <= k.k) _min(k, l, mid, ls);
        else _min(k, mid + 1, r, rs);
    }
    void upd(int L, int R, Node c, int l, int r, int rt) {
        if(L > R) return ;
        if(L <= l && r <= R) {
            _upd(c, l, r, rt);
            _min(c, l, r, rt);
            return ;
        }
        int mid = l + r >> 1;
        if(L <= mid) upd(L, R, c, l, mid, ls);
        if(R > mid) upd(L, R, c, mid + 1, r, rs);
    }
    ll qry(int p, int l, int r, int rt) {
        ll ans = max(abs(nd[rt].getf(v[p])), abs(mi[rt].getf(v[p])));
        if(l == r) return ans;
        int mid = l + r >> 1;
        if(p <= mid) ans = max(ans, qry(p, l, mid, ls));
        else ans = max(ans, qry(p, mid + 1, r, rs));
        return ans;
    }
}seg;
```

3.16 perTreap

```
// init
// id starts from 1
// 不要修改 0 节点的值
struct Node { int val, cnt, sz, ls, rs; ll r, sum; bool rev; };
struct fhqTreap {
   static const int N = 3e7;
   int rt[::N], L; Node nd[N];
   void init() { fill_n(rt, L + 1, 0); L = 0; srand(time(0)); }
   ll Rand() { return ((rand() * 1ll << 32) ^ (rand() * 1ll << 16) ^ rand()); }
   int newnode(int c) {
      nd[++L].r = Rand();
      nd[L].val = nd[L].sum = c;
      nd[L].cnt = nd[L].sz = 1;
      nd[L].ls = nd[L].rs = nd[L].rev = 0;
      return L;
   }
```

```
    int newcopy(int x) { nd[++L] = nd[x]; return L; }
    void up(int x) {
        if(!x) return ;
        int ls = nd[x].ls, rs = nd[x].rs;
        nd[x].sz = nd[ls].sz + nd[rs].sz + nd[x].cnt;
        nd[x].sum = nd[ls].sum + nd[rs].sum + nd[x].val;
    }
    void gao(int &x) {
        if(!x) return ;
        x = newcopy(x);
        nd[x].rev ^= 1, swap(nd[x].ls, nd[x].rs);
    }
    void down(int x) { if(nd[x].rev) gao(nd[x].ls), gao(nd[x].rs), nd[x].rev = 0; }
    // u -> (<= c) (> c)
    void splitc(int u, int c, int &x, int &y) {
        if(u) {
            u = newcopy(u), down(u);
            if(nd[u].val <= c) x = u, splitc(nd[u].rs, c, nd[u].rs, y);
            else y = u, splitc(nd[u].ls, c, x, nd[u].ls);
            up(u);
        } else x = y = 0;
    }
    // u -> (1 ~ k) (k+1 ~ L)
    // !!!: nd[].cnt == 1
    void splitk(int u, int k, int &x, int &y) {
        if(u) {
            u = newcopy(u), down(u);
            int sz = nd[nd[u].ls].sz;
            if(sz < k) x = u, splitk(nd[u].rs, k - sz - 1, nd[u].rs, y);
            else y = u, splitk(nd[u].ls, k, x, nd[u].ls);
            up(u);
        } else x = y = 0;
    }
    // sometimes do not need to newcopy
    int merge(int x, int y) {
        if(x && y) {
            if(nd[x].r < nd[y].r) { x = newcopy(x), down(x), nd[x].rs = merge(nd[x].rs, y),
             up(x); return x; }
            else { y = newcopy(y), down(y), nd[y].ls = merge(x, nd[y].ls), up(y); return y;
             }
        } else return x + y;
    }
}T;
```

3.17 动态 dp_bst

```
int n, m, a[N], sz[N], wson[N], f[N][2], h[N][2];
int to[N << 1], ne[N << 1], hd[N], _;
inline void ae(int u, int v) { to[++_] = v, ne[_] = hd[u], hd[u] = _; }
void dfs(int u, int fa) {
    sz[u] = 1;
    for(int i = hd[u]; i; i = ne[i]) if(to[i] != fa) {
        int v = to[i];
        dfs(v, u);
        // upd f[u]
        sz[u] += sz[v];
        (sz[v] > sz[wson[u]]) && (wson[u] = v);
    }
    int s = wson[u];
    // h[u] = f[u]
    if(s) {
        // upd h[u]
    }
}
struct BST {
    Mat val[N], sum[N]; bool isr[N];
    int fa[N], son[N][2], rt, sta[N], top, n;
    inline void up(int x) {
        sum[x] = val[x];
        if(son[x][0]) sum[x] = sum[son[x][0]] * sum[x];
        if(son[x][1]) sum[x] = sum[x] * sum[son[x][1]];
    }
    inline int sbuild(int l, int r) {
        if(l > r) return 0;
        int tot = 0, now = 0;
        rep(i, l, r + 1) tot += sz[sta[i]] - sz[wson[sta[i]]];
        rep(i, l, r + 1) {
            now += sz[sta[i]] - sz[wson[sta[i]]];
            if((now << 1) >= tot) {
                int x = sta[i];
                son[x][0] = sbuild(l, i - 1);
                son[x][1] = sbuild(i + 1, r);
                fa[son[x][0]] = fa[son[x][1]] = x;
                up(x);
                return x;
            }
        }
    }
    int build(int tp, int f) {
        for(int u = tp; u; f = u, u = wson[u]) {
            for(int i = hd[u]; i; i = ne[i]) if(to[i] != f && to[i] != wson[u]) {
                fa[build(to[i], u)] = u;
            }
            // upd val[u]
        }
        top = 0;
        for(int u = tp; u; u = wson[u]) sta[++top] = u;
        return sbuild(1, top);
    }
    void build(int _n) {
        n = _n; rt = build(1, 0);
        rep(i, 1, n + 1) isr[i] = (son[fa[i]][0] != i && son[fa[i]][1] != i);
    }
    inline int upd(int x, int y) {
        // upd h[x], a[x] = y, val[x]
        while(x) {
            if(isr[x] && fa[x]) {
                // get old f[x]
                up(x);
                // get new f[x]
            // u = fa[x], get h[u], val[u]
         } else {
            up(x);
         }
         x = fa[x];
      }
      // get dp[1] by sum[rt]
   }
}bst;
void init() {
   dfs(1, 0);
   bst.build(n);
}
```

3.18 动态 dp_lct

```
int n, m, a[N], f[N][2];
namespace DP {
   struct Node { int fa, son[2]; Mat val, sum; };
   struct LCT {
      inline void up(int x) {
         nd[x].sum = nd[x].val;
         if(ls) nd[x].sum = nd[ls].sum * nd[x].sum;
         if(rs) nd[x].sum = nd[x].sum * nd[rs].sum;
      }
      inline void access(int x) {
         for(int y = 0; x; y = x, x = nd[x].fa) {
            splay(x);
            if(nd[x].son[1]) {
               // upd val[x]
            }
            if(y) {
               // upd val[x]
            }
            nd[x].son[1] = y; up(x);
         }
      }
      inline int upd(int x, int y) {
         access(x); splay(x); nd[x].val.a[1][0] += y - a[x];
         up(x); a[x] = y;
         return max(nd[x].sum.a[0][0], nd[x].sum.a[1][0]);
      }
   }lct;
}
```

3.19 动态 dp_树链剖分

```
int n, a[N]; vi g[N];
namespace DP {
   int sz[N], wson[N], top[N], dep[N], id[N], _, par[N], who[N], leaf[N];
   ll f[N][2], F[N][2];
   struct Mat {
      ll a[3][3];
      inline Mat operator * (const Mat &c) const {
         Mat r; rep(i, 0, 3) rep(j, 0, 3) {
            r.a[i][j] = a[i][0] + c.a[0][j];
            rep(k, 1, 3) r.a[i][j] = max(r.a[i][j], a[i][k] + c.a[k][j]);
         } return r;
      }
      inline void e() { rep(i, 0, 3) rep(j, 0, 3) a[i][j] = (i != j) * (-inf); }
   };
   struct Seg {
      Mat m[N << 2];
      inline void build(int l, int r, int rt) {
         if(l == r) {
            int u = who[l];
            // calc F, f
            // set m[rt]
            return ;
         }
         int mid = l + r >> 1; build(mid + 1, r, rs); build(l, mid, ls);
         m[rt] = m[ls] * m[rs];
      }
      inline void upd(int u, int l, int r, int rt) {
         if(l == r) {
            // set m[rt]
            return ;
         }
         int mid = l + r >> 1;
         (id[u] <= mid) ? upd(u, l, mid, ls) : upd(u, mid + 1, r, rs);
         m[rt] = m[ls] * m[rs];
      }
      inline void qry(int L, int R, int l, int r, int rt, Mat &ans) {
         if(rt == 1) ans.e();
         if(L <= l && r <= R) return ans = ans * m[rt], void();
         int mid = l + r >> 1;
         if(L <= mid) qry(L, R, l, mid, ls, ans);
         if(R > mid) qry(L, R, mid + 1, r, rs, ans);
      }
   }seg;
   struct HeavyChain{
      // if(s) top[s] = top[c], dfs2(s, c, g), leaf[c] = leaf[s];
      // else leaf[c] = c;
   }hc;
   inline pair<ll, ll> qry(int x) {
      Mat tmp; seg.qry(id[x], id[leaf[x]], 1, n, 1, tmp);
      ll f0 = max(tmp.a[0][0], tmp.a[0][1], tmp.a[0][2]);
      ll f1 = max(tmp.a[1][0], tmp.a[1][1], tmp.a[1][2]);
      return mp(f0, f1);
   }
   void upd(int p, int c) {
      F[p][1] += c - a[p], a[p] = c;
      int v = p;
      while(v) {
         int u = top[v], fa = par[u];
         seg.upd(v, 1, n, 1);
         pair<ll, ll> _f = qry(u);
         if(fa) {
            // upd F[fa]
         }
         f[u][0] = _f.fi, f[u][1] = _f.se, v = fa;
      }
   }
   void work() {
      hc.Build(g);
      seg.build(1, n, 1);
   }
}
```

3.20 常见转化

```
/*
 * 单点修改，区间查询 -> 单点修改，前缀查询 -> 后缀修改，单点查询
 * 树剖路径问题：重链区间修改，轻边暴力维护。轻边深度小的点一定在重链上，深度大的一定是 top 。
 */
```

3.21 线段树优化建图

```
struct SegGraph {
   static const int N = :: N << 3;
   int id[N][2], p[N], tim;
   vector<pii> g[N];
   void init() { rep(i, 0, tim) g[i].clear(); tim = 0; }
   void liu(int u, int v, int w) { g[u].pb(mp(v, w)); }
   void build(int l, int r, int rt) {
      int *t = id[rt], *fa = id[rt / 2], mid = l + r >> 1;
      t[0] = ++tim, t[1] = ++tim, liu(t[0], t[1], 0);
      if (rt / 2) liu(fa[0], t[0], 0), liu(t[1], fa[1], 0);
      if (l == r) { p[l] = t[0]; return; }
      build(l, mid, ls); build(mid+1, r, rs);
   }
   void link(int l, int r, int rt, int L, int R, int w, int o) {
      int *t = id[rt], mid = l + r >> 1;
      if (L <= l && R >= r) {
         if (o) liu(t[o], tim, w);
         else liu(tim, t[o], w);
         return;
      }
      if (L <= mid) link(l, mid, ls, L, R, w, o);
      if (R > mid)  link(mid+1, r, rs, L, R, w, o);
   }
   // [l1, r1] -> [l2, r2] weight = w
   void link(int l1, int r1, int l2, int r2, int w, int n) {
      ++tim;
      link(1, n, 1, l1, r1, w, 0);
      link(1, n, 1, l2, r2, w, 1);
   }
} G;
```

3.22 覆盖大于 k 次的矩形面积

```
// 这里是覆盖次数大于 1 次的
```

```
// 这份代码没 down 函数，感觉有问题
struct Seg {
   static const int N = ::N << 2;
   int la[N], len[2][N];
   void up(int rt, int l, int r) {
      if(la[rt] >= 2) {
         len[0][rt] = r - l + 1;
         len[1][rt] = r - l + 1;
      } else if(la[rt] >= 1) {
         len[0][rt] = r - l + 1;
         len[1][rt] = (l == r) ? 0 : len[0][ls] + len[0][rs];
      } else {
         len[0][rt] = (l == r) ? 0 : len[0][ls] + len[0][rs];
         len[1][rt] = (l == r) ? 0 : len[1][ls] + len[1][rs];
      }
   }
   void upd(int L, int R, int c, int l, int r, int rt) {
      if(L <= l && r <= R) {
         la[rt] += c;
         up(rt, l, r);
         return ;
      }
      int mid = l + r >> 1;
      if(L <= mid) upd(L, R, c, l, mid, ls);
      if(R > mid) upd(L, R, c, mid + 1, r, rs);
      up(rt, l, r);
   }
}seg;
```

3.23 高维偏序

```
// 如果有 O2 比较快，不然可能比较慢要手写 bitset
namespace PX{
   const int N = :: N, M = sqrt(N) + 5, K = 7;
   int n, k, B, pos[K][N];
   bitset<N> s[K][M];
   vector<pii> V[K];
   struct node { int d[K]; } a[N];
   void init(int _n, int _k) {
      n = _n; k = _k;
      rep(i, 1, n+1) rep(j, 0, k) cin >> a[i].d[j];
      rep(i, 1, n+1) rep(j, 0, k) V[j].pb(mp(a[i].d[j], i));
      rep(j, 0, k) sort(all(V[j]));
      rep(i, 1, n+1) rep(j, 0, k) {
         a[i].d[j] = lower_bound(all(V[j]), mp(a[i].d[j], i)) - V[j].begin();
         pos[j][a[i].d[j]] = i;
      }
      B = sqrt(n);
      rep(j, 0, k) {
         bitset<N> tmp; int id = 1;
         rep(i, 0, n) {
            tmp.set(pos[j][i]);
            if (i == id * B - 1) s[j][id++] = tmp;
         }
      }
   }
   int qry(node a) {
      bitset<N> ans; ans.set();
      rep(j, 0, k) {
         int ed = lower_bound(all(V[j]), mp(V[j][a.d[j]].fi, n+1)) - V[j].begin() - 1;
         bitset<N> tmp;int id = ed / B, st = id ? (id - 1) * B : 0;
         if (id) tmp = s[j][id - 1];
         rep(i, st, ed+1) tmp[pos[j][i]] = 1;
         ans &= tmp;
      }
      return ans.count();
   }
}
```

4 Game

4.1 Nim 积

```
/*
 * 注：高维硬币游戏
 */
namespace Nim {
   int nimPow(int x, int y) {
      if (x < 2) return x && y; int a = 0;
      while (x < (1 << (1 << a)) || x >= (1 << (1 << (a + 1)))) a++;
      int m = 1 << (1 << a), p = x / m, s = y / m, t = y%m;
      int d1 = nimPow(p, s), d2 = nimPow(p, t);
      return (m*(d1^d2)) ^ nimPow(m / 2, d1);
   }
   int Mul(int x, int y) {
      if (x < y) return Mul(y, x);
      if (x < 2) return x && y; int a = 0;
      while (x < (1 << (1 << a)) || x >= (1 << (1 << (a + 1)))) a++;
      int m = 1 << (1 << a), p = x / m, q = x%m, s = y / m, t = y%m;
      int c1 = Mul(p, s), c2 = Mul(p, t) ^ Mul(q, s), c3 = Mul(q, t);
      return (m*(c1^c2)) ^ c3^nimPow(m / 2, c1);
   }
}
```

4.2 SurNum

```
int sgn(ll x) { return !x ? 0 : (x > 0 ? 1 : -1); }
struct SurNum {
   ll x, k; int op;
   SurNum() { x = k = op = 0; }
   SurNum(ll x, ll k, ll op = 0) :x(x), k(k), op(op) { }
   SurNum(const SurNum &a) { *this = a; }
   inline SurNum Simplify() {
      while (x % 2 == 0 && k > 0) x /= 2, k--;
      return *this;
   }
   friend inline int sgn(const SurNum &a) { return sgn(a.x); }
   inline bool Grow(int kk) {
      Simplify();
      if (kk < k) return 0;
      x *= 1ll << kk - k, k = kk;
      return 1;
   }
   friend inline void grow(SurNum &a, SurNum &b) {
      int k = max(a.k, b.k);
      a.Grow(k), b.Grow(k);
   }
   friend inline int compare(SurNum a, SurNum b) {
      if (a.op < b.op) return -1;
      if (a.op > b.op) return 1;
      if (a.op != 0) return 0;
      int opa = sgn(a), opb = sgn(b);
      if (opa < opb) return -1;
      if (opa > opb) return 1;
      grow(a, b);
      return sgn(a.x - b.x);
   }
   friend inline bool operator < (const SurNum &a, const SurNum &b) {
      return compare(a, b) == -1;
   }
   friend inline bool operator > (const SurNum &a, const SurNum &b) {
      return compare(a, b) == 1;
   }
   friend inline bool operator == (const SurNum &a, const SurNum &b) {
      return compare(a, b) == 0;
   }
   friend inline bool operator != (const SurNum &a, const SurNum &b) {
      return compare(a, b) != 0;
   }
   friend inline bool operator <= (const SurNum &a, const SurNum &b) {
      return compare(a, b) <= 0;
   }
   friend inline bool operator >= (const SurNum &a, const SurNum &b) {
      return compare(a, b) >= 0;
   }
   friend inline SurNum operator - (const SurNum &a) {
      return SurNum(-a.x, a.k, -a.op);
   }
   friend inline SurNum operator + (SurNum a, SurNum b) {
      if (a.op == 1 || b.op == 1) return SurNum(0, 0, 1);
      if (a.op == -1 || b.op == -1) return SurNum(0, 0, -1);
      grow(a, b); return SurNum(a.x + b.x, a.k, 0).Simplify();
   }
   friend inline SurNum operator - (SurNum a, SurNum b) {
      return a + (-b);
   }
   friend inline SurNum operator += (SurNum &a, SurNum b) {
      return a = a + b;
   }
   friend inline SurNum operator -= (SurNum &a, SurNum b) {
      return a = a - b;
   }
   friend inline SurNum operator >> (SurNum a, ll k) {
        return a.k += k, a.Simplify();
    }
    friend inline SurNum getMid(SurNum a, SurNum b) {
        return a + b >> 1;
    }
    inline void print() const {
        printf("SurNum:\n");
        if (op == 1) { printf("+inf\n"); return; }
        if (op == -1) { printf("-inf\n"); return; }
        printf("%lld/%lld\n", x, 1 << k);
    }
    inline static SurNum read() {
        ll a1, a2, a3;
        scanf("%lld%lld%lld", &a1, &a2, &a3);
        return SurNum(a1, a2, a3).Simplify();
    }
} _0(0, 0, 0), _inf(0, 0, 1);
struct SurTri {
    SurNum p, x, q;
    SurTri() { p = x = q = _0; }
    SurTri(SurNum p, SurNum x, SurNum q) :p(p), x(x), q(q) {}
    SurTri(const SurTri &a) { *this = a; }
    SurTri goRight() {
        SurNum y;
        if (x.x >= 0 && x.k == 0) y = x, y.x++; else y = x + q >> 1;
        return SurTri(x, y, q);
    }
    SurTri goLeft() {
        SurNum y;
        if (x.x <= 0 && x.k == 0) y = x, y.x--; else y = p + x >> 1;
        return SurTri(p, y, x);
    }
    void print() {
        printf("\n\nSurTri:\n\n");
        p.print(), x.print(), q.print();
        printf("\nend\n\n");
    }
};
struct SurCalculator {
    int getDir(SurTri S, SurNum a, SurNum b) {
        if (a < S.x && S.x < b) return 0;
        if (a <= S.x && b <= S.x) return -1;
        if (a >= S.x && b >= S.x) return 1;
        assert(0);
    }
    SurNum getValue(SurNum a, SurNum b) {
        int op;
        SurTri S(-_inf, _0, _inf);
        while (op = getDir(S, a, b)) S = ((op == 1) ? S.goRight() : S.goLeft());
        return S.x;
    }
};
```

5 Geo

5.1 1、基础点、向量

```
struct P {
    int quad() const { return sign(y) > 0 || (sign(y) == 0 && sign(x) >= 0); }
    P rot90() { return P(-y, x); }
    P rot(db a) { return P(cos(a) * x - sin(a) * y, cos(a) * y + sin(a) * x); }
    P norm() { return *this / len(); }
};
db rad(P p1, P p2) { return atan2l(det(p1, p2), dot(p1, p2)); } // p1 与 p2 的夹角，有方
向
bool cmp(const pii &a, const pii &b) { // 级角排序
    int o = a > pii(0, 0), t = b > pii(0, 0);
    if(o != t) return o < t;
    return det(a, b) > 0;
}
// 【点集中最近点对】
namespace NearestPoints { // sz(A) <= 1e5
    db solve(int l, int r, vector<P> &p) {
        if(l == r) return 1e100;
        int m = l + r >> 1;
        db Xm = p[m].x, lim = min(solve(l, m, p), solve(m + 1, r, p));
        inplace_merge(p.begin() + l, p.begin() + m + 1, p.begin() + r + 1, [&](P a, P b){
        return a.y < b.y;});
        vector<P> V;
        rep(i, l, r + 1) if(fabs(p[i].x - Xm) <= lim) V.pb(p[i]);
        rep(i, 0, sz(V)) rep(j, i + 1, sz(V)) {
            if(fabs(V[j].y - V[i].y) >= lim) break;
            T dis = (V[i] - V[j]).len();
            lim = min(lim, dis);
        }
        return lim;
    }
    db solve(vector<P> A) {
        sort(all(A), [&](P a, P b){return a.x < b.x;});
        return solve(0, sz(A) - 1, A);
    }
}
// 【最小圆覆盖】
C Mincir(P *p,int n){
    random_shuffle(p , p + n);
    P o = p[0];db r = 0;
    rep(i,1,n) {
        if(sgn(abs(o-p[i])-r) <= 0) continue;
        o = p[i] , r = 0;
        rep(j,0,i) {
            if(sgn(abs(o-p[j])-r) <= 0) continue;
            o = (p[i] + p[j]) / 2 , r = abs(o-p[j]);
            rep(k,0,j) {
                if(sgn(abs(o-p[k])-r) <= 0) continue;
                o = outC(p[i],p[j],p[k]) , r = abs(o-p[k]);
            }}}
    return C(o,r);
}
// 【费马点】
// sqrt((a ^ 2 + b ^ 2 + c ^ 2 + 4 * sqrt(3) * area) / 2)
// 如果有重点，大于 2 的直接用模拟退火法
P fermat(vector<P> p) {
  int n = sz(p); assert(n);
  if(n == 1) return p[0];
  if(n == 2) return (p[0] + p[1]) / 2;
  if(n == 3) {
    db a[3];
    rep(i, 0, 3) a[i] = (p[(i + 2) % 3] - p[(i + 1) % 3]).len();
    rep(i, 0, 3) {
      int j = (i + 1) % 3, k = (i + 2) % 3;
      if(sign(a[i] * a[i] - a[j] * a[j] - a[k] * a[k] - a[j] * a[k]) >= 0) return p[i];
    }
    if(det(p[0], p[1], p[2]) < 0) swap(p[1], p[2]);
    P q1 = (p[2] - p[0]).rot(pi / 3) + p[0];
    P q2 = (p[0] - p[1]).rot(pi / 3) + p[1];
    return isLL(L(q1, p[1]), L(q2, p[2]));
  }
  auto Rand = [&] () { return rand() % 10000 / 5000 * pi; };
  P ans(0, 0); rep(i, 0, n) ans = ans + p[i]; ans = ans / n;
  db len = 0; rep(i, 0, n) len += (ans - p[i]).len();
  db t = 10000; // modify
  while(t > eps) {
    db ang = Rand();
    P np(ans.x + t * sin(ang), ans.y + t * cos(ang));
    db k = 0; rep(i, 0, n) k += (np - p[i]).len();
    if(sign(len - k) > 0) ans = np, len = k;
    t *= 0.999;
  }
  return ans;
}
```

5.2 2、线段、直线、曲线

```
// 【直线交点】
P isLL(L l1, L l2) {
   db s1 = det(l2.b - l2.a, l1.a - l2.a);
   db s2 = -det(l2.b - l2.a, l1.b - l2.a);
   return (l1.a * s2 + l1.b * s1) / (s1 + s2);
}
P isLL(L l, db a, db b, db c) { // ax + by + c = 0
   db u = a * l.a.x + b * l.a.y + c;
   db v = -(a * l.b.x + b * l.b.y + c);
   return (l.a * v + l.b * u) / (u + v);
}
P isLL(db a0, db b0, db c0, db a1, db b1, db c1) {
   db d = a0 * b1 - a1 * b0;
   return P(b0 * c1 - b1 * c0, a1 * c0 - a0 * c1) / d;
}
// 【线相交判定】
bool isSSr(const L &a, const L &b){
   db c1 = det(a.t - a.s, b.s - a.s), c2 = det(a.t - a.s, b.t - a.s);
   db c3 = det(b.t - b.s, a.s - b.s), c4 = det(b.t - b.s, a.t - b.s);
   return sign(c1) * sign(c2) < 0 && sign(c3) * sign(c4) < 0;
}
bool isSS(L a,L b){
   db c1 = det(a.t - a.s, b.s - a.s), c2 = det(a.t - a.s, b.t - a.s);
   db c3 = det(b.t - b.s, a.s - b.s), c4 = det(b.t - b.s, a.t - b.s);
   return sign(c1) * sign(c2) <= 0 && sign(c3) * sign(c4) <= 0 &&
      sign(max(a.s.x, a.t.x) - min(b.s.x, b.t.x)) >= 0 &&
      sign(max(b.s.x, b.t.x) - min(a.s.x, a.t.x)) >= 0 &&
      sign(max(a.s.y, a.t.y) - min(b.s.y, b.t.y)) >= 0 &&
      sign(max(b.s.y, b.t.y) - min(a.s.y, a.t.y)) >= 0;
}
bool isLS(P a1, P a2, P b1, P b2) { // 判断直线线段是否相交（端点也算）
   db c1 = det(a2 - a1, b1 - a1), c2 = det(a2 - a1, b2 - a1);
   return sign(c1) * sign(c2) <= 0;
}
// 【点到线距离】
db disToL(L l, P p) {
  return fabs(det(l.a, p, l.b) / (l.b - l.a).len());
}
db disToS(L l, P p) {
  return sign(dot(l.a, p, l.b)) * sign(dot(l.b, p, l.a)) == 1 ? disToL(l, p) : min((p -
  l.a).len(), (p - l.b).len());
}
// 【线到线距离】
db disSS(L a, L b){
   if(isSS(a, b)) return 0;
   return min(min(disToSeg(b, a.s), disToSeg(b, a.t)), min(disToSeg(a, b.s), disToSeg(a,
    b.t)));
}
```

5.3 3、凸包

```
// 【求凸包】
vector<P> convexHull(vector<P> ps) {
  int n = sz(ps); if(n <= 1) return ps;
  sort(all(ps)); vector<P> qs;
  for(int i = 0; i < n; qs.pb(ps[i++])) {
    while(sz(qs) > 1 && sign(det(qs[sz(qs) - 2], qs.back(), ps[i])) <= 0) qs.pop_back();
  }
  for(int i = n - 2, t = sz(qs); i >= 0; qs.pb(ps[i--])) {
    while(sz(qs) > t && sign(det(qs[sz(qs) - 2], qs.back(), ps[i])) <= 0) qs.pop_back();
  }
  qs.pop_back(); return qs;
}
// 【凸包最远点对】
db diameter(vector<P> A) {
   int n = sz(A);
   if(n <= 1) return 0;
   int l = 0, r = 0;
   rep(i, 1, n) (A[i] < A[l]) && (l = i), (A[r] < A[i]) && (r = i);
   db res = (A[l]-A[r]).len();
   int i = l, j = r;
   do (++(det(A[(i + 1) % n]- A[i], A[(j + 1) % n] - A[j]) >= 0 ? j : i)) %= n,
      res = max(res, (A[i] - A[j]).len());
   while(i != l || j != r);
   return res;
}
// 【动态凸包】
// O(nlogn)
// 插入点，询问点在不在凸包内部（包括边界）
namespace DCH {
   map<int, P> h1, h2;
   bool ao(P a, P b, P c) {
      // 包括边界：小等于
      return (b.y - a.y) * 1ll * (c.x - b.x) <= (c.y - b.y) * 1ll * (b.x - a.x);
   }
   bool in(map<int, P> &h, P p) {
      if(!sz(h)) return 0;
      if(p.x < h.begin()->se.x || p.x > h.rbegin()->se.x) return 0;
      auto l = h.lower_bound(p.x);
      if(p.x == l->se.x) return p.y <= l->se.y;
      auto r = l--;
      return ao(l->se, p, r->se);
   }
   void ins(map<int, P> &h, P p) {
      if(in(h, p)) return ;
      h[p.x] = p;
      auto pos = h.find(p.x);
      while(1) {
         auto l = pos; if(l == h.begin()) break; --l;
         auto ll = l; if(ll == h.begin()) break; --ll;
         if(ao(ll->se, l->se, p)) h.erase(l); else break;
      }
      while(1) {
         auto r = pos; r++; if(r == h.end()) break;
         auto rr = r; rr++; if(rr == h.end()) break;
         if(ao(p, r->se, rr->se)) h.erase(r); else break;
      }
   }
   void ins(int x, int y) { ins(h1, P(x, y)); ins(h2, P(x, -y)); }
   bool in(int x, int y) { return in(h1, P(x, y)) && in(h2, P(x, -y)); }
}
// 【凸包交】
namespace ConvecIntersection{ // ?
   const int N = 1005;
   struct Rec {
      P d[10];int dn;// d[dn] = d[0]
      P operator [] (const int&n) {return d[n];}
   }r[N];
   typedef pair<db,int> pdi;
   int n;pdi res[1000005];
   db getLoc(P a,P b,P p){
      if(sgn(b.x - a.x)) return (p.x - a.x) / (b.x - a.x);
      return (p.y - a.y) / (b.y - a.y);
   }
   db work() {
      db rt=0;
      rep(i,0,n) rep(j,0,r[i].dn){
         int sz=0;
         res[sz++] = pdi(0,0);res[sz++] = pdi(1,0);
         rep(t,0,n) {
            if(t == i) continue;
            rep(g,0,r[t].dn) {
               int du = sgn((r[i][j+1] - r[i][j]) / (r[t][g] - r[i][j]));
               int dv = sgn((r[i][j+1] - r[i][j]) / (r[t][g+1] - r[i][j]));
               if(!du && !dv) {
                  if(sgn((r[i][j+1] - r[i][j]) * (r[t][g+1] - r[t][g])) < 0 || i < t){
                     res[sz++] = pdi(getLoc(r[i][j] , r[i][j+1] , r[t][g]) , 1);
                     res[sz++] = pdi(getLoc(r[i][j] , r[i][j+1] , r[t][g+1]) , -1);
                  }} else {
                     db s1 = (r[i][j] - r[t][g]) / (r[t][g+1] - r[t][g]);
                     db s2 = (r[t][g+1] - r[t][g]) / (r[i][j+1] - r[t][g]);
                     if(du >= 0 && dv < 0) res[sz++] = pdi(s1 / (s1 + s2) , 1);
                     else if(du < 0 && dv >= 0) res[sz++] = pdi(s1 / (s1 + s2) , -1);
                  }}}
         sort(res , res + sz);
         int cnt = 0; --sz;
         rep(t,0,sz) {
            cnt += res[t].se;
            if(cnt == 0 && sgn(res[t].fi - res[t+1].fi)) {
               db a = res[t].fi;
               if(a < 0) a = 0; if(a > 1) break;
               db b = res[t+1].fi;
               if(b < 0) continue; if(b > 1) b = 1;
               rt += ((r[i][j+1] - r[i][j]) * a + r[i][j]) / ((r[i][j+1]-r[i][j]) * b +
               r[i][j]);
            }}}
      return rt / 2;}}
```

5.4 4、三角形

```
// 【心】
P outC(P A, P B, P C) { // 外心
   P b = B - A, c = C - A;
   db dB = b.len2(), dC = c.len2(), d = 2 * det(b, c);
   return A - P(b.y * dC - c.y * dB, c.x * dB - b.x * dC) / d;
}
P baryC(P p[], int n) { // 重心
   P fz(0, 0); db fm = 0;
   rep(i, 1, n - 1) {
      db t = det(p[0], p[i], p[i + 1]);
      fm += t;
      fz = fz + (p[0] + p[i] + p[i + 1]) * t / 3;
   }
   return fz / fm;
}
```

5.5 5、多边形

```
// 【平面图欧拉定理】 V + F - E = 2
// 【简单多边形求面积交】
db polyInter(vector<P> &p, vector<P> &q) {
   int n = sz(p), m = sz(q);
   if(n < 3 || m < 3) return 0;
   // if(area(p) < 0) reverse(all(p));
   // if(area(q) < 0) reverse(all(q));
   db ans = 0;
   rep(i, 1, n - 1) {
      P p1 = p[i], p2 = p[i + 1];
      bool f1 = 0;
      if(det(p[0], p1, p2) < 0) swap(p1, p2), f1 = 1;
      rep(j, 1, m - 1) {
         P q1 = q[j], q2 = q[j + 1];
         bool f2 = 0;
         if(det(q[0], q1, q2) < 0) swap(q1, q2), f2 = 1;
         vector<P> ps({p[0], p1, p2});
         convexCut(ps, L(q[0], q1));
         convexCut(ps, L(q1, q2));
         convexCut(ps, L(q2, q[0]));
         db res = f1 == f2 ? area(ps) : -area(ps);
         ans += res;
      }
   }
   return fabs(ans);
}
```

5.6 6、圆

```
// 【两圆关系】
// 注意相等关系
// 相离4: 外切3: 相交2: 内切1: 内含0:
int relCC(C A, C B) { // 两圆关系
   db dis = (A.o - B.o).len();
   if(sign(dis - (A.r + B.r)) == 1) return 4;
   if(sign(dis - (A.r + B.r)) == 0) return 3;
   if(sign(dis - fabs(A.r - B.r)) == 1) return 2;
   if(sign(dis - fabs(A.r - B.r)) == 0) return 1;
   return 0;
}
// 【点圆切点】
bool tanCP(O c, P p0, P &p1, P &p2) {
   db x = (p0 - c.o).len2(), d = x - c.r * c.r;
   if(d < eps) return 0;
   P p = (p0 - c.o) * (c.r * c.r / x);
   P det = ((p0 - c.o) * (-c.r * sqrt(d) / x)).rot90();
   p1 = c.o + p + det;
   p2 = c.o + p - det;
   return 1;
}
// 【圆圆切点】
vector<P> tanCC(const C &c1, const C &c2) {
   vector<P> res;
   db dis = (c1.o - c2.o).len();
   if(sign(dis - (c1.r + c2.r)) == 0) {
      res.pb(c1.o + (c2.o - c1.o) * c1.r / (c1.r + c2.r));
   }
   if(sign(dis - fabs(c1.r - c2.r) == 0)) {
      res.pb(c1.o + (c2.o - c1.o) * c1.r / (c1.r - c2.r));
   }
   return res;
}
// 【直线和圆求交】
bool isCL(O a, L l, P &p1, P &p2) {
  db x = dot(l.a - a.o, l.b - l.a);
  db y = (l.b - l.a).len2();
  db d = x * x - y * ((l.a - a.o).len2() - a.r * a.r);
  if(sign(d) < 0) return 0;
  d = max(d, 0.);
  P p = l.a - ((l.b - l.a) * (x / y)), det = (l.b - l.a) * (sqrt(d) / y);
  p1 = p - det, p2 = p + det; // dir : l.a -> l.b
  return 1;
}
// 【圆与三角形交面积】
db areaCT(db r,P s,P t) { // 需要除 2
   P p1, p2;
   bool f = isCL(C(P(0, 0), r), L(s, t), p1, p2);
   if(!f) return r * r * rad(s, t);
   bool b1 = sign(s.len2() - r * r) == 1 , b2 = sign(t.len2() - r * r) == 1;
   if(b1 && b2) {
      if(sign(dot(s - p1, t - p1)) <= 0 && sign(dot(s - p2, t - p2) <= 0))
         return r * r * (rad(s, p1) + rad(p2, t)) + det(p1, p2);
      else return r * r * rad(s, t);
   } else if(b1) return r * r * rad(s, p1) + det(p1, t);
   else if(b2) return r * r * rad(p2, t) + det(s, p2);
   return det(s, t);
}
// 【圆与多边形交面积】
db areaCPoly(C c, vector<P> p) {
   int n = sz(p);
   db ans = 0;
   rep(i, 0, n) {
      P u = p[i], v = p[(i + 1) % n];
      ans += areaCT(c.r, u - c.o, v - c.o);
   }
   return fabs(ans) / 2;
}
// 【圆交】
namespace CircleIntersection{ // ?
   struct E{
      P p;T ang;int delta;
      E(){} E(P p,T ang,int delta):p(p),ang(ang),delta(delta){}
      bool operator < (const E&b) const {return ang<b.ang;}
   };
   bool overlap(C a,C b) {return sgn(a.r-b.r-abs(a.o-b.o))>=0;}
   void solve(C *c,int n,T *ans) {
      memset(ans , 0 , sizeof(T) * (n + 1));
      rep(i,0,n) {
         int cnt=1;
         vector<E> evt;
         rep(j,0,i) if(c[i]==c[j]) cnt++;
         rep(j,0,n) if(j!=i&&!(c[i]==c[j])&&overlap(c[j],c[i])) cnt++;
         rep(j,0,n) if(j!=i){
            vector<P> pts=insCC(c[i],c[j]);
            if(sz(pts)) {
```

```
            T a[2];
            rep(j,0,2) a[j]=(pts[j]-c[i].o).arg();
            evt.pb(E(pts[0],a[0],1));
            evt.pb(E(pts[1],a[1],-1));
            cnt += a[0] > a[1];
         }
      }
      if(!sz(evt)) ans[cnt] += pi*c[i].r*c[i].r;
      else{
         sort(all(evt));
         evt.pb(evt.front());
         rep(j,0,sz(evt)-1) {
            cnt+=evt[j].delta;
            ans[cnt] += evt[j].p / evt[j+1].p / 2;
            db ang = evt[j + 1].ang - evt[j].ang;
            if(ang < 0) ang += pi * 2;
            ans[cnt] += ang * c[i].r * c[i].r / 2 - sin(ang) * c[i].r * c[i].r / 2;
         }}}}}
```

5.7 7、3D

```
// 【最小球覆盖】
P3 MinSphere(vector<P3> p) {
  int n = sz(p); assert(n);
  db t = 1; P3 ans(0, 0, 0);
  rep(i, 0, n) ans = ans + p[i]; ans = ans / n;
  while(t > eps) {
    int j = -1; db ret = -1;
    rep(i, 0, n) {
      db tmp = (p[i] - ans).len();
      if(ret < tmp) ret = tmp, j = i;
    }
    ans = ans + (p[j] - ans) * t;
    t *= 0.999;
  }
  return ans;
}
// 【三维向量变换】
struct Mat {
  db a[4][4];
  void set() { rep(i, 0, 4) rep(j, 0, 4) a[i][j] = 0; }
  void e() { rep(i, 0, 4) a[i][i] = 1; }
  Mat operator * (const Mat &c) {
    Mat r; r.set();
    rep(i, 0, 4) rep(j, 0, 4) rep(k, 0, 4) r.a[i][j] += a[i][k] * c.a[k][j];
    return r;
  }
};
Mat kpow(Mat a, int b) {
  Mat r; r.set(); r.e();
  while(b) {
    if(b & 1) r = r * a;
    a = a * a;
    b >>= 1;
  }
  return r;
}
Mat translate(db tx, db ty, db tz) { // 平移，以下矩阵均为左乘
  db p[4][4] = {
    1, 0, 0, tx,
    0, 1, 0, ty,
    0, 0, 1, tz,
    0, 0, 0, 1};
  Mat r; rep(i, 0, 4) rep(j, 0, 4) r.a[i][j] = p[i][j]; return r;
}
Mat scale(db a, db b, db c) { // 缩放
  db p[4][4] = {
    a, 0, 0, 0,
    0, b, 0, 0,
    0, 0, c, 0,
    0, 0, 0, 1};
  Mat r; rep(i, 0, 4) rep(j, 0, 4) r.a[i][j] = p[i][j]; return r;
}
Mat rotate(P3 s, db a) { // 绕 s 为轴旋转 a 度，右手方向
  db l = s.len(), x = s.x / l, y = s.y / l, z = s.z / l, si = sin(a), co = cos(a);
  db p[4][4] = {
    co + (1 - co) * x * x, (1 - co) * x * y - si * z, (1 - co) * x * z + si * y, 0,
    (1 - co) * y * x + si * z, co + (1 - co) * y * y, (1 - co) * y * z - si * x, 0,
    (1 - co) * z * x - si * y, (1 - co) * z * y + si * x, co + (1 - co) * z * z, 0,
    0, 0, 0, 1};
  Mat r; rep(i, 0, 4) rep(j, 0, 4) r.a[i][j] = p[i][j]; return r;
}
```

5.8 HalfPlane_n2

```
// l: a->b 逆时针方向
void convexCut(vector<P> &p, L l) {
   vector<P> q;
   rep(i, 0, sz(p)) {
      P p1 = p[i], p2 = p[(i + 1) % sz(p)];
      int d1 = sign(det(l.a, l.b, p1));
      int d2 = sign(det(l.a, l.b, p2));
      if(d1 >= 0) q.pb(p1);
      if(d1 * d2 < 0) q.pb(isLL(L(p1, p2), l));
   } p = q;
}
// ax + by + c >= 0
void convexCut(vector<P> &p, db a, db b, db c) {
   vector<P> q;
   rep(i, 0, sz(p)) {
      P p1 = p[i], p2 = p[(i + 1) % sz(p)];
      int d1 = sign(a * p1.x + b * p1.y + c);
      int d2 = sign(a * p2.x + b * p2.y + c);
      if(d1 >= 0) q.pb(p1);
      if(d1 * d2 < 0) q.pb(isLL(L(p1, p2), a, b, c));
   } p = q;
}
```

5.9 HalfPlane_nlogn

```
struct P {
   int quad() const { return sign(y) > 0 || (sign(y) == 0 && sign(x) >= 0); }
};
struct L {
   // ax + by + c >= 0, (a != 0 || b != 0)
   L(db a, db b, db c) {
      if(sign(a)==0) {
         this->a=P(0,-c/b);this->b=P(sign(b),-c/b);
      } else if(sign(b)==0) {
         this->a=P(-c/a,0);this->b=P(-c/a,-sign(a));
      } else {
         if(sign(c)!=0) {
            int x=sign(c)*sign(det(P(-c/a,0), P(0,-c/b)));
            if(x==1) this->a=P(-c/a,0),this->b=P(0,-c/b);
            else this->a=P(0,-c/b),this->b=P(-c/a,0);
         } else {
            this->a=P(0,0);this->b=P(sign(b),sign(b)*(-a/b));
         }
      }
   }
   bool includer(const P &p) const { return sign(det(b - a, p - a)) > 0; }
   bool include(const P &p) const { return sign(det(b - a, p - a)) >= 0; }
   // 向内（右手方向）推
   L push(db len) {
      P det = (b - a).rot90().norm() * len;
      return L(a + det, b + det);
   }
};
bool sameDir(L l0, L l1) {
   P a = l0.a - l0.b, b = l1.a - l1.b;
   return sign(det(a, b)) == 0 && sign(dot(a, b)) == 1;
}
bool operator < (const P &a,  const P &b) {
   if(a.quad() != b.quad()) return a.quad() < b.quad();
   return sign(det(a, b)) > 0;
}
bool operator < (const L &l0, const L &l1) {
   if(sameDir(l0, l1)) return l1.includer(l0.a);
   return (l0.b - l0.a) < (l1.b - l1.a);
}
bool check(L u, L v, L w) { return w.include(isLL(u, v)); }
deque<L> halfPlane(vector<L> l) {
   sort(all(l)); deque<L> q;
   rep(i, 0, sz(l)) {
      if(i && sameDir(l[i], l[i - 1])) continue;
      while(sz(q) > 1 && !check(q[sz(q) - 2], q.back(), l[i])) q.pop_back();
      while(sz(q) > 1 && !check(q[1], q[0], l[i])) q.pop_front();
      q.pb(l[i]);
   }
   while(sz(q) > 2 && !check(q[sz(q) - 2], q.back(), q[0])) q.pop_back();
   while(sz(q) > 2 && !check(q[1], q[0], q.back())) q.pop_front();
   return q;
}
```

5.10 MaxAreaPoly

```
ld solve_poly(vi &S) {
   assert(sz(S) > 0);
   int sum = 0, hi = S[0];
   vi vals;
   rep(i, 1, sz(S)) {
      int cur = S[i];
      if (cur > hi) swap(cur, hi);
      sum += cur;
      vals.pb(cur);
   }
   if (sum <= hi) return 0;
   auto getAngle = [&](ld D) -> ld{
      ld tot = 0;
      for (int l : vals) tot += 2 * asin(ld(l) / ld(D));
      return tot;
   };
   bool isReflex = (getAngle(hi) < PI);
   auto tooSmall = [&](ld D) {
      ld ang = getAngle(D);
      ld hiAng = 2 * asin(ld(hi) / ld(D));
      if (isReflex) return ang < hiAng;
      else return ang + hiAng >= 2 * PI;
   };
   ld mi = hi, ma = hi + 1;
   int numExpand = 0;
   while (tooSmall(ma)) numExpand++, ma += (ma - mi);
   rep(tim, 0, 50 + numExpand) {
      ld md = mi + (ma - mi) / 2;
      if (tooSmall(md)) mi = md;
      else ma = md;
   }
   ld D = mi, area = 0;
   for (int l : vals) area += ld(l) * sqrt(ld(D) * ld(D) - ld(l) * ld(l)) / 4;
   ld hiArea = ld(hi) * sqrt(ld(D) * ld(D) - ld(hi) * ld(hi)) / 4;
   if (isReflex) area -= hiArea;
   else area += hiArea;
   return area;
}
```

5.11 MaxAreaTri

```
// O(n ^ 2)
void maxAreaTri(P *p, int n, P &a, P &b, P &c) {
   int i = 0, j = 1, k = 2;
   a = p[i], b = p[j], c = p[k];
   T res = area(a, b, c), cur = res, tmp;
   do {
      while(1) {
         while(cur <= (tmp = area(p[i], p[j], p[(k + 1) % n]))) (++k) %= n, cur = tmp;
         if(cur <= (tmp = area(p[i], p[(j + 1) % n], p[k]))) (++j) %= n, cur = tmp;
         else break;
      }
      if(cur > res) a = p[i], b = p[j], c = p[k], res = cur;
      (++i) %= n;
      if(i == j) (++j) %= n;
      if(j == k) (++k) %= n;
      cur = area(p[i], p[j], p[k]);
   } while(i);
}
```

5.12 MinAreaTri

```
// 无重点、三点共线
// O(n^2log_2n)
struct P { int x, y, ind, u, v; };
namespace MinAreaTri {
   const int N = 2020;
   const ll inf = 4e18;
   int n, m, pos[N];
   P p[N], l[N * N];
   bool cmp(const P &x, const P &y) { return det(x, y) < 0; }
   void solve() {
      sort(p + 1, p + 1 + n);
      rep(i, 1, n + 1) p[i].ind = i, pos[i] = i;
      m = 0; rep(i, 1, n + 1) rep(j, i + 1, n + 1) {
         l[++m] = p[i] - p[j];
         if(l[m].x < 0) l[m].x *= -1, l[m].y *= -1;
         else if(l[m].x == 0 && l[m].y < 0) l[m].y *= -1;
         l[m].u = i, l[m].v = j;
      }
      sort(l + 1, l + 1 + m, cmp);
      mi = inf, ma = 0;
      rep(i, 1, m + 1) {
         int u = l[i].u, v = l[i].v;
         int pu = pos[u], pv = pos[v];
         if(pu > pv) swap(u, v), swap(pu, pv);
         if(pu == 1 || pv == n) continue;
         mi = min(mi, area(p[pu - 1], p[pu], p[pv + 1], p[v]));
         ma = max(ma, area(p[1], p[pu], p[n], p[v]));
         swap(p[pu], p[pv]);
         swap(pos[u], pos[v]);
      }
      cout << mi << " " << ma << endl;
   }
}
```

5.13 凹四边形计数

```
const int N = 1010;
int n; P p[N], q[N]; ll s[N];
namespace CNT {
   bool gao(P a) { return a.y > 0 || (a.y == 0 && a.x >= 0); }
   bool cmp(P a, P b) {
      bool o = gao(a), t = gao(b);
      if(o != t) return o > t;
      return det(a, b) > 0;
   }
   void solve(int u, ll &ans) {
      rep(i, 1, n + 1) q[i] = p[i]; swap(q[1], q[u]);
      rep(i, 2, n + 1) q[i] = q[i] - p[u];
      sort(q + 2, q + n + 1, cmp);
      int k = n; while(k >= 2 && q[k].y <= 0) --k;
      int j = k, cnt = 0;
      per(i, k + 1, n + 1) {
         while(j >= 2 && det(q[j], q[i]) > 0) --j, ++cnt;
         s[i] = s[i + 1] + cnt;
      }
      int c = j = k + 1;
      rep(i, 2, k + 1) {
         while(c <= n && det(q[i], q[c]) > 0) ++c;
         while(j <= n && det(q[i], q[j]) >= 0) ++j;
         ans += s[j] + (n - j + 1) * 1ll * (c - k - 1);
      }
   }
   ll solve() {
      ll ans = 0; rep(i, 1, n + 1) solve(i, ans);
      return ans;
   }
}
```

5.14 平面图转对偶图

```
struct Planar {
   static const int N = 101010, M = 101010;
   // ps id starts from 0
   vector<P> ps;
   // cnte id starts from 0
   int cnte, ne[M];
   bool vis[M];
   // u -> (v, cnte)
   vector<pii> g[N];
   pii E[M];
   vector<db> areas;

   void init() {
      rep(i, 0, sz(ps)) g[i].clear();
      fill_n(vis, cnte, false);
      ps.clear(); cnte = 0;
      areas.clear();
   }
   void adde(int u, int v) {
      g[u].pb(mp(v, cnte));
      E[cnte++] = mp(u, v);
      g[v].pb(mp(u, cnte));
      E[cnte++] = mp(v, u);
   }
   int V;
   bool cmp(const pii &i, const pii &j) {
      P a = ps[i.fi] - ps[V], b = ps[j.fi] - ps[V];
      int o = P(0, 0) < a, t = P(0, 0) < b;
      if(o != t) return o < t;
```

```
        return det(a, b) > 0;
    }
    void go(int e) {
        db res = 0;
        while(!vis[e]) {
            res += det(ps[E[e].se], ps[E[e].fi]); vis[e] = 1;
            e = ne[e ^ 1];
        }
        if(res > 0) areas.pb(res / 2);
    }
    void solve(const vector<P> &_ps, const vector<pii> &es) {
        init(); ps = _ps;
        for(auto e : es) adde(e.fi, e.se);
        rep(i, 0, sz(ps)) {
            V = i; sort(all(g[i]), cmp);
            rep(j, 0, sz(g[i])) {
                ne[g[i][j].se] = g[i][(j + 1) % sz(g[i])].se;
            }
        }
        rep(i, 0, cnte) if(!vis[i]) go(i);
    }
};
```

5.15 旋转卡壳

```
// 凸包都是顺时针给出
// 【凸包直径】点 - 点
T diameter(vector<P> ps) {
    n = sz(ps); T ans = 0;
    if(n <= 1) return 0;
    if(n == 2) return (ps[1] - ps[0]).len();
    rep(i, 0, n) {
        P t = ps[i] - ps[(i + 1) % n];
        while(det(t, ps[(p + 1) % n] - ps[p]) > 0) (++p) %= n;
        ans = max(ans, (ps[i] - ps[p]).len());
        ans = max(ans, (ps[(i + 1) % n] - ps[p]).len());
    }
    return ans;
}
// 【凸包宽度】点 - 边
// 【凸包间的最大距离】点 - 点
// 【凸包间的最小距离】
T solve(P p[], int n, P q[], int m) {
    int o = 0, t = 0; T ans = inf;
    rep(i, 1, n) if(p[i].y > p[o].y) o = i;
    rep(i, 1, m) if(q[i].y < q[t].y) t = i;
    rep(i, 0, n) {
        P a = p[(o + 1) % n] - p[o]; db tmp;
        while((tmp = det(a, q[(t + 1) % m] - q[t])) < 0) (++t) %= m;
        if(sign(tmp)) ans = min(ans, disToSeg(L(p[o], p[(o + 1) % n]), q[t]));
        else ans = min(ans, disSS(L(p[o], p[(o + 1) % n]), L(q[t], q[(t + 1) % m])));
        (++o) %= n;
    }
    return ans;
}
T work(P p[], int n, P q[], int m) {
    return min(solve(p, n, q, m), solve(q, m, p, n));
}
// 【凸包最小面积外接矩形】
T solve(vector<P> ps) {
    int n = sz(ps); T ans = 1e18;
    int p = 1, l = 1, r;
    rep(i, 0, n) {
        P t = ps[i] - ps[(i + 1) % n];
        while(det(t, ps[(p + 1) % n] - ps[p]) > 0) (++p) %= n;
        while(dot(t, ps[(l + 1) % n] - ps[l]) < 0) (++l) %= n;
        r = (p + 1) % n;
        while(dot(t, ps[(r + 1) % n] - ps[r]) > 0) (++r) %= n;
        ll et = abs(det(ps[p], ps[i], ps[(i + 1) % n]));
        ll ot = abs(dot(t, ps[l] - ps[r]));
        ans = min(ans, (db)et * ot / t.len2());
    }
    return ans;
}
// 【凸包最小周长外接矩形】
```

6 Graph

6.1 2-sat

```
struct TwoSat {
    static const int N = ::N << 1;
    int dfn[N], low[N], id[N], st[N], _st, _, cc;
    vi g[N];
    int mark[N], n;
    void init(int _n) { per(i, 0, (n = _n << 1)) g[i].clear(); }
    int new_node() { rep(i, 0, 2) g[n++].clear(); return n / 2 - 1; }
    /// optionals begin
    void addedge(int a, int va, int b, int vb) { // va 选了 vb 必选
        a = a << 1 | va; b = b << 1 | vb;
        g[a].pb(b); g[b ^ 1].pb(a ^ 1);
    }
    void add_set(int a, int va) { a = a << 1 | va; g[a ^ 1].pb(a); } // va 必选
    void add_then(int a, int va, int b, int vb) { // va 和 vb 不能同时取
        addedge(a, va, b, vb ^ 1);
    }
    void add_or (int a, int va, int b, int vb) { // va 和 vb 不能同时不取
        addedge(a, va ^ 1, b, vb);
    }
    void add_xor(int a, int va, int b, int vb) { // va 和 vb 同时取或同时不取
        addedge(a, va, b, vb);
        addedge(b, vb, a, va);
    }
    // 需要 sz(vu) 个额外的 dp 变量
    void add_at_most_one(vector<pii> vu) {
        int pre = -1;
        rep(i, 0, sz(vu)) {
            int a = vu[i].fi, va = vu[i].se;
            int dpi = new_node();
```

```
      addedge(a, va, dpi, 1);
      if (i) {
         addedge(pre, 1, dpi, 1);
         addedge(pre, 1, a, va ^ 1);
      }
      pre = dpi;
   }
}
// optionals end
void dfs(int c, vi g[]){
   dfn[c] = low[c] = ++cc;
   st[_st++] = c;
   for(auto t : g[c])
      if(!dfn[t]) dfs(t, g), low[c] = min(low[c], low[t]);
      else if(!id[t]) low[c] = min(low[c], dfn[t]);
   if(low[c] == dfn[c]){
      ++_;
      do{id[st[--_st]]=_;}while(st[_st] != c);
   }
}
void find(){
   fill_n(dfn, n, cc=0);
   fill_n(low, n, _st=0);
   fill_n(id, n, _=0);
   rep(i, 0, n) if(!dfn[i]) dfs(i, g);
   rep(i, 0, n) --id[i];
   return;
}
bool solve() { // 构造任意解
   find();
   for (int i = 0; i < n; i += 2) {
      if (id[i] == id[i + 1]) return 0;
      mark[i >> 1] = (id[i] > id[i + 1]);
   }
   return 1;
}
int col[N], ans[N], tot;
bool dfs(int u) {
   if (col[u] == -1) return 0;
   if (col[u] ==  1) return 1;
   ans[tot++] = u;
   col[u] = 1; col[u ^ 1] = -1;
   for (auto v : g[u]) if (!dfs(v)) return 0;
   return 0;
}
bool solve2() { // 构造字典序最小解
   for (int i = 0; i < n; i += 2) if (!col[i]) {
      tot = 0;
      if (!dfs(i)) {
         rep(j, 0, tot) col[ans[j]] = col[ans[j] ^ 1] = 0;
         if (!dfs(i ^ 1)) return 0;
      }
   }
   return 1;
}
```

```
}ts;
```

6.2 BCC

```
// key contains the id of edges
// _ starts from 0
namespace BCC{
    const int N = 202020;
    vi key , bcc[N];
    int dfn[N] , low[N] , id[N] , st[N] , _st , _;
    void dfs(int c,int dep,vector<pii> g[]){
        int cc=0;st[_st++]=c;
        dfn[c]=low[c]=dep;
        for(auto e:g[c]){
            int t=e.fi;
            if(!dfn[t]){
                dfs(t,dep+1,g);
                low[c]=min(low[c],low[t]);
                if(low[t]>dfn[c]) key.pb(e.se);
            } else if(dfn[t] != dfn[c] - 1 || cc++)
                low[c] = min(low[c] , dfn[t]);
        }
        if(low[c]==dfn[c]){
            do{id[st[--_st]]=_;}while(st[_st]!=c);
            _++;
        }
    }
    int solve(int n,vector<pii> g[]){
        fill_n(dfn,n,_=0);
        fill_n(low,n,_st=0);
        fill_n(bcc,n,key=vi());
        rep(i,0,n) if(!dfn[i]) dfs(i,1,g);
        rep(i,0,n) for(auto j:g[i]) if(id[i]!=id[j.fi])
            bcc[id[i]].pb(id[j.fi]);
        return _;
    }
};
```

6.3 CircleCount

```
struct circle4 {
   static const int N = 1e5 + 7;
   int n, m, u, v, x, y;
   bool vis[N];
   // cnt3,4 中为包含 i 号点的三 , 四元环数量
   ll f[N][5], du[N], D[N], cnt4[N], cnt3[N], cntl[N], t, ans;
   priority_queue<pii> q;
   vi w[N], gg[N], d2, d1;
   set<int> g[N];
   void dfs(int u, int d, int fa) {
      if (d == 2) { d2.pb(u); w[u].pb(fa); return; }
      if (d == 1) d1.pb(u), vis[u] = 1;
      for (auto v : g[u]) if (v != fa) dfs(v, d+1, u);
   }
   void solve(int n, vi gg[]) {
      rep(i, 1, n+1) {
         D[i] = du[i] = sz(gg[i]); cnt3[i] = cnt4[i] = 0;
         for (auto v : gg[i]) g[i].insert(v);
      }
      rep(i, 1, n+1) q.push(mp(du[i], i));
      rep(i, 1, n+1) {
         rep(j, 1, 5) f[i][j] = 0; f[i][0] = 1;
      }
      rep(i, 1, 5) rep(j, 1, n+1) for (auto v : gg[j]) f[j][i] += f[v][i-1];
      while (!q.empty()) {
         x = q.top().se; y = q.top().fi; q.pop();
         if (du[x] != y) continue;
         dfs(x, 0, -1);
         for (auto u : d2) {
            ll s = sz(w[u]);
            for (auto v : w[u]) {
               cnt4[v] += s - 1;
               if (vis[u]) cnt3[v]++, t++;
            }
            cnt4[x] += s * (s - 1) / 2;
            cnt4[u] += s * (s - 1) / 2;
            w[u].clear();
         }
         for(auto u : d1) vis[u] = 0; d1.clear(); d2.clear();
         cnt3[x] += t / 2; t = 0;
         for (auto u : g[x]) {
            q.push(mp(--du[u], u));
            g[u].erase(x);
         }
      }
      //以第一次产生重复位置分类计数
      rep(i, 1, n+1) { // 计算边数为 4 的链数
         ans = f[i][4];
         //第一次重复为第 4 步
         ans -= cnt4[i] * 2; ll ans3 = f[i][3] - D[i] * D[i] - 2 * cnt3[i];
         for (auto v : gg[i]) ans3 -= D[v] - 1; ans -= ans3; // 边数为 3 的链数
         for (auto v : gg[i]) ans -= 2 * cnt3[v]; ans += 4 * cnt3[i];
         //第一次重复为第 3 步
         ans -= 2 * cnt3[i] * D[i]; for (auto v : gg[i]) ans -= (D[v] - 1) * D[v];
         //第一次重复为第 2 步
         ans -= D[i] * f[i][2];
         cntl[i] = ans;
      }
   }
} c4;
```

6.4 DCC

```
// cactus: n multi by 2
// key is cuts
// dcc i->j , i(points) , j(bcc_block)
// st is stack
// _st is top of stack
// _ is number of dcc
// can handle isolate point and not connected graph and muti edge
// can handle self circle ?
namespace DCC{
    const int N = 202020;
    vi key , dcc[N];
    int dfn[N] , low[N] , st[N] , _st , _;
    void dfs(int c,int dep,const vi g[]){
        int cc=0,out=1<dep;st[_st++]=c;
        dfn[c]=low[c]=dep;
        for(auto t:g[c])
            if(!dfn[t]){
                dfs(t,dep+1,g);
                low[c]=min(low[c],low[t]);
                if(low[t]>=dfn[c]){
                    if(++out==2) key.pb(c);
                    while(st[--_st]!=t) dcc[st[_st]].pb(_);
                    dcc[c].pb(_);dcc[t].pb(_++);
                }
            } else if(dfn[t] != dfn[c] - 1 || cc++)
                low[c] = min(low[c] , dfn[t]);
    }
    int solve(int n,const vi g[]){// n is size of points
        fill_n(dfn,n,_=0);
        fill_n(low,n,_st=0);
        fill_n(dcc,n,key=vi());
        rep(i,0,n) if(!dfn[i]) dfs(i,1,g);
        rep(i,0,n) if(sz(dcc[i]) == 0) dcc[i].pb(_++);
        return _;
    }
}
```

6.5 DLX

```
struct DLX{
#define FOR(i, ne, t) for(int i = ne[t]; i != t; i = ne[i])
   static const int N = 2e4 + 8, D = 4, len = 16;
   int n, m, tim, ansd, row[N], col[N], s[N], ans[N], l[N], r[N], u[N], d[N];
   pair<pii, int> pos[N]; string ss[100];
   void init(int _m) {
      m = _m;
      rep(i, 0, m+1) l[i] = i-1, r[i] = i+1, u[i] = d[i] = i;
      l[0] = m, r[m] = 0, tim = m+1;
      rep(i, 0, m+1) s[i] = 0;
   }
   void add(int R, const vi &tmp){
      int first = tim;
      rep(i, 0, sz(tmp)) {
         int c = tmp[i];
         l[tim] = tim-1, r[tim] = tim+1, u[tim] = u[c], d[tim] = c;
         u[c] = tim; d[u[tim]] = tim;
         row[tim] = R, col[tim] = c;
         tim++, s[c]++;
      }
      if (sz(tmp)) l[first] = tim-1, r[tim-1] = first;
   }
```

```
  inline void remove(int c) {
    l[r[c]] = l[c]; r[l[c]] = r[c];
    FOR(i, d, c) FOR(j, r, i) u[d[j]] = u[j], d[u[j]] = d[j], --s[col[j]];
  }
  inline void restore(int c) {
    FOR(i, u, c) FOR(j, l, i) u[d[j]] = j, d[u[j]] = j, ++s[col[j]];
    l[r[c]] = c; r[l[c]] = c;
  }
  bool dance(int dep) {
    if (!r[0]) return ansd = dep, 1;
    int c = r[0];
    FOR(i, r, 0) if (s[c] > s[i]) c = i;
    remove(c);
    FOR(i, d, c) {
      ans[dep] = row[i];
      FOR(j, r, i) remove(col[j]);
      if (dance(dep+1)) return 1;
      FOR(j, l, i) restore(col[j]);
    }
    restore(c); return 0;
  }
  vi tmp;
  void ins(int x, int y, int c) {
    n++; pos[n] = mp(mp(x, y), c);
    int p = ((x - 1) / D * D + (y - 1) / D) * len + c;
    tmp[0] = ((x - 1) * len + y);
    tmp[1] = (len * len * 1 + (x - 1) * len + c);
    tmp[2] = (len * len * 2 + (y - 1) * len + c);
    tmp[3] = (len * len * 3 + p);
    add(n, tmp);
  }
  void work() {
    tmp.resize(4);
    while (cin >> ss[1]) {
      n = 0; init(len * len * 4);
      rep(i, 1, len+1) {
        if (i > 1) cin >> ss[i];
        rep(j, 1, len+1) {
          if (ss[i][j-1] == '-') rep(k, 1, len+1) ins(i, j, k);
          else ins(i, j, ss[i][j-1] - 'A' + 1);
        }
      }
      bool ok = dance(1);
      if (ok) {
        rep(i, 1, ansd) {
          //cout << ans[i] << " \n"[i == ansd - 1];
          int p = ans[i], x = pos[p].fi.fi, y = pos[p].fi.se, c = pos[p].se;
          ss[x][y - 1] = c + 'A' - 1;
        }
        rep(i, 1, len+1) cout << ss[i] << endl;
        cout << endl;
      }
    }
  }
} T;
```

6.6 DMST

```
// id starts from 0
// can handle multi edge, self ring
struct edge {int u, v, d, U, V;bitset<1005> b;};
struct DMST{
  static const int N = ::N , M = N * N , inf = 2e9;
  edge e[M];int n, m, vis[N], pre[N], id[N], index[N], Pre[N];
  bitset<1005> fang;
  int in[N];
  void ini(int n) {this->n = n, m = 0;}
  void addedge(int u, int v, int d) {e[m] = edge({u,v,d,u,v}); e[m].reset();e[m].b[m] =
   1;m++;}
  int run(int root){
    int ans = 0;
    while(1){
      rep(i, 0, n) in[i] = inf;
      rep(i, 0, m){
        int u = e[i].u ,v = e[i].v;
        if(e[i].d < in[v] && u != v) in[v] = e[i].d, pre[v] = u, index[v] = i;
      }
      rep(i, 0, n) {
        if(i == root) continue;
        if(in[i] == inf) return -1;
        fang ^= e[index[i]].b;
      }
      int cnt = 0;in[root] = 0;
      memset(id, -1, sizeof(*id)*n);
      memset(vis, -1, sizeof(*vis)*n);
      rep(i, 0, n){
        ans += in[i]; int v = i;
        int t = index[i];
        while(vis[v] != i && id[v] == -1 && v!=root) vis[v] = i, v = pre[v];
        if(v != root && id[v] == -1) {
          for(int u=pre[v];u != v;u = pre[u]) id[u] = cnt;
          id[v] = cnt++;
        }
      }
      if(cnt == 0) break;
      rep(i, 0, n) if(id[i] == -1) id[i] = cnt++;
      rep(i, 0, m) {
        int v=e[i].v;
        e[i].u = id[e[i].u]; e[i].v = id[e[i].v];
        if(e[i].u != e[i].v) {e[i].d -= in[v];e[i].b ^= e[index[v]].b;}
      }
      n = cnt; root = id[root];
    }
    return ans;
  }
} dmst;
```

6.7 Dinic

```
// [0,n) init!!
// double need eps
template<class T>
struct Dinic{
    const static int N = 10101 , M = N * 10;
    int s , t , n , h[N] , cur[N] , lv[N] , q[N] , e , ne[M] , to[M];
    T cap[M] , flow;
    void liu(int u,int v,T w){ to[e] = v;ne[e] = h[u];cap[e] = w;h[u] = e++;}
    void link(int u,int v,T w){ liu(u , v , w);liu(v , u , 0);}
    void ini(int _n = N) { fill(h , h + (n=_n) , -1);e = 0;}
    bool bfs(){
        int L = 0 , R = 0;
        fill(lv , lv + n , -1);
        lv[q[R++] = s] = 0;
        while(L < R && !~lv[t]){
            int c = q[L++];
            for(int k = h[c]; ~k ; k = ne[k])
                if(cap[k] > 0 && !~lv[to[k]])
                    lv[q[R++] = to[k]] = lv[c] + 1;
        }
        return ~lv[t];
    }
    T dfs(int c,T mx){
        if(c == t) return mx;
        T ret = 0;
        for(int &k = cur[c]; ~k; k = ne[k]){
            if(lv[to[k]] == lv[c] + 1 && cap[k] > 0){
                T flow = dfs(to[k] , min(mx , cap[k]));
                ret += flow;cap[k] -= flow , cap[k^1] += flow;mx -= flow;
                if(!mx) return ret;
            }
        }
        lv[c] = -1;
        return ret;
    }
    T run(int _s,int _t){
        s = _s , t = _t;
        flow = 0;
        while(bfs()){
            copy(h , h + n , cur);
            flow += dfs(s, ~0U>>1);
        }
        return flow;
    }
};
```

6.8 DominatorTree

```
const int N = 1e5 + 7;
vi revg[N], g[N], buf[N], ord;
int stamp, vis[N], dfn[N], fa[N];
int fs[N], mins[N], dom[N], sem[N], buf2[N];
void dfs(int u) {
   vis[u] = stamp; dfn[u] = sz(ord); ord.pb(u);
   for (auto v : g[u]) if (vis[v] != stamp) fa[v] = u, dfs(v);
}
int find(int u) {
   if (u == fs[u]) return u;
   int v = fs[u];
   fs[u] = find(fs[u]);
   if (~mins[v] && dfn[sem[mins[v]]] < dfn[sem[mins[u]]]) mins[u] = mins[v];
   return fs[u];
}
void mark(int s) {
   ord.clear(); ++stamp; dfs(s);
   for (auto u : ord) fs[u] = u, mins[u] = buf2[u] = -1;
   per(i, 1, sz(ord)) {
      int u = ord[i], p = fa[u]; sem[u] = p;
      for(auto v : revg[u]) if (vis[v] == stamp) {
         if (dfn[v] > dfn[u]) find(v), v = sem[mins[v]];
         if (dfn[v] < dfn[sem[u]]) sem[u] = v;
      }
      buf[sem[u]].pb(u); mins[u] = u; fs[u] = p;
      per(j, 0, sz(buf[p])) {
         int v = buf[p][j]; find(v);
         if (sem[v] == sem[mins[v]]) dom[v] = sem[v]; else buf2[v] = mins[v];
      }
      buf[p].clear();
   }
   dom[ord[0]] = ord[0];
   for (auto u : ord) if (~buf2[u]) dom[u] = dom[buf2[u]];
}
```

6.9 DualMST

对偶图最小生成树，等于平面图所有边边权和减去平面图最大生成树。

6.10 EulerianPath

```
vi ans; bool vis[N]; int p[N];
vector<pii> g[N];
void dfs(int u) {
   for( ; p[u] < sz(g[u]); ++p[u]) {
      auto v = g[u][p[u]];
      if(!vis[abs(v.se)]) {
         vis[abs(v.se)] = 1;
         dfs(v.fi);
         ans.pb(-v.se);
      }
   }
}
```

6.11 FindCircle

```
// 支持基环树森林和自环重边
const int N = 1e5 + 7;
vector<pair<pii>, int> g[N]; // 点编号边权边编号
```

```
int tim, dfn[N], fa[N], d[N], k;
vi cir[N];
int ne[N]; // 有向图的出度
int id[N]; // 点属于的环编号
void dfs(int u, int pre) { // pre 为边编号
   dfn[u] = ++tim;
   rep(i, 0, sz(g[u])) {
      if (g[u][i].se == pre) continue;
      int v = g[u][i].fi.fi, w = g[u][i].fi.se;
      if (dfn[v] && dfn[v] <= dfn[u]) {
         k++;
         int p = u; cir[k].pb(p); id[p] = k;
         if (p != v) {do { p = fa[p]; cir[k].pb(p); id[p] = k;
         } while (p != v);}
         if (sz(cir[k]) > 1 && ne[cir[k][0]] != cir[k][1]) reverse(all(cir[k]));
         continue;
      }
      if (!dfn[v]) {fa[v] = u; d[v] = d[u] + w; dfs(v, g[u][i].se);}
   }
}
```

6.12 Gomory-HuTree

```
Dinic<int> G;
struct GHT{
   static const int N = 1e5 + 100, M = 17; // (1 << M) > n
   int id[N], tmp[N], n, f[N][M], h[N][M], dep[N];
   vector<pii> g[N];
   void ini(int _n) { n = _n; G.ini(n + 5); rep(i, 1, n+1) id[i] = i, g[i].clear(); }
   void link(int u, int v, int w) { G.link(u, v, w); G.link(v, u, w);}
   void solve(int l, int r) {
      if (l == r) return;
      int s = id[l], t = id[l+1];
      for(int i = 0; i < G.e; i += 2) G.cap[i] += G.cap[i+1], G.cap[i+1] = 0;
      int w = G.run(s, t);
      g[s].pb(mp(t, w));
      g[t].pb(mp(s, w));
      int cl = l, cr = 0;
      rep(i, l, r+1) {
         if (G.lv[id[i]] != -1) id[cl++] = id[i];
         else tmp[cr++] = id[i];
      }
      rep(i, 0, cr) id[cl + i] = tmp[i];
      solve(l, cl - 1);
      solve(cl, r);
   }
   void dfs(int u, int fa) {
      dep[u] = dep[fa] + 1;
      for (auto v : g[u]) if (v.fi != fa) {
         f[v.fi][0] = u; h[v.fi][0] = v.se;
         rep(i, 1, M) {
            f[v.fi][i] = f[f[v.fi][i-1]][i-1];
            h[v.fi][i] = min(h[v.fi][i-1], h[f[v.fi][i-1]][i-1]);
         }
         dfs(v.fi, u);
      }
   }
   void build() { solve(1, n); dfs(1, 0); }
   int get(int u, int v) { // 注意 long long
      int res = pw(30);
      if (dep[u] < dep[v]) swap(u, v);
      per(i, 0, M) if (dep[f[u][i]] >= dep[v]) res = min(res, h[u][i]), u = f[u][i];
      per(i, 0, M) if (f[u][i] != f[v][i]) res = min(res, min(h[u][i], h[v][i])), u = f[
      u][i], v = f[v][i];
      if (u != v) res = min(res, min(h[u][0], h[v][0]));
      return res;
   }
} tr;
```

6.13 Lindstrom__Gessel__Viennot__Lemma

```
/*
 * 对于一张无边权的 DAG 图，给定 n 个起点和对应的 n 个终点，这 n 条不相交路径的方案数为矩阵
 * e(a1,b1),e(a1,b2)...e(a1,bn)
 * e(a2,b1),e(a2,b2)...e(a2,bn)
 * ....
 * ....
 * e(an,b1),e(an,b2)...e(an,bn)
 * 的行列式。

 * 即 M[i][j]=e(ai,bj)
 * e(a,b) 为 a 到 b 的路径方案数
 */
```

6.14 ManhattanDistance

```
(x, y) -> (x + y, x - y)            Manhattan distance -> Chebyshev distance
(x, y) -> (x + y >> 1, x - y >> 1)  Chebyshev distance -> Manhattan distance
```

6.15 ManhattanDistanceMST

```
// 曼哈顿最小距离生成树（可以求最大）
// 这份代码处理的区域是 Y 轴右转 45 度
namespace MMST {
   const int N = 101010, inf = 1e9 + 7;
   vector<pair<int, pii> > E;
   vi V;
   // 最大只要把这里所有 mi 改成 ma 就行了
   pii mi[N];
   void init() { rep(i, 1, sz(V) + 1) mi[i] = mp(inf, inf); }
   void upd(int p, pii c) {
      p = sz(V) + 1 - p;
      for( ; p <= sz(V); p += lb(p)) mi[p] = min(mi[p], c);
   }
   pii qry(int p) {
      p = sz(V) + 1 - p;
      pii ans = mp(inf, inf);
      for( ; p >= 1; p ^= lb(p)) ans = min(ans, mi[p]);
```

```
        return ans;
    }
    int F(int x) { return lower_bound(all(V), x) – V.begin() + 1; }
    void _solve(vector<pair<pii, int> > v) {
        V.clear();
        rep(i, 0, sz(v)) v[i].fi.se –= v[i].fi.fi, V.pb(v[i].fi.se);
        sort(all(V));
        V.erase(unique(all(V)), V.end());
        sort(all(v));
        reverse(all(v));
        init();
        for(auto u : v) {
            pii t = qry(F(u.fi.se));
            int s = u.fi.fi * 2 + u.fi.se;
            if(t.se != inf) E.pb(mp(t.fi – s, mp(t.se, u.se)));
            upd(F(u.fi.se), mp(s, u.se));
        }
    }
    void solve(vector<pair<pii, int> > v) {
        _solve(v);
        rep(i, 0, sz(v)) swap(v[i].fi.fi, v[i].fi.se);
        _solve(v);
        rep(i, 0, sz(v)) v[i].fi.fi *= –1;
        _solve(v);
        rep(i, 0, sz(v)) swap(v[i].fi.fi, v[i].fi.se);
        _solve(v);
    }
}
```

6.16 MaxMatch

```
namespace MaxMatch {
    const int N = 1050;
    int link[N], vis[N], use[N], in[N];
    queue<int> Q;
    int dfs(int u, vi g[]) {
        for(auto v : g[u]) {
            if(!vis[v]) {
                vis[v] = 1;
                if(!link[v] || dfs(link[v], g)) { return link[v] = u, 1; }
            }
        }
        return 0;
    }
    int solve(int n, int m, vi g[]) {
        fill_n(link, m+1, 0);
        int ret = 0;
        rep(i, 1, n+1) {
            fill_n(vis, m+1, 0);
            ret += dfs(i, g);
        }
        return ret;
    }
    void MVC(int n, vi g[]) {
        fill_n(vis, n+1, 0);
        per(i, 1, n+1) link[link[i]] = i;
        rep(i, 1, n+1) if (!link[i]) vis[i] = use[i] = 1, Q.push(i);
        while (!Q.empty()) {
            int u = Q.front(); Q.pop();
            if (use[u] == 1) {
                for (auto v : g[u]) {
                    use[v] = 2;
                    if (!vis[v]) vis[v] = 1, Q.push(v);
                }
            }else {
                int v = link[u];
                use[v] = 1;
                if (!vis[v]) vis[v] = 1, Q.push(v);
            }
        }
        rep(i, 1, n+1) if (link[i] && !use[link[i]]) use[i] = 2;
        return;
    }
}
```

6.17 Max_clique_BK

```
// g[i][i] should be 0
// g[i] is i's edge
// index [0..N)
// O(n ^ 3)
typedef unsigned long long T;
struct BK {
    static const int N = 100; T g[N];
    inline int ctz(T s){ return s ? __builtin_ctzll(s) : 64;}
    int n, ans;
    void ini(int _n) {
        //per(i, 0, n = _n) g[i] = (1ull << n) – 1 – (1ull << i); }
        n = _n;rep(i, 0, n) g[i] = 0;
        rep(i, 0, n) rep(j, 0, n) if (a[i][j]) g[i] |= 1ull << j;
    }
    void gao(T cur, T can, T ban) {
        if (!can && !ban) { ans = max(ans, __builtin_popcountll(cur)); return; }
        if (!can) return;
        int piv = ctz(can | ban), ret = 0;
        T z = can & ~g[piv];
        for(int u = ctz(z); u < n; u += ctz(z >> (u + 1)) + 1) {
            gao(cur | (1ull << u), can & g[u], ban & g[u]);
            can ^= 1ull << u, ban |= 1ull << u;
        }
    }
    int run() { gao(ans = 0, (1ull << n) – 1, 0); return ans; }
}  bk;
```

6.18 Max_clique_fastest

```
const int N = 130;
typedef bool BB[N];
struct Maxclique {
    const BB *e; int pk, lv; db Tlimit;
    struct ve {int i, d; ve(int i): i(i),d(0) {}}; //ve : Vertex
    struct sc {int a, b; sc(     ): a(0),b(0) {}}; //sc : StepCount
    typedef vector<ve> ves; ves V;                 //ves: Vertices
    typedef vector<int> cc; cc  Q, QMAX;           //cc : ColorClass
    vector<cc> C;
    vector<sc> S;
    Maxclique(BB *conn, int sz, const db tt = 0.025): pk(0), lv(1), Tlimit(tt) {
       rep(i, 0, sz) V.pb(ve(i)); e = conn;
       C.resize(sz + 1);
       S.resize(sz + 1);
    }
    static bool desc_deg(const ve &a, const ve &b) { return a.d > b.d; }
    void ini_col(ves &v) { per(i, 0, sz(v)) v[i].d = min(i, v[0].d) + 1; }
    void set_deg(ves &v) { rep(i, 0, sz(v)){v[i].d = 0; rep(j, 0, sz(v)) v[i].d += e[v[i
    ].i][v[j].i]; } }
    void deg_sort(ves &R) { set_deg(R); sort(all(R), desc_deg); }
    bool cut1(int pi , cc  &va) { rep(i, 0, sz(va)) if (e[pi][va[i]]) return true; return
     false; }
    void cut2(ves &va, ves &vb) { rep(i, 0, sz(va) - 1) if (e[va.back().i][va[i].i]) vb.
    pb(va[i].i); }
    void co_sort(ves &R) {
       int j = 0, maxno = 1, min_k = max(sz(QMAX) - sz(Q) + 1, 1);
       rep(i, 1, 3) C[i].clear();
       rep(i, 0, sz(R)) {
          int pi = R[i].i, k = 1;
          while (cut1(pi, C[k])) k++;
          if (k > maxno) C[(maxno = k) + 1].clear(); C[k].pb(pi);
          if (k < min_k) R[j++].i = pi;
       }
       if (j > 0) R[j - 1].d = 0;
       rep(k, min_k, maxno + 1) rep(i, 0, sz(C[k])) R[j].i = C[k][i], R[j++].d = k;
    }
    void exp_dyn(ves &R) {  // expand_dyn
       S[lv].a += S[lv - 1].a - S[lv].b;
       S[lv].b  = S[lv - 1].a;
       for (; sz(R); Q.pop_back(), R.pop_back()) {
          if (sz(Q) + R.back().d <= sz(QMAX)) return;
          Q.pb(R.back().i);
          ves Rp; cut2(R, Rp);
          if (sz(Rp)) {
             if ((db) S[lv].a / ++pk < Tlimit) deg_sort(Rp);
             co_sort(Rp); S[lv++].a++;
             exp_dyn(Rp); --lv;
          } else if (sz(Q) > sz(QMAX)) QMAX = Q;
       }
    }
    void mcqdyn(int *mxc, int &sz) { // mcqdyn(int maxclique, int &siz)
       set_deg(V); sort(all(V), desc_deg);
       ini_col(V); rep(i, 0, sz(V) + 1) S[i].a = S[i].b = 0;
       exp_dyn(V); per(i, 0, sz(QMAX)) mxc[i] = QMAX[i];
       sz = sz(QMAX);
    }
};
```

6.19 MinCostMaxFlow

```
// [0,n) , init!! , inf modify
template<class U,class V>
struct MCMF{
    static const int N = 6000, M = 201010;
    int h[N], ing[N], pre[N], to[M], ne[M], e, s, t, n;
    U cap[M]; V dis[N], cost[M];
    void ini(int _n = N){ fill(h , h + (n=_n) , -1);e = 0;}
    void liu(int u,int v,U c,V w){ to[e] = v;ne[e] = h[u];cap[e] = c;cost[e] = w;h[u] =
    e++;}
    void link(int u,int v,U c,V w){ liu(u,v,c,w);liu(v,u,0,-w); }
    bool spfa(){
        queue<int> Q;
        fill(dis, dis+n, inf);
        Q.push(s), ing[s] = 1, dis[s] = 0;
        while(!Q.empty()){
            int c = Q.front(); Q.pop(); ing[c] = 0;
            for(int k = h[c]; ~k; k = ne[k]) if (cap[k] > 0) {
                int v = to[k];
                if(dis[c] + cost[k] < dis[v]){
                    dis[v] = dis[c] + cost[k];
                    pre[v] = k;
                    if(!ing[v]) Q.push(v) , ing[v] = 1;
                }
            }
        }
        return dis[t] != inf;
    }
    U flow; V mincost;
    pair<U, V> run(int _s,int _t){
        s = _s , t = _t;
        flow = mincost = 0;
        while(spfa()){
            U pl = inf; int p, k;
            for(p = t;p != s;p = to[k^1]) pl = min(pl, cap[k = pre[p]]);
            for(p = t;p != s;p = to[k^1]) cap[k = pre[p]] -= pl, cap[k^1] += pl;
         mincost += pl * dis[t];
            flow += pl;
        }
        return mp(flow, mincost);
    }
};
```

6.20 SCC

```
// _ starts from 0
namespace SCC{
    const int N = 100050;
    int dfn[N],low[N],id[N],st[N],_st,_,cc;
    void dfs(int c,vi g[]){
        dfn[c]=low[c]=++cc;
        st[_st++]=c;
        for(auto t:g[c])
```

```
            if(!dfn[t]) dfs(t,g),low[c]=min(low[c],low[t]);
            else if(!id[t]) low[c] =min(low[c],dfn[t]);
        if(low[c]==dfn[c]){
            ++_;
            do{id[st[--_st]]=_;}while(st[_st]!=c);
        }
    }
    vi ng[N];
    int solve(int n,vi g[]){
        fill_n(dfn,n,cc=0);
        fill_n(low,n,_st=0);
        fill_n(id,n,_=0);
        rep(i,0,n) if(!dfn[i]) dfs(i,g);
        rep(i,0,n) --id[i];
        fill_n(ng,_,vi());
        rep(i,0,n) for(auto j:g[i]) if(id[i]!=id[j]) ng[id[i]].pb(id[j]);
        return _;
    }
}
```

6.21 SteinerTree

```
// 要视图的情况使用 spfa, dijstra, 多源 bfs
const int N = 11, M = 10, inf = 0x3f3f3f3f;
int n, m, k, a[N][N], st[N][N], dp[1 << M][N][N], S, ans;
bool use[N][N], vis[1 << M][N][N];
int dx[] = {1, -1, 0, 0};
int dy[] = {0, 0, 1, -1};
queue<pii> q;
struct node {
   int x, y, msk;
   node(int x = 0, int y = 0, int msk = 0):x(x), y(y), msk(msk){}
} now;
pair<node, node> pre[1 << M][N][N];
void spfa(int msk) {
   while (!q.empty()) {
      pii u = q.front(); q.pop();
      int x = u.fi, y = u.se;
      vis[msk][x][y] = 0;
      rep(i, 0, 4) {
         int nx = x + dx[i], ny = y + dy[i], t = msk | st[nx][ny];
         if (nx > n || nx < 1 || ny > m || ny < 1) continue;
         int &z = dp[t][nx][ny], w = dp[msk][x][y] + a[nx][ny];
         if (z > w) {
            z = w, pre[t][nx][ny] = mp(node(x, y, msk), node(x, y, 0));
            if (t == msk && !vis[msk][nx][ny]) {
               vis[msk][nx][ny] = 1;
               q.push(mp(nx, ny));
            }
         }
      }
   }
}
void dfs(node now) {
   pair<node, node> t = pre[now.msk][now.x][now.y];
   node t1 = t.fi, t2 = t.se;
   use[now.x][now.y] = 1;
   if (!t1.x) return;
   dfs(t1);
   if (t2.msk) dfs(t2);
}
int SteinerTree(int n, int m) {
   memset(dp, 0x3f, sizeof(dp));
   rep(i, 1, n+1) rep(j, 1, m+1) {
      cin >> a[i][j];
      if (!a[i][j]) {
         st[i][j] = pw(k);
         dp[pw(k)][i][j] = 0;
         pre[pw(k++)][i][j] = mp(node(0, 0, 0), node(0, 0, 0));
      }
   }
   S = pw(k) - 1;
   rep(msk, 1, S+1) {
      rep(i, 1, n+1)
         rep(j, 1, m+1) {
            if (st[i][j] && !(st[i][j] & msk)) continue;
            int &z = dp[msk][i][j];
            for (int t = msk & (msk - 1); t > 0; t = (t - 1) & msk) {
               int t1 = t | st[i][j], t2 = msk ^ t | st[i][j];
               int w = dp[t1][i][j] + dp[t2][i][j] - a[i][j];
               if (z > w) z = w, pre[msk][i][j] = mp(node(i, j, t1), node(i, j, t2));
            }
            if (z < inf) q.push(mp(i, j)), vis[msk][i][j] = 1;
         }
      spfa(msk);
   }
   ans = inf;
   rep(i, 1, n+1) rep(j, 1, m+1) if (ans > dp[S][i][j])
      ans = dp[S][i][j], now = node(i, j, S);
   dfs(now);
   return ans == inf ? -1 : ans;
}
```

6.22 StoerWagner_O(n3)

```
struct StoerWagner{
   static const int N = 305, INF = 0x3f3f3f3f;
   int n, g[N][N], val[N];
   bool vis[N], use[N];
   void init(int _n) {
      n = _n;
      fill_n(use + 1, n, 0);
      rep(i, 1, n+1) fill_n(g[i] + 1, n, 0);
   }
   void add_edge(int u, int v, int w) { g[u][v] += w; g[v][u] += w; }
   void merge(int u, int v) {
      rep(i, 1, n+1) {
         g[v][i] += g[u][i];
         g[i][v] += g[i][u];
      }
      use[u] = 1;
   }
   int MinimumCutPhase(int cnt, int &s, int &t) {
      fill_n(val + 1, n, 0);
      fill_n(vis + 1, n, 0);
      t = 1;
      while (--cnt) {
         vis[s = t] = 1;
         rep(i, 1, n+1) if (!vis[i] && !use[i]) val[i] += g[t][i];
         int ma = 0;
         rep(i, 1, n+1) if (!vis[i] && !use[i] && val[i] >= ma) ma = val[i], t = i;
         if (!ma) return 0;
      }
      return val[t];
   }
   int solve() {
      int res = INF;
      for (int i = n, s, t; i > 1; --i) {
         res = min(res, MinimumCutPhase(i, s, t));
         if (res == 0) break;
         merge(s, t);
      }
      return res;
   }
} SW;
```

6.23 StoerWagner_O(nmlog(m))

```
struct StoerWagner{
   static const int N = 3005, M = 100005 * 2, INF = 0x3f3f3f3f;
   int head[N], val[N], e, n, to[M], ne[M], data[M], fa[N], link[N];
   bool vis[N];
   void init(int _n) {
      n = _n;
      fill_n(head + 1, n, -1);
      fill_n(link + 1, n, -1);
      rep(i, 1, n+1) fa[i] = i;
      e = 0;
   }
   void add_edge(int u, int v, int w) {
      to[e] = v; data[e] = w; ne[e] = head[u]; head[u] = e++;
      to[e] = u; data[e] = w; ne[e] = head[v]; head[v] = e++;
   }
   int findset(int u) { return u == fa[u] ? u : fa[u] = findset(fa[u]); }
   void merge(int u, int v) {
      int p = u;
      while (~link[p]) p = link[p];
      link[p] = v;
      fa[v] = u;
   }
   int MinimumCutPhase(int cnt, int &s, int &t) {
      fill_n(val + 1, n, 0);
      fill_n(vis + 1, n, 0);
      priority_queue<pii> q;
      t = 1;
      while (--cnt) {
         vis[s = t] = 1;
         for (int u = s; ~u; u = link[u]) {
            for (int p = head[u]; ~p; p = ne[p]) {
               int v = findset(to[p]);
               if (!vis[v]) q.push(mp(val[v] += data[p], v));
            }
         }
         while (!q.empty() && (vis[q.top().se] || val[q.top().se] != q.top().fi)) q.pop
         ();
         if (q.empty()) return 0;
         t = q.top().se; q.pop();
      }
      return val[t];
   }
   int solve() {
      int res = INF;
      for (int i = n, s, t; i > 1; --i) {
         res = min(res, MinimumCutPhase(i, s, t));
         if (res == 0) break;
         merge(s, t);
      }
      return res;
   }
} SW;
```

6.24 ZKW

```
// [0,n) , init!! , inf modify
template<class U,class V>
struct ZKW{
    static const int N = 1010 , M = 40404;
    int h[N] , ing[N] , v[N] , to[M] , ne[M] , e , s , t , n;
    U cap[M];V dis[N] , cost[M];
    void ini(int _n = N){ fill(h , h + (n=_n) , -1);e = 0;}
    void liu(int u,int v,U c,V w){ to[e] = v;ne[e] = h[u];cap[e] = c;cost[e] = w;h[u] =
    e++;}
    void link(int u,int v,U c,V w){ liu(u,v,c,w);liu(v,u,0,-w); }
    void spfa(){
        queue<int> Q;
        fill(dis,dis+n,inf);
        ing[t] = true , dis[t] = 0;
        Q.push(t);
        while(!Q.empty()){
            int c = Q.front();Q.pop();ing[c] = false;
            for(int k=h[c];~k;k=ne[k]){
                int v = to[k];
                if(cap[k^1] <= 0) continue;
                if(dis[c] + cost[k^1] < dis[v]){
                    dis[v] = dis[c] + cost[k^1];
                    if(!ing[v]) Q.push(v) , ing[v] = true;
                }
            }
        }
    }
    U flow;V mincost;
    bool modlable(){
        V Min = inf;
        rep(c,0,n) if(v[c]) for(int k=h[c];~k;k=ne[k]){
         int t=to[k];
         if(!v[t] && cap[k] > 0) Min = min(Min , dis[t] + cost[k] - dis[c]);
      }
        if(Min == inf) return false;
        rep(i,0,n) if(v[i]) dis[i] += Min;
        return true;
    }
    U dfs(int c,U mx){
        if(c == t) return flow += mx , mincost += mx * dis[s] , mx;
        v[c] = true;U ret = 0;
        for(int k=h[c];~k;k=ne[k]){
            int t = to[k];
            if(!v[t] && cap[k] > 0 && dis[c] - cost[k] == dis[t]){
                U tmp = dfs(t , min(cap[k] , mx - ret));
                cap[k] -= tmp , cap[k^1] += tmp;
                ret += tmp;
                if(ret == mx) return ret;
            }
        }
        return ret;
    }
    pair<U,V> run(int _s,int _t){
        s = _s , t = _t;
        spfa();
        flow = mincost = 0;
        do do memset(v,0,sizeof(v[0])*n);
            while(dfs(s,inf));
        while(modlable());
        return make_pair(flow , mincost);
    }
};
```

6.25 k 短路

```
// S -> T 可重复经过点的第 K 短路
// time : O(klogk + mlogn) space : O(nlogn)
const int N = 5050, M = 200005, B = 20;
const db eps = 1e-9, inf = 1e16;
db dis[N], w;
bool vis[N], tree[M];
int n, m, S, T, fa[N], st[N], top, u, v;
struct Graph{
   int h[N], ne[M], to[M], e;
   db w[M];
   inline void add(int u, int v, db val){
      ne[++e] = h[u], h[u] = e, to[e] = v, w[e] = val;
   }
} g, rg;
void Dij(){
   rep(i, 1, n+1) dis[i] = inf;
   priority_queue<pair<db, int> > pq;
   pq.push(mp(dis[T] = 0, T));
   while(!pq.empty()){
      int u = pq.top().se; pq.pop();
      if (vis[u]) continue; vis[u] = 1;
      for (int i = rg.h[u]; i; i = rg.ne[i]) {
         int v = rg.to[i];
         if (dis[v] > dis[u] + rg.w[i] + eps) {
            dis[v] = dis[u] + rg.w[i];
            pq.push(mp(-dis[v], v));
         }
      }
   }
}
void dfs(int u){
   st[++top] = u; vis[u] = 1;
   for (int i = rg.h[u]; i; i = rg.ne[i]) {
      int v = rg.to[i];
      if (!vis[v] && fabs(dis[v] - dis[u] - rg.w[i]) <= eps) {
         fa[v] = u; tree[i] = 1;
         dfs(v);
      }
   }
}
int rt[N];
namespace LT{
   int ls[M*B], rs[M*B], ht[M*B], id[M*B], tot;
   db val[M*B];
   inline int newnode(db _val, int _id, int _dis = 0){
      int p = ++tot;
      val[p] = _val, id[p]=_id, ht[p] = _dis;
      ls[p] = rs[p] = 0;
      return p;
   }
   inline int _copy(int ori){
      int p = ++tot;
      val[p] = val[ori], id[p] = id[ori], ht[p] = ht[ori];
      ls[p] = ls[ori], rs[p] = rs[ori];
      return p;
   }
   inline int merge(int a, int b){
      if(!a || !b) return a|b;
      if(val[a] > val[b]) swap(a, b);
      int now = _copy(a);
      rs[now] = merge(rs[now], b);
      if(ht[ls[now]] < ht[rs[now]]) swap(ls[now],rs[now]);
      ht[now] = ht[rs[now]] + 1;
      return now;
   }
   inline void ins(int &rt, db val, int id){ rt = merge(rt, newnode(val, id)); }
}
void build_heap(){
   rep(j, 1, top+1) {
      int u = st[j];
      rt[u] = rt[fa[u]];
      for (int i = g.h[u]; i ; i = g.ne[i]) {
            int v = g.to[i];
            if (!tree[i] && dis[v] < inf) LT::ins(rt[u], dis[v] - dis[u] + g.w[i], v);
        }
    }
}
typedef pair<db, int> pdi;
db E;
inline int calc_K(){
   int ans = 1; E -= dis[S];
   priority_queue<pdi, vector<pdi>, greater<pdi> > pq;
   if (rt[S]) pq.push(mp(dis[S] + LT::val[rt[S]], rt[S]));
   while(!pq.empty()) {
      pdi t = pq.top(); pq.pop();
      db w = t.fi; int u = t.se, o = LT::id[u];
      E -= w; if (E >= 0) ++ans; else return ans;
      int ls = LT::ls[u], rs = LT::rs[u];
      if (rt[o]) pq.push(mp(w + LT::val[rt[o]], rt[o]));
      if (ls) pq.push(mp(w + LT::val[ls] - LT::val[u], ls));
      if (rs) pq.push(mp(w + LT::val[rs] - LT::val[u], rs));
   }
   return ans;
}
int main(){
   ios::sync_with_stdio(0);
   cin.tie(0);
   cin >> n >> m >> E;
   S = 1; T = n;
   rep(i, 1, m+1) {
      cin >> u >> v >> w;
      g.add(u, v, w);
      rg.add(v, u, w);
   }
   Dij();
   rep(i, 1, n+1) vis[i] = 0;
   dfs(T);
   build_heap();
   cout << calc_K() << endl;
   return 0;
}
```

6.26 仙人掌最短路

```
/*
 * 建出圆方树，选任意圆点作为根，环的根指的是环上深度最小的点。
 * 圆圆边边权不变，圆方边边权是圆点到它所在环的根的最短距离。
 * 如果询问两点的 lca 是圆点， ans = dep[a] + dep[b] - dep[lca]
 * 如果是方点， ans = dep[a] + dep[b] - dep[A] - dep[B] + dis(A, B)
 */
```

6.27 前向星

```
struct Gra {
   static const int N = ::N << 1;
   int L, hd[::N], ne[N], to[N]; ll val[N];
   inline void init(int n) { L = 0; rep(i, 1, n + 1) hd[i] = -1; }
   inline void _add(int u, int v, ll w) { to[L] = v; val[L] = w; ne[L] = hd[u]; hd[u] =
L++; }
   inline void add(int u, int v, ll w) { _add(u, v, w); _add(v, u, w); }
};
```

6.28 带花树

```
// time : O(n^3)
// id : 0 .. n-1
struct blossom {
   static const int N = 5005;
   vi g[N];
   int u, v, n, match[N], q[N], L, R, pred[N], b[N], s, t, newb;
   bool inq[N], inb[N], inp[N];
   void init(int _n) { n = _n; rep(i, 0, n) g[i].clear(); }
   void link(int u, int v) { g[u].pb(v); g[v].pb(u); }
   void push(int u) { q[R++] = u; inq[u] = 1; }
   int pop() { return q[L++]; }
   int LCA(int u,int v) {
      rep(i, 0, n) inp[i]=0;
      while(1) {
         inp[u = b[u]] = 1;
         if (u == s) break;
         u = pred[match[u]];
      }
      while(1) {
         if (inp[v = b[v]]) break;
         v = pred[match[v]];
      }
      return v;
   }
   void ResetTrace(int u) {
      int v;
      while(b[u] != newb) {
         v = match[u];
         inb[b[u]] = inb[b[v]] = 1;
         u = pred[v];
         if(b[u] != newb) pred[u] = v;
      }
   }
   void Blossom(int u,int v) {
      newb = LCA(u,v);
      rep(i, 0, n) inb[i] = 0;
      ResetTrace(u);
      ResetTrace(v);
      if(b[u] != newb) pred[u] = v;
      if(b[v] != newb) pred[v] = u;
      rep(i, 0, n) if (inb[b[i]]) {
         b[i] = newb;
         if (!inq[i]) push(i);
      }
   }
   bool Find(int u) {
      bool found = 0;
      rep(i, 0, n) pred[i] = –1, b[i] = i, inq[i] = 0;
      s = u, t = –1, L = R = 0;
      push(s);
      while(L < R) {
         int u = pop();
         per(i, 0, sz(g[u])) {
            int v = g[u][i];
            if (b[u] != b[v] && match[u] != v)
               if(v == s || (match[v] >= 0 && pred[match[v]] >= 0))
                  Blossom(u, v);
               else if(pred[v] == –1) {
                  pred[v]=u;
                  if (match[v] >= 0) push(match[v]);
                  else return t = v, 1;
               }
         }
      }
      return found;
   }
   void AugmentPath() {
      int u = t, v, w;
      while(u >= 0) {
         v = pred[u], w = match[v];
         match[v] = u, match[u] = v;
         u = w;
      }
   }
   int solve() {
      int res = 0;
      rep(i, 0, n) match[i] = –1;
      // random_shuffle maybe faster
      rep(i, 0, n) if (match[i] == –1) if (Find(i)) AugmentPath();
      rep(i, 0, n) if (match[i] != –1) res++;
      return res / 2;
   }
} G;
```

6.29 最短路矩阵中第 k 小

```
const int N = 2e5 + 7;
vector<pii> g[N]; // ( 边权 , 终点 ) 需要排序
int n, m, k, u, v, w;
struct data { // 距离起点当前点当前扩展过的边编号
   ll w; int st, last, id;
   data(ll W, int S, int L, int I) { w = W; st = S; last = L; id = I; }
   bool operator < (const data &c) const { return w > c.w; }
};
// 连通图的话 k <= n * (n – 1)
// 复杂度最坏应该是 O( min(nmlogn, k^2logk) ) 正常应该是 O(klogk + nlogn)
ll solve(int n, vector<pii> g[], int k) {
   priority_queue<data> pq;
   set<pii> vis;
   rep(i, 1, n+1) {
      if (sz(g[i])) pq.push(data(g[i][0].fi, i, i, 0));
      vis.insert(mp(i, i));
   }
   while (!pq.empty()) {
      data u = pq.top(); pq.pop();
      int v = g[u.last][u.id].se;
      if (!vis.count(mp(u.st, v))) {
         vis.insert(mp(u.st, v));
         k—; if (k == 0) return u.w;
         if (sz(g[v])) pq.push(data(u.w + g[v][0].fi, u.st, v, 0));
      }
      if (u.id + 1 < sz(g[u.last]))
         pq.push(data(u.w – g[u.last][u.id].fi + g[u.last][u.id + 1].fi, u.st, u.last, u
         .id + 1));
   }
}
```

6.30 生成树计数与欧拉回路方案数

```
// d[][]:
//     i!=j d[i][j]=0
//     i==j d[i][j]=in_deg(i)
// b[][]:
//     from i to j has b[i][j] directed edges
// a[][] = d[][] – b[][]

// 无向图生成树个数： a[][] 任何一个 n–1 阶主子式的绝对值
// 有向图以 i 为根的生成树个数： a[][] 去掉第 i 行第 i 列的行列式的绝对值

int det(int n) { // det(a[1..n–1][1..n–1])
   int ans=1;
   rep(i, 1, n) {
      rep(j, i+1, n) while(a[j][i]) {
         int t = a[i][i] / a[j][i];
         rep(k, i, n) a[i][k] = sub(a[i][k], mul(a[j][k], t)), swap(a[i][k], a[j][k]);
         ans = P – ans;
      }
      if(a[i][i] == 0) return 0;
      ans = mul(ans, a[i][i]);
   }
   return ans;
}

// 有向图要记得判断每个点的出度入度是否相等
// 无向图需要转换成有向图
// tw(G): 以 w 为根的生成树个数
// ec(G) = tw(G) * pi((deg[v] – 1)!)
// ans = ec(G) * deg[w]; 如果求的不是本质不同的，就还需要这个
// 本质相同：    1231341 1341231
// 本质不同：    1231341 1312341
```

6.31 稳定婚姻匹配

```
int mat1[N], mat[N], pos[N];
vi g1[N], g2[N];
queue<int> q;
```

```
void match(int n, vi *g, vi *rank) {
   rep(i, 1, n+1) q.push(i), pos[i] = 0, mat[i] = 0;
   while (!q.empty()) {
      int u = q.front(); q.pop();
      int &p = pos[u], v = g[u][p];
      if (!mat[v]) mat[v] = u;
      else if (rank[v][mat[v]] > rank[v][u]) {
         q.push(mat[v]);
         mat[v] = u;
      }else q.push(u);
      p++;
   }
   rep(i, 1, n+1) mat1[mat[i]] = i;
}
```

7 Math

7.1 BerlekampMassey

```
// O(len^2)
// s_{m} = \sum_{j=0}^{m-1}s_{j}*c_{j} 系数直接适配线性递推
vi BM(vi s) {
   vi C(1, 1), B(1, 1);
   int L = 0, m = 1, b = 1;
   rep(n, 0, sz(s)) {
      ll d = 0;
      rep(i, 0, L+1) (d += 1ll * C[i] * s[n-i]) %= P;
      if(d == 0) ++m;
      else {
         vi T = C;
         ll c = P - d * kpow(b, P - 2) % P;
         while(sz(C) < sz(B) + m) C.pb(0);
         rep(i, 0, sz(B)) C[i + m] = add(C[i + m], mul(c, B[i]));
         if(2 * L <= n) L = n + 1 - L, B = T, b = d, m = 1;
         else ++m;
      }
   }
   reverse(all(C));
   rep(i, 0, sz(C)) C[i] = P - C[i];
   return vi(C.begin(), C.end() - 1);
}
```

7.2 Bernoulli

```
// desc : 0^k + 1^k + 2^k + .. + (n-1)^k
// time-ini : O(n^2)
// time-cal : k + log
namespace Bernoulli {
   const int N = 1000;
   int C[N][N], B[N];
   void ini() {
      rep(i, 0, N) C[i][0] = 1;
      rep(i, 0, N) rep(j, 1, i + 1) C[i][j] = add(C[i - 1][j - 1], C[i - 1][j]);
      B[0] = 1;
      rep(i, 1, N) {
         B[i] = 0;
         rep(j, 0, i) B[i] = add(B[i], MOD - mul(C[i + 1][j], B[j]));
         B[i] = mul(B[i], qpow(C[i + 1][i], MOD - 2)) % MOD;
      }
   }
   int cal(int n, int k) {
      int sum = 0;
      rep(i, 0, k + 1) sum = add(sum, mul(C[k + 1][i], mul(B[i], qpow(n, k + 1 - i))));
      return mul(sum, qpow(k + 1, MOD - 2));
   }
};
```

7.3 CRT

```
const int N = 1e5+7;
ll a[N], mod[N];

struct CRT{
   ll M, R;
   void exgcd(ll a, ll b, ll &x, ll &y){
      if (!b) { x = 1; y = 0; return;}
      exgcd(b, a % b, y, x);
      y -= a / b * x;
   }
   ll Inv(ll a, ll mod){
      ll x = 0, y = 0;
      exgcd(a, mod, x, y);
      x %= mod;
      return x < 0 ? x + mod : x;
   }
   ll solve(int n, ll *a, ll *mod){
      M = mod[1], R = a[1];
      rep(i, 2, n+1) {
         ll g = __gcd(M, mod[i]);
         ll inv = Inv(M / g, mod[i] / g);
         if ((a[i] - R) % g) return -1; // 无解
         R += inv * ((a[i] - R) / g) % (mod[i] / g) * M;
         M = M / g * mod[i];
         R = (R % M + M) % M; // 可能为 0 看是否需要是正整数
      }
      return R;
   }
} crt;
```

7.4 EulerPower

```
// a[l] ^ a[l+1] ^ a[l+2] ... ^ a[r] % mod 注意结果要再模 mod
map<int, int> M;
int phi(int n) {
   if (M.count(n)) return M[n];
   int r = n, nn = n;
```

```
    for(int i = 2; i * i <= n; i++) if (n % i == 0){
        r = r / i * (i-1);
        while (n % i == 0) n /= i;
    }
    if (n > 1) r = r / n * (n-1);
    M[nn] = r;
    return r;
}
ll Euler_qpow(ll a, ll b, ll mod) {
    ll res = 1; bool ok = (b > 0 && a >= mod);
    while (b) {
        if (b & 1) {
            res = res * a;
            ok |= (res >= mod);
            res %= mod;
        }
        a = a * a;
        ok |= (b > 1 && a >= mod);
        a %= mod;
        b >>= 1;
    }
    return res + mod * ok;
}
ll work(int l, int r, int mod) {
    if (mod == 1) return 1;
    if (l == r) return a[l];
    return Euler_qpow(a[l], work(l+1, r, phi(mod)), mod);
}
```

7.5 FFT

```
const int M = 1 << 17 << 1;
const db pi = acos(-1);

struct vir{
    db r, i;
    vir(db r = 0.0, db i = 0.0) : r(r), i(i){}
    void print() {printf("%f %f\n", r, i);}
    vir operator +(const vir &c) {return vir(r + c.r, i + c.i);}
    vir operator -(const vir &c) {return vir(r - c.r, i - c.i);}
    vir operator *(const vir &c) {return vir(r * c.r - i * c.i, r * c.i + i * c.r);}
} a[M], b[M], w[2][M];

struct FFT{
    int N, na, nb, rev[M];
    void fft(vir *a, int f){
        vir x, y;
        rep(i, 0, N) if (i < rev[i]) swap(a[i], a[rev[i]]);
        for (int i = 1; i < N; i <<= 1)
            for (int j = 0, t = N/(i<<1); j < N; j += i<<1)
                for (int k = 0, l = 0; k < i; k++, l += t)
                    x = w[f][l] * a[j+k+i], y = a[j+k], a[j+k] = y+x, a[j+k+i] = y-x;
        if (f) rep(i, 0, N) a[i].r /= N;
    }
    void work(){
        int d = __builtin_ctz(N);
        rep(i, 0, N) {
            rev[i] = (rev[i>>1] >> 1) | ((i&1) << (d-1));
            w[1][i] = w[0][i] = vir(cos(2*pi*i/N), sin(2*pi*i/N));
            w[1][i].i = -w[1][i].i;
        }
    }
    void doit(vir *a, vir *b, int na, int nb){ // [0, na)
        for (N = 1; N < na + nb - 1; N <<= 1);
        rep(i, na, N) a[i] = vir(0, 0);
        rep(i, nb, N) b[i] = vir(0, 0);
        work(), fft(a, 0), fft(b, 0);
        rep(i, 0, N) a[i] = a[i] * b[i];
        fft(a, 1);
        //rep(i, 0, N) a[i].print();
    }
} fft;
```

7.6 FFTMOD

```
const db PI = acos(-1);
const int M = 1 << 18 << 1;
int na, nb, a[M], b[M];
struct vir{
    db r, i;
    vir(db r = 0.0, db i = 0.0) : r(r), i(i){}
    inline vir operator +(const vir &c) {return vir(r + c.r, i + c.i);}
    inline vir operator -(const vir &c) {return vir(r - c.r, i - c.i);}
    inline vir operator *(const vir &c) {return vir(r * c.r - i * c.i, r * c.i + i * c.r)
    ;}
    inline vir operator !() const {return vir(r, -i);}
    void print() {printf("%lf %lf\n", r, i);}
};
struct FFTMOD{
    static const int M = 1 << 18 << 1;
    int N, L, MASK;
    vir w[M], A[M], B[M], C[M], D[M];
    void FFT(vir p[], int n) {
        for (int i = 1, j = 0; i < n - 1; ++i) {
            for (int s = n; j ^= s >>= 1, ~j & s;);
            if (i < j) swap(p[i], p[j]);
        }
        for (int d = 0; (1 << d) < n; ++d) {
            int m = 1 << d, m2 = m * 2, rm = n >> (d + 1);
            for (int i = 0; i < n; i += m2) {
                for (int j = 0; j < m; ++j) {
                    vir &p1 = p[i + j + m], &p2 = p[i + j];
                    vir t = w[rm * j] * p1;
                    p1 = p2 - t, p2 = p2 + t;
                }
            }
        }
    }
    void doit(int *a, int *b, int na, int nb){
        for (N = 1; N < na + nb - 1; N <<= 1);
    rep(i, 0, na) a[i] = (a[i] % P + P) % P; rep(i, na, N) a[i] = 0;
    rep(i, 0, nb) b[i] = (b[i] % P + P) % P; rep(i, nb, N) b[i] = 0;
    L = 15;  MASK = (1<<L) - 1;
    rep(i, 0, N) w[i] = vir(cos(2 * i * PI / N), sin(2 * i * PI / N));
    rep(i, 0, N) {
      A[i] = vir(a[i] >> L, a[i] & MASK);
      B[i] = vir(b[i] >> L, b[i] & MASK);
    }
    mul(a);
  }
  void mul(int *a) {
    FFT(A, N), FFT(B, N);
    rep(i, 0, N) {
      int j = (N - i) % N;
      vir da = (A[i] - !A[j]) * vir(0, -0.5),
          db = (A[i] + !A[j]) * vir(0.5, 0),
          dc = (B[i] - !B[j]) * vir(0, -0.5),
          dd = (B[i] + !B[j]) * vir(0.5, 0);
      C[j] = da * dd + da * dc * vir(0, 1);
      D[j] = db * dd + db * dc * vir(0, 1);
    }
    FFT(C, N), FFT(D, N);
    rep(i, 0, N) {
      ll  da = (ll)(C[i].i / N + 0.5) % P,
          db = (ll)(C[i].r / N + 0.5) % P,
          dc = (ll)(D[i].i / N + 0.5) % P,
          dd = (ll)(D[i].r / N + 0.5) % P;
      a[i] = ((dd << (L * 2)) + ((db + dc) << L) + da) % P;
    }
  }
} fft;
```

7.7 FFT_fast

```
const int N = 1 << 21;
const double pi=acos(-1.0);
struct vir{
  double a,b;
  vir(double r=0.0,double i=0.0) {a=r,b=i;}
  vir operator +(const vir &o) const{return vir(a+o.a,b+o.b);}
  vir operator -(const vir &o) const{return vir(a-o.a,b-o.b);}
  vir operator *(const vir &o) const{return vir(a*o.a-b*o.b,b*o.a+a*o.b);}
  vir operator *(const double &o) const{return vir(a*o,b*o);}
  vir operator !() const{return vir(a,-b);}
} x[N|1], y[N|1], z[N|1], w[N|1];
int K;
void fft(vir x[],int k,int v){
  for(int i=0,j=0; i<k; i++){
    if(i>j)swap(x[i],x[j]);
    for(int l=k>>1; (j^=l)<l; l>>=1);
  }
  w[0] = vir(1, 0);
  for(int i=2; i<=k; i<<=1){
    vir g = vir(cos(2*pi/i), (v ? -1 : 1) * sin(2*pi/i));
    for(int j=(i>>1); j>=0; j-=2) w[j] = w[j>>1];
    for(int j=1; j<i>>1; j+=2) w[j] = w[j-1] * g;
    for(int j=0; j<k; j+=i){
      vir *a = x+j, *b = a+(i>>1);
      for(int l=0; l<i>>1; l++){
        vir o = b[l] * w[l];
        b[l] = a[l] - o;
        a[l] = a[l] + o;
      }
    }
  }
  if (v) for(int i=0; i<k; i++) x[i] = vir(x[i].a/k,x[i].b/k);
}
void doit(int *a, int *b, int na, int nb) {
  for(K = 1; K <= na+nb>>1; K <<= 1);
  rep(i, 0, K) x[i] = y[i] = vir(0, 0);
  for(int i=0; i<=na; i++) (i&1 ? x[i>>1].b : x[i>>1].a) = a[i];
  for(int i=0; i<=nb; i++) (i&1 ? y[i>>1].b : y[i>>1].a) = b[i];
  fft(x, K, 0); fft(y, K, 0);
  rep(i, 0, K){
    int j = K-1 & K-i;
    vir tmp = (i&K>>1) ? vir(1, 0) - w[i^K>>1] : w[i] + vir(1, 0);
    z[i] =(x[i]*y[i]*4 - (x[i] - !x[j])*(y[i] - !y[j])*tmp)*0.25;
  }
  fft(z, K, 1);
  rep(i, 0, na+nb+1) a[i] = i&1 ? z[i>>1].b + 0.1 : z[i>>1].a + 0.1;
}
```

7.8 FWT_k 进制_xor 版本

```
const int _p = 998244353;
ll add(ll x, ll y) { x += y; return x%_p; }
ll mul(ll x, ll y) { return x*y%_p; }
ll Pow(ll x, ll k) {
  ll ret = 1;
  for (; k; k >>= 1, x = mul(x, x)) if (k & 1) ret = mul(ret, x);
  return ret;
}

template <int K>
struct Num {
  ll a[K];
  Num(int x = 0) { mem(a, 0), a[0] = x; }
  inline Num& operator = (const Num &t) {
    rep(i, 0, K) a[i] = t.a[i];
    return *this;
  }
  inline Num& operator = (int x) { mem(a, 0), a[0] = x; return *this; }
  inline friend Num operator + (const Num &a, const Num &b) {
    Num c;
    rep(i, 0, K) c.a[i] = add(a.a[i], b.a[i]);
    return c;
  }
  inline friend Num operator - (const Num &a, const Num &b) {
    Num c;
    rep(i, 0, K) c.a[i] = add(a.a[i], -b.a[i]);
```

```
      return c;
   }
   inline friend Num operator * (const Num &a, const Num &b) {
      Num c;
      rep(i, 0, K) rep(j, 0, K) (c.a[(i + j) % K] += mul(a.a[i], b.a[j])) %= _p;
      return c;
   }
   inline friend Num operator >> (const Num &a, int k) {
      Num c;
      rep(i, 0, K) c.a[(i + k) % K] = a.a[i];
      return c;
   }
   inline friend Num operator ^ (Num x, ll k) {
      Num ret = 1;
      for (; k; k >>= 1, x = x*x) if (k & 1) ret = ret*x;
      return ret;
   }
   inline ll Value() {
      int cnt = K&-K, L = K / cnt; ll ret = add(a[0], -(L > 1)*a[cnt]);
      if (K & 1 ^ 1) ret -= add(a[K >> 1], -(L > 1)*a[(K >> 1) + cnt]), ret %= _p;
      return ret;
   }
   inline void print(string s = "") {
      printf("\n\n\n%s\n", s.c_str());
      rep(i, 0, K) printf("a[%d] => %d\n", i, a[i]);
   }
};

template <int M, int N, int K>
struct FT {
   Num<K> tmp[M << 1], a[N], b[N]; int t;
   void FWT(Num<K> a[], int S, int n, int op) {
      if (n == 1) return; int L = n / M;
      rep(i, 0, M) FWT(a, S + L*i, n / M, op);
      rep(i, 0, L) {
         rep(j, 0, M) tmp[j] = 0;
         rep(j, 0, M) rep(k, 0, M) {
            t = op*j*k%M, t < 0 ? t += M : 0;
            tmp[j] = tmp[j] + (a[S + L*k + i] >> t);
         }
         rep(j, 0, M) a[S + L*j + i] = tmp[j];
      }
   }
   void Multiply(ll A[], ll B[], int n, ll C[]) {
      rep(i, 0, n) a[i] = A[i], b[i] = B[i];
      FWT(a, 0, n, 1), FWT(b, 0, n, 1);
      rep(i, 0, n) a[i] = a[i] * b[i];
      FWT(a, 0, n, -1); ll inv = Pow(n, _p - 2);
      rep(i, 0, n) C[i] = mul(a[i].Value(), inv), C[i] < 0 ? C[i] += _p : 0;
   }
   int get(int x, int y) {
      int ret = 0, B = 1;
      for (; x || y; x /= M, y /= M, B *= M) ret += (x%M + y%M) % M*B;
      return ret;
   }
```

```
   void Multiply_B(ll A[], ll B[], int n, ll C[]) {
      rep(i, 0, n) rep(j, 0, n) (C[get(i, j)] += mul(A[i], B[j])) %= _p;
      rep(i, 0, n) C[i] < 0 ? C[i] += _p : 0;
   }
};
```

7.9 FWT_k 进制_xor 版本_模域

```
const int _p = (int)1e9 + 9, w0 = 115381398ll;
ll add(ll x, ll y) { x += y; return x%_p; }
ll mul(ll x, ll y) { return x*y%_p; }
ll Pow(ll x, ll k) {
   ll ret = 1;
   for (; k; k >>= 1, x = mul(x, x)) if (k & 1) ret = mul(ret, x);
   return ret;
}
template <int N, int K>
struct FT {
   ll tmp[K << 1], a[N], b[N], w[K]; int t;
   void Init(ll w0) { w[0] = 1; rep(i, 1, K) w[i] = mul(w[i - 1], w0); }
   void FWT(ll a[], int S, int n, int op) {
      if (n == 1) return; int L = n / K;
      rep(i, 0, K) FWT(a, S + L*i, n / K, op);
      rep(i, 0, L) {
         rep(j, 0, K) tmp[j] = 0;
         rep(j, 0, K) rep(k, 0, K) {
            t = op*j*k%K, t < 0 ? t += K : 0;
            tmp[j] = add(tmp[j], mul(a[S + L*k + i], w[t]));
         }
         rep(j, 0, K) a[S + L*j + i] = tmp[j];
      }
   }
   void Multiply(ll A[], ll B[], int n, ll C[]) {
      rep(i, 0, n) a[i] = A[i], b[i] = B[i];
      FWT(a, 0, n, 1), FWT(b, 0, n, 1);
      rep(i, 0, n) a[i] = mul(a[i], b[i]);
      FWT(a, 0, n, -1); ll inv = Pow(n, _p - 2);
      rep(i, 0, n) C[i] = mul(a[i], inv), C[i] < 0 ? C[i] += _p : 0;
   }
   int get(int x, int y) {
      int ret = 0, B = 1;
      for (; x || y; x /= K, y /= K, B *= K) ret += (x%K + y%K) % K*B;
      return ret;
   }
   void Multiply_B(ll A[], ll B[], int n, ll C[]) {
      rep(i, 0, n) rep(j, 0, n) (C[get(i, j)] += mul(A[i], B[j])) %= _p;
      rep(i, 0, n) C[i] < 0 ? C[i] += _p : 0;
   }
};
// w0 表示单位根模域表示，默认是  3 进制的，进制要整除模数  -1
/*
1000000009
3 115381398
4 430477711
6 115381399
```

```
7 95932470
8 118835338
9 246325263
12 86475609
14 9196980
18 4138593
21 32705801
24 304035978

998244353
4 86583718
7 14553391
8 372528824
14 467509451
*/
```

7.10 Fib

```
// sum(fib[1..n]) + 1=fib[n + 2]
// gcd(fib[n], fib[m]) = fib[gcd(n, m)]
```

7.11 Fraction

```
template<class T>
struct Fra{
   T a, b;
   Fra() : a(0), b(1) {}
   Fra(string s) {
      stringstream ss(s); char c;
      ss >> a >> c >> b;
      *this = Fra(a, b);
   }
   Fra(T c) : a(c), b(1) {}
   Fra(T _a, T _b) {
      T d = __gcd(_a, _b);
      a = _a / d, b = _b / d;
      if(b < 0) a = -a, b = -b;
   }
   Fra operator + (const Fra &c) const { return Fra(a * c.b + b * c.a, b * c.b); }
   Fra operator - (const Fra &c) const { return Fra(a * c.b - b * c.a, b * c.b); }
   Fra operator * (const Fra &c) const { return Fra(a * c.a, b * c.b); }
   Fra operator / (const Fra &c) const { return Fra(a * c.b, b * c.a); }
   Fra operator * (const T &c) const { return Fra(a * c, b); }
   Fra operator / (const T &c) const { return Fra(a, b * c); }
   bool operator == (const Fra &c) const { return a == c.a && b == c.b; }
   bool operator != (const Fra &c) const { return !(*this == c); }
   void print() { cout << a << "/" << b; }
};
typedef Fra<ll> fll;
```

7.12 GaussDB

```
namespace GaussDB{
   static const int N = 505;
   db a[N][N], x[N]; //增广矩阵和解集
   int free[N], fnum, k, col, p; // 一组合法自由变元
   const db eps = 1e-14;

   void genx(int var) {
      int pre = var; fnum = 0;
      per(i, 0, k) {
         rep(j, 0, var) if (fabs(a[i][j]) > eps) { p = j; break; }
         rep(j, 0, i) if (fabs(a[j][p]) > eps) {
            db t = a[j][p] / a[i][p];
               rep(l, p, var+1) a[j][l] -= a[i][l] * t;
         }
         rep(j, p+1, pre) free[fnum++] = j, x[j] = (?); pre = p;
         x[p] = a[i][var];
         rep(j, p+1, var) x[p] -= a[i][j] * x[j];
         x[p] /= a[i][p];
      }
      rep(j, 0, pre) free[fnum++] = j;
   }

   int Gauss(int equ, int var){
      for(k = col = 0; k < equ && col < var; ++k, ++col){
         p = k; rep(i, k+1, equ) if(fabs(a[i][col]) > fabs(a[p][col])) p = i;
         if (p != k) rep(j, col, var+1) swap(a[p][j], a[k][j]);
         if(fabs(a[k][col]) < eps) {k--; continue;}
         rep(i, k+1, equ){
            if (fabs(a[i][col]) < eps) continue;
            db t = a[i][col] / a[k][col];
            rep(j, col, var+1) a[i][j] -= a[k][j] * t;
         }
      }
      rep(i, k, equ) if (fabs(a[i][var]) > eps) return -1;//无解
      if(k < var){
         // genx(var);
         return var - k;//自由变元个数
      }
      per(i, 0, var) {
         db t = a[i][var];
         rep(j, i+1, var) if (fabs(a[i][j]) > eps) t -= x[j] * a[i][j];
         x[i] = t / a[i][i];
      }
      return 0;
   }
}
```

7.13 GaussInt

```
namespace GaussInt{
   static const int N = ::N, P = 1e9 + 7;
   int a[N][N], x[N]; //增广矩阵和解集
   int free[N], fnum, k, col, p; // 一组合法自由变元
   int add(int a, int b) {if ((a += b) >= P) a -= P; return a < 0 ? a + P : a;}
   int mul(int a, int b) {return 1ll * a * b % P;}
```

```
    int kpow(int a, int b) {int r=1;for(;b;b>>=1,a=mul(a,a)) {if(b&1)r=mul(r,a);}return r
    ;}

    void genx(int var) {
        int pre = var; fnum = 0;
        per(i, 0, k) {
            rep(j, 0, var) if (a[i][j]) { p = j; break; }
            rep(j, 0, i) if (a[j][p]) {
                int t = a[j][p];
                rep(l, p, var+1) a[j][l] = add(a[j][l], -mul(a[i][l], t));
            }
            rep(j, p+1, pre) free[fnum++] = j, x[j] = (?); pre = p;
            x[p] = a[i][var];
            rep(j, p+1, var) x[p] = add(x[p], -mul(a[i][j], x[j]));
        }
        rep(j, 0, pre) free[fnum++] = j;
    }

    int Gauss(int equ, int var){
        for(k = col = 0; k < equ && col < var; ++k, ++col){
            p = k; rep(i, k, equ) if (a[i][col]) {p = i; break;}
            if (p != k) rep(j, col, var+1) swap(a[p][j], a[k][j]);
            if(!a[k][col]) {k--; continue;}
            int inv = kpow(a[k][col], P - 2);
            rep(i, col, var+1) a[k][i] = mul(a[k][i], inv);
            rep(i, k+1, equ) if (a[i][col]) {
                int t = a[i][col];
                rep(j, col, var+1) a[i][j] = add(a[i][j], -mul(a[k][j], t));
            }
        }
        rep(i, k, equ) if (a[i][var]) return -1;//无解
        if(k < var){
            //genx(var);
            return var - k;//自由变元个数
        }
        per(i, 0, var) {
            int t = a[i][var];
            rep(j, i+1, var) if (a[i][j]) t = add(t, -mul(a[i][j], x[j]));
            x[i] = t;
        }
        return 0;
    }
}
```

7.14 GaussXor

```
//对 2 取模的 01 方程组
namespace Gauss{
    static const int N = 2e3 + 10;
    //有 equ 个方程， var 个变元。增广矩阵行数为 equ 列数为, [0..var]
    bitset<N> a[N]; //增广矩阵 modif
    int x[N]; //解集
    int p, col, k; // k 为增广矩阵的秩
    int free[N], fnum; //一组合法自由变元（多解枚举自由变元可以使用）
    //返回值为 -1 表示无解，为 0 是唯一解，否则返回自由变元个数
    void genx(int msk, var) {
        rep(i, 0, fnum) x[free[i]] = (msk >> i) & 1;
        per(i, 0, k) {
            rep(j, 0, var) if(a[i][j]) { p = j; break; }
            x[p] = a[i][var];
            rep(j, p+1, var) x[p] ^= (a[i][j] && x[j]);
        }
    }

    int Gauss(int equ, int var){
        fnum = 0;
        for(k = 0, col = 0; k < equ && col < var; k++, col++){
            p = k; rep(i, k, equ) if (a[i][col]) {p = i; break;}
            if (p != k) swap(a[k], a[p]);
            if (!a[k][col]){
                k--; free[fnum++] = col;//这个是自由变元
                continue;
            }
            rep(i, 0, equ) if (i != k && a[i][col]) a[i] ^= a[k];
        }
        rep(i, col, var) free[fnum++] = i;
        rep(i, k, equ) if (a[i][var]) return -1;
        if(k < var) {
            // genx(0, var);
            return var - k;//自由变元个数
        }
        //唯一解，回代
        per(i, 0, var){
            x[i] = a[i][var];
            rep(j, i+1, var) x[i] ^= (a[i][j] && x[j]);
        }
        return 0;
    }
}
```

7.15 LikeEuclid

```
ll cal(ll a,ll b,ll c,ll n) { // sum_{i=0...n-1}floor((a*i+b)/c)
  if(n == 0) return 0;
  return (b/c)*n+(a/c)*n*(n-1)/2+(a%c?cal(c,(a*n+b)%c,a%c,(a%c*n+b%c)/c):0);
}
```

7.16 LinearBasis

```
// 普通集合线性基
const int M=63;
struct LB{
    ll a[M];
    void Clear() { memset(a,0,sizeof(a)); }
    void ins(ll x){
        for(int i=M-1; ~i && x; --i) if (x>>i&1)
            if (a[i]) x^=a[i]; else { a[i]=x; break; }
    }
};
```

```
// 可持久化线性基 ( 序列前缀最右线性基 )
const int M=32;
struct LB{
   ll a[M]; int id[M];
   void Clear() { memset(a,0,sizeof(a)); }
   void Copy(const LB &L) { rep(i,0,M) a[i]=L.a[i],id[i]=L.id[i]; }
   void Ins(LB &L,ll x,int no) {
      Copy(L);
      for (int i=M-1; ~i && x; --i) if (x>>i&1) {
         if (!a[i]) a[i]=x,id[i]=no;
         else if (no>id[i]) swap(a[i],x),swap(id[i],no);
         x^=a[i];
      }
   }
   ll Qry(int l,ll x=0) { per(i,0,M) if (id[i]>=l) x=max(x,x^a[i]); return x; }
} B[N];

// 集合线性基求交与查询
const int M=33;
ll tmp[M];
struct LB{
   ll a[M];
   LB() { mem(a,0); }
   void Clear() { mem(a,0); }
   void Copy(LB &A) { rep(i,0,M) a[i]=A.a[i]; }
   // 向 this 中插入 x ，返回  y 在后来插入元素中的投影
   bool I(ll x,ll &y) {
      y=x;
      for(int i=M-1; ~i && x; --i) if (x>>i&1)
         if (a[i]) x^=a[i],y^=tmp[i]; else { a[i]=x,tmp[i]=y; return 1; }
      return 0;
   }
   bool Q(ll x) {
      for(int i=M-1; ~i && x; --i) if (x>>i&1)
         if (a[i]) x^=a[i]; else return 0;
      return 1;
   }
   friend void Intersect(LB &A,LB &B,LB &C) {
      LB AA; ll y,z; AA.Copy(A),C.Clear(); mem(tmp,0);
      per(i,0,M) if (B.a[i]) if (!AA.I(B.a[i],y)) C.I(y,z);
   }
   // 化为最简型，方便线性空间的  hash
   void build () { per(i,0,M) per(j,0,i) a[i]=min(a[i],a[i]^a[j]); }
};
```

7.17 LinearRecursion

```
// a_{m} = \sum_{j=0}^{-m1}a_{j}*c_{j} O(m^2lgn)
int linear_recurrence(ll n, int m, vi a, vi c) {
   if (n<m) return (a[n]+P)%P;
   vector<ll> v(m, 0), u(m<<1, 0);
   v[0] = 1;
   for(ll x = 0, W = n ? 1ll<<(63 - __builtin_clzll(n)) : 0; W; W >>= 1, x <<= 1) {
      fill(all(u), 0);
      int b = !!(n & W); if(b) x++;
      if(x < m) u[x] = 1;
      else {
         rep(i, 0, m) rep(j, 0, m) (u[i + b + j] += v[i] * v[j]) %= P;
         per(i, m, 2*m) rep(j, 0, m) (u[i - m + j] += c[j] * u[i]) %= P;
      }
      copy(u.begin(), u.begin() + m, v.begin());
   }
   ll ans = 0;
   rep(i, 0, m) (ans += v[i] * a[i]) %= P;
   return (ans+P)%P;
}
```

7.18 MathFunction

```
const int N = 1e6 + 7;
int n, M, f[N], g[N], h[N], phi[N], u[N], p[N];
// f[n] 为 n 的最小质因子 ; g[n]=f[n]^k; phi[n] 为欧拉函数 ; u[n] 为莫比乌斯函数 ; h[n] 为一
般积性函数

void prime(int n) {
   u[1]=phi[1]=1,h[1]=(0); // 1 的时候特判
   rep(i, 2, n+1) {
      if (!f[i]) {
         p[++M]=i;
         f[i] = g[i] = i;
         phi[i] = i - 1;
         u[i] = -1;
         h[i] = (0);
      } // 质数的时候特判
      for (int j = 1, k; j <= M && p[j] <= f[i] && i * p[j] <= n; j++){
         f[k = i * p[j]] = p[j];
         if (p[j] < f[i]) {
            g[k] = p[j];
            phi[k] = phi[i] * phi[p[j]];
            u[k] = u[i] * u[p[j]];
            h[k] = h[i] * h[p[j]];
         }else {
            g[k] = g[i] * p[j];
            phi[k] = phi[i] * p[j];
            u[k] = 0;
            h[k] = h[i / g[i]] * (0);
         } /* 质数次幂特判 */
         // phi[i*p[j]]=phi[i]*(p[j]<f[i]?phi[p[j]]:p[j]);
         // u[i*p[j]]=u[i]*(p[j]<f[i]?u[p[j]]:0);
      }
      /*phi[i*j]=phi[i]*phi[j] (gcd(i,j)=1)
               phi[i]*j (j|i)
        u[i*j]=u[i]*u[j] (gcd(i,j)=1)
               0 (j|i)
      */
   }
}
```

7.19 NTT

```
const int M = 1 << 17 << 1;
int a[M], b[M];

struct NTT{
   static const int G = 3, P = 1004535809; //P = C*2^k + 1
   int N, na, nb, w[2][M], rev[M];
   ll kpow(ll a, int b){
      ll c = 1;
      for (; b; b >>= 1,a = a * a % P) if (b & 1) c = c * a %P;
      return c;
   }
   void FFT(int *a, int f){
      rep(i, 0, N) if (i < rev[i]) swap(a[i], a[rev[i]]);
      for (int i = 1; i < N; i <<= 1)
         for (int j = 0, t = N / (i << 1); j < N; j += i << 1)
            for (int k = 0, l = 0, x, y; k < i; k++, l += t)
               x = (ll) w[f][l] * a[j+k+i] % P, y = a[j+k], a[j+k] = (y+x) % P, a[j+k+i]
                 = (y-x+P) % P;
      if (f) for (int i = 0, x = kpow(N, P-2); i < N; i++) a[i] = (ll)a[i] * x % P;
   }
   void work(){
      int d = __builtin_ctz(N);
      w[0][0] = w[1][0] = 1;
      for (int i = 1, x = kpow(G, (P-1) / N), y = kpow(x, P-2); i < N; i++) {
         rev[i] = (rev[i>>1] >> 1) | ((i&1) << (d-1));
         w[0][i] = (ll)x * w[0][i-1] % P, w[1][i] = (ll) y * w[1][i-1] % P;
      }
   }
   void doit(int *a, int *b, int na, int nb){ // [0, na)
      for (N = 1; N < na + nb - 1; N <<= 1);
      rep(i, na, N) a[i] = 0;
      rep(i, nb, N) b[i] = 0;
      work(), FFT(a,0), FFT(b,0);
      rep(i, 0, N) a[i] = (ll)a[i] * b[i] % P;
      FFT(a, 1);
      //rep(i, 0, N) cout << a[i] << endl;
   }
} ntt;
```

7.20 Polya

```
/*
 * Burnside's lemma
 * 首先列出所有可能的染色方案，然后找出每个置换下保持不变的方案（不动点）数。
 * 等价类数目：所有置换的不动点数的平均值。
 * Polya enumeration theorem
 * 一个循环的颜色需相同
 */
```

7.21 Rho

```
using uint128 = __uint128_t;
using ull = unsigned long long;
using ll = long long;
using uint = unsigned int;
using pli = pair<ull, uint>;
namespace prime {
   inline ull sqr(ull x) { return x * x; }
   inline uint isqrt(ull x) { return sqrtl(x); }
   inline uint ctz(ull x) { return __builtin_ctzll(x); }
   template <class T>
   T gcd(T a, T b) {  while (b) { T t = a % b; a = b; b = t; } return a; }

   template <class T, class dT, class sT>
   struct Mod {
      static T mod, inv, r2;
      static const int wb = sizeof(T) * 8;
      T x;
      Mod(): x(0) {}
      Mod(T _x): x(init(_x)) {}
      bool operator == (const Mod& rhs) const { return x == rhs.x; }
      bool operator != (const Mod& rhs) const { return x != rhs.x; }
      Mod operator += (const Mod& rhs) { if ((x += rhs.x) >= mod) x -= mod; return *this
      ; }
      Mod operator -= (const Mod& rhs) { if (sT(x -= rhs.x) < 0) x += mod; return *this;
       }
      Mod operator *= (const Mod& rhs) { x = reduce(dT(x) * rhs.x); return *this; }
      Mod operator + (const Mod &rhs) const { return Mod(*this) += rhs; }
      Mod operator - (const Mod &rhs) const { return Mod(*this) -= rhs; }
      Mod operator * (const Mod &rhs) const { return Mod(*this) *= rhs; }
      Mod operator - () const { return Mod() - *this; }
      Mod pow(ull e) const {
         Mod r(1); for (Mod a = *this; e; e >>= 1, a *= a) if (e & 1) r *= a;
         return r;
      }
      T get() const { return reduce(x); }
      static T modulus() { return mod; }
      static T init(T w) { return reduce(dT(w) * r2); }
      static void set_mod(T m) { mod = m, inv = mul_inv(mod), r2 = -dT(mod) % mod; }
      static T reduce(dT x) {
         T y = T(x >> wb) - T((dT(T(x) * inv) * mod) >> wb);
         return sT(y) < 0 ? y + mod : y;
      }
      static T mul_inv(T n, int e = 6, T x = 1) {
         return !e ? x : mul_inv(n, e - 1, x * (2 - x * n));
      }
   };

   using Mod64 = Mod<ull, uint128, ll>;
   using Mod32 = Mod<uint, ull, int>;
   template <> ull Mod64::mod = 0;
   template <> ull Mod64::inv = 0;
   template <> ull Mod64::r2 = 0;
   template <> uint Mod32::mod = 0;
   template <> uint Mod32::inv = 0;
   template <> uint Mod32::r2 = 0;

   template <class T, class mod>
```

```
bool composite(T n, const uint* bases, int m) {
   mod :: set_mod(n);
   int s = __builtin_ctzll(n - 1);
   T d = (n - 1) >> s;
   mod one(1), fone(n - 1);
   for (int i = 0, j; i < m; ++i) {
      mod a = mod(bases[i]).pow(d);
      if (a == one || a == fone) continue;
      for (j = s - 1; j > 0; --j) { if ((a *= a) == fone) break; }
      if (j == 0) return 1;
   }
   return 0;
}
bool is_prime(ull n) { // reference: http://miller-rabin.appspot.com
   assert(n < (ull(1) << 63));
   static const uint bases[][7] = {
      {2, 3},
      {2, 299417},
      {2, 7, 61},
      {15, 176006322, 4221622697u},
      {2, 2570940, 211991001, 3749873356u},
      {2, 2570940, 880937, 610386380, 4130785767u},
      {2, 325, 9375, 28178, 450775, 9780504, 1795265022}
   };
   if (n <= 1) return 0;
   if (!(n & 1)) return n == 2;
   if (n <= 8) return 1;
   int x = 6, y = 7;
   if (n < 1373653) x = 0, y = 2;
   else if (n < 19471033) x = 1, y = 2;
   else if (n < 4759123141) x = 2, y = 3;
   else if (n < 154639673381) x = y = 3;
   else if (n < 47636622961201) x = y = 4;
   else if (n < 3770579582154547) x = y = 5;
   if (n < (1u << 31)) return !composite<uint, Mod32>(n, bases[x], y);
   return !composite<ull, Mod64>(n, bases[x], y);
}
struct ExactDiv {
   ExactDiv() {}
   ExactDiv(ull n) : n(n), i(Mod64::mul_inv(n)), t(ull(-1) / n) {}
   friend ull operator / (ull n, ExactDiv d) { return n * d.i; };
   bool divide(ull n) { return n / *this <= t; }
   ull n, i, t;
};
vector<ExactDiv> primes;
void init(uint n) {
   uint sqrt_n = sqrt(n);
   vector<bool> is_prime(n + 1, 1);
   primes.clear();
   for (uint i = 2; i <= sqrt_n; ++i) if (is_prime[i]) {
      if (i != 2) primes.pb(ExactDiv(i));
      for (uint j = i * i; j <= n; j += i) is_prime[j] =  0;
   }
}
```

```
template <class T, class mod>
T brent(T n, T c) { // n must be composite and odd.
   const ull s = 256;
   mod::set_mod(n);
   const mod one = mod(1), mc = mod(c);
   mod y = one;
   for (ull l = 1; ; l <<= 1) {
      auto x = y;
      for (int i = 0; i < (int)l; ++i) y = y * y + mc;
      mod p = one;
      for (int k = 0; k < (int)l; k += s) {
         auto sy = y;
         for (int i = 0; i < (int)min(s, l - k); ++i) {
            y = y * y + mc;
            p *= y - x;
         }
         T g = gcd(n, p.x);
         if (g == 1) continue;
         if (g == n) for (g = 1, y = sy; g == 1; ) y = y * y + mc, g = gcd(n, (y - x)
         .x);
         return g;
      }
   }
}
ull brent(ull n, ull c) {
   if (n < (1u << 31)) return brent<uint, Mod32>(n, c);
   return brent<ull, Mod64>(n, c);
}
vector<pli> factors(ull n) {
   assert(n < (1ull << 63));
   if (n <= 1) return {};
   vector<pli> ret;
   ull v2 = sqrtl(n), v3 = cbrtl(n), v = v2, b = 2;
   if (v2 * v2 == n || v3 * v3 * v3 == n) {
      if (v2 * v2 != n) v = v3, b = 3;
      ret = factors(v);
      for (auto &&e: ret) e.se *= b;
      return ret;
   }
   if (!(n & 1)) {
      uint e = ctz(n);
      ret.emplace_back(2, e);
      n >>= e;
   }
   ull lim = sqr(primes.back().n);
   for (auto &&p: primes) {
      if (sqr(p.n) > n) break;
      if (p.divide(n)) {
         uint e = 1; n = n / p;
         while (p.divide(n)) n = n / p, e++;
         ret.emplace_back(p.n, e);
      }
   }
   uint s = ret.size();
   while (n > lim && !is_prime(n)) {
        for (ull c = 1; ; ++c) {
            ull p = brent(n, c);
            if (!is_prime(p)) continue;
            uint e = 1; n /= p;
            while (n % p == 0) n /= p, e += 1;
            ret.emplace_back(p, e);
            break;
        }
    }
    if (n > 1) ret.emplace_back(n, 1);
    if (ret.size() - s >= 2) sort(ret.begin() + s, ret.end());
    return ret;
  }
}
```

7.22 Simplex

```
const db EPS = 1e-8, DINF = 1e15;
struct Simplex {
  static const int M = 550;
  int n, m, B[M], N[M];
  db v, ans[M], b[M], c[M], A[M][M]; // 全幺模矩阵可以改整数
  /* n - variables, m - equations
   * maxf(x)=cx
   * s.t.Ax<=b,x>=0
   */
  void init(int _n, int _m) {
    n = _n, m = _m, v = 0;
    rep(i, 1, n + 1) N[i] = i;
    rep(i, 1, m + 1) B[i] = i + n;
  }
  inline int sgn(db x) { return (x > EPS) - (x < -EPS); }
  void pivot(int l, int e) {
    db tmp = A[l][e];
    b[l] /= tmp, A[l][e] = 1 / tmp;
    rep(i, 1, n + 1) if (i != e) A[l][i] /= tmp;
    rep(i, 1, m + 1) if (i != l && sgn(A[i][e])) {
      b[i] -= A[i][e] * b[l];
      rep(j, 1, n + 1) A[i][j] -= (j!=e) * A[i][e] * A[l][j]; // 可以链式优化
      A[i][e] = - A[i][e] / tmp;
    }
    rep(i, 1, n + 1) c[i] -= (i!=e) * c[e] * A[l][i];
    v += b[l] * c[e]; c[e] *= -A[l][e]; swap(B[l], N[e]);
  }
  bool ini(){ // 随机化初始解
    while(1){
      int l = -1, e = -1;
      rep(i, 1, m+1) if (sgn(b[i]) < 0 && (l == -1 || (rand() & 1))) l = i;
      if(l == -1) break;
      rep(j, 1, n+1) if (sgn(A[l][j]) < 0 && (e == -1 || (rand() & 1))) e = j;
      if(e == -1) return 0;
      pivot(l, e);
    }
    return 1;
  }
  db run() {
    //if (!ini()) return -DINF; // 无解  b < 0 need ini()
    rep(i, 1, n+1) ans[i] = 0;
    while (1) {
      int r, l, e = -1;
      db delt = -DINF;
      rep(j, 1, n + 1) if (sgn(c[j]) > 0) { // 找非基变量
        db tmp = DINF;
        rep(i, 1, m + 1) if (sgn(A[i][j]) > 0 && b[i] / A[i][j] < tmp) // 找基变量
          r = i, tmp = b[i] / A[i][j];
        if (tmp == DINF) return DINF; // 无界
        if (delt < tmp * c[j]) l = r, e = j, delt = tmp * c[j];
        break;
        // 贪心取最大如果矩阵为全幺模或  0 很多可以加上 break 因为转轴代价可能较小
      }
      if (e == -1) break; // 找到最优解
      pivot(l, e);
    }
    rep(i, 1, m+1) if (B[i] <= n) ans[B[i]] = b[i];
    return v;
  }
} sp;
```

7.23 Simpson

```
namespace Simpson {
  const db eps = 1e-10; // 精度感觉一般要多设 1e-3 左右
  inline db F(db x) { F(x) = (?) }
  inline db simpson(db fa, db fb, db fc, db a, db c) {
    return (fa + 4 * fb + fc) * (c - a) / 6;
  }
  db asr(db a, db b, db c, db esp, db A, db fa, db fb, db fc) {
    db ab = (a + b) / 2, bc = (b + c) / 2;
    db fab = F(ab), fbc = F(bc);
    db L = simpson(fa, fab, fb, a, b), R = simpson(fb, fbc, fc, b, c);
    if (fabs(L + R - A) <= 15 * esp) return L + R + (L + R - A) / 15.0;
    return asr(a, ab, b, esp / 2, L, fa, fab, fb) + asr(b, bc, c, esp / 2, R, fb, fbc,
     fc);
  }
  // f(a, c)
  db asr(db a, db c, db eps) {
    db b = (a + c) / 2;
    db fa = F(a), fb = F(b), fc = F(c);
    return asr(a, b, c, eps, simpson(fa, fb, fc, a, c), fa, fb, fc);
  }
}
```

7.24 SternBrocotTree

```
namespace SBT {
  typedef long double db;
  typedef int U;
  typedef pair<U, U> pii;
  const U INF = 1e9 + 7;
```

```
    typedef __int128 T;
    typedef pair<T, T> V; // V = [double|long double|fraction]
    inline int cmp(const V &a, const V &b) {
        T x = a.fi * b.se - a.se * b.fi;
        return (x > 0) - (x < 0);
    }
    inline bool in(const V &a, const V &b, const V &c) {
        return 0 <= cmp(c, a) && cmp(c, b) < 0;
    }
    pii operator+(const pii &a, const pii &b) { return mp(a.fi + b.fi, a.se + b.se); }
    pii operator*(const pii &a, U x) { return mp(a.fi * x, a.se * x); }
    bool search(V v, U MAXB, pii &lo, pii &hi, int f) {
        V x;
        U l = 0, r = f > 0 ? (hi.se ? (MAXB - lo.se) / hi.se : INF) :
            (lo.se ? (MAXB - hi.se) / lo.se : INF);
        while (l + 1 < r) {
            U z = (l + r) >> 1;
            x = f > 0 ? lo + hi * z : lo * z + hi;
            f * cmp(x, v) <= 0 ? l = z : r = z;
        }
        x = f > 0 ? lo + hi * r : lo * r + hi;
        r = f * cmp(x, v) <= 0 ? r : l;
        f > 0 ? lo = lo + hi * r : hi = lo * r + hi;
        return r > 0;
    }
    pii solve(V v, U MAXB) { // find ROUND_HALF_UP(a / b) = v, b <= MAXB
        V L = mp(v.fi * 10 - 5, v.se * 10);
        V R = mp(v.fi * 10 + 5, v.se * 10);
        pii lo(0, 1), hi(1, 0);
        while (true) {
            bool ok = 0;
            //V m = mp(lo.fi + hi.fi, lo.se + hi.se);
            //if (in(L, R, m)) return mp(m.fi, m.se);
            ok |= search(v, MAXB, lo, hi, 1);
            ok |= search(v, MAXB, lo, hi, -1);
            if (!ok) break;
        }
        db t1 = (db) lo.fi / lo.se;
        db t2 = (db) hi.fi / hi.se;
        db t3 = (db) v.fi / v.se;
        if (t2 - t3 <= t3 - t1) return hi;else return lo;
        //if (in(L, R, lo)) return lo;
        //if (in(L, R, hi)) return hi;
        return mp(-1, -1);
    }
};
```

7.25 min_25

```
struct Min_25{
    // F(i) 要拆成多个完全积性函数的和
    static const int N = 1e6 + 7;
    int Sqr, m, p[N], id1[N], id2[N], tot, cntp;
    ll g[N], sp[N], h[N], n, w[N];
    bool isp[N];
    // f(p) = p ^ k
    ll f(int p) { return 1;}

    // 要求的积性函数 F(p ^ e)
    ll F(int p, int e) { return e == 1 ? -1 : 0;}

    // 假设都是质数的完全积性函数前缀和去掉  f(1)
    ll calc(ll n) { return n - 1;}

    void prime(int n){
        cntp = 0;isp[1] = 1;
        rep(i, 2, n+1) {
            if(!isp[i]) p[++cntp] = i;
            for(int j = 1; j <= cntp && i * p[j] <= n; j++){
                    isp[i * p[j]] = 1;
                    if(i % p[j] == 0)break;
            }
        }
        rep(i, 1, cntp+1) sp[i] = sp[i - 1] + f(p[i]);
        p[++cntp] = INT_MAX;
    }

    ll S(ll x, int y){
        if(x <= 1 || p[y] > x) return 0;
        int k = (x <= Sqr ? id1[x] : id2[n/x]);
        ll ret = -(g[k] - sp[y-1]);// 质数的答案
        for(int i = y; i <= tot && 1ll * p[i] * p[i] <= x; i++){
            ll t1 = p[i], t2 = 1ll * p[i] * p[i];
            for(int e = 1; t2 <= x; e++, t1 = t2, t2 *= p[i]) {
                if (F(p[i], e)) ret += S(x / t1, i + 1) * F(p[i], e);
                ret += F(p[i], e + 1);// 合数的答案
            }
        }
        return  ret;
    }

    ll solve(ll _n) {
        n = _n;if (n == 0) return 0;
        m = 0;Sqr = sqrt(n);
        tot = upper_bound(p + 1, p + cntp + 1, Sqr) - (p + 1);
        for(ll i = 1, j; i <= n; i = j + 1){
            j = n / (n / i);
            w[++m] = n / i;
            g[m] = calc(w[m]);
            w[m] <= Sqr ? id1[w[m]] = m : id2[j] = m;
        }
        rep(j, 1, tot + 1)
            for(int i = 1; i <= m && 1ll * p[j] * p[j] <= w[i]; i++){
                ll t = w[i] / p[j];
                int k = t <= Sqr ? id1[t] : id2[n / t];
                g[i] -= f(p[j]) * (g[k] - sp[j - 1]);
            }
        return S(n,1) + 1;
    }
} _U;
```

7.26 ploynomial

```
template<class T>
struct polynomial{
   static const int N = 101010;
   static const int P = 998244353;
   T a1[N], b1[N], c[N], a[N], pre[N], suf[N], ifac[N], fac[N];
   T add(T a, T b) {a = (a + b) % P; return a < 0 ? a + P : a;}
   T mul(T a, T b) {a = 1ll * a * b % P; return a < 0 ? a + P : a;}
   T kpow(T a, T b) {T r=1;for(;b;b>>=1,a=mul(a,a)) {if(b&1)r=mul(r,a);}return r;}
   void calc(int n, T *a, T *b) {
      fill_n(c, n+1, 0);
      rep(i, 0, n+1) rep(j, 0, 2) c[i+j] = add(c[i+j], mul(a[i], b[j]));
      memcpy(a, c, sizeof(a[0]) * (n+1));
   }
   void solve(int n, T *x, T *y){ // a[0]*x^0 ... a[n]*x^n
      fill_n(a, n+1, 0);
      rep(i, 0, n+1) {
         fill_n(a1, n+1, 0); a1[0] = 1;
         rep(j, 0, n+1) if (j != i) a1[0] = mul(a1[0], x[i] - x[j]);
         a1[0] = mul(y[i], kpow(a1[0], P - 2));
         rep(j, 0, n+1) if (j != i) {
            b1[0] = -x[j]; b1[1] = 1;
            calc(n, a1, b1);
         }
         rep(j, 0, n+1) a[j] = add(a[j], a1[j]);
      }
   }
   T get(int n, int k, T *x, T *y) { // f(k)
      T res = 0;
      rep(i, 0, n+1) {
         T s1 = y[i], s2 = 1;
         rep(j, 0, n+1) if (j != i) s1 = mul(s1, k - x[j]);
         rep(j, 0, n+1) if (j != i) s2 = mul(s2, x[i] - x[j]);
         res = add(res, mul(s1, kpow(s2, P - 2)));
      }
      return res;
   }
   T get(int n, int k, T *y) { // x is [1..n]
      fac[0] = 1;rep(i, 1, n+1) fac[i] = mul(fac[i-1], i);
      ifac[n] = kpow(fac[n], P - 2);
      per(i, 0, n) ifac[i] = mul(ifac[i+1], i+1);
      pre[0] = suf[n+1] = 1;
      rep(i, 1, n+1) pre[i] = mul(pre[i-1], k - i);
      per(i, 1, n+1) suf[i] = mul(suf[i+1], k - i);
      T ans=0;
      rep(i, 1, n+1){
         T s1 = mul(pre[i-1], suf[i+1]);
         T s2 = mul(ifac[i-1], ifac[n-i]);
         T fg = (n-i)&1 ? -1 : 1;
         ans = add(ans, mul(fg*s1, mul(s2, y[i])));
      }
      return ans;
   }
};
```

7.27 polysum

```
struct polysum {
   static const int D = 101000, P = 998244353;
   ll a[D], fac[D], ifac[D], p1[D], p2[D], h[D][2], C[D];
   ll add(ll a, ll b) {a = (a + b) % P; return a < 0 ? a + P : a;}
   ll mul(ll a, ll b) {a = 1ll * a * b % P; return a < 0 ? a + P : a;}
   ll kpow(ll a, ll b) {ll r=1;for(;b;b>>=1,a=mul(a,a)) {if(b&1)r=mul(r,a);}return r;}
   void init(int M) {
      fac[0] = 1; rep(i, 1, M+5) fac[i] = mul(fac[i-1], i);
      ifac[M+4] = kpow(fac[M+4], P - 2);
      per(i, 0, M+4) ifac[i] = mul(ifac[i+1], i+1);
   }
   ll calcn(int d, ll *a, ll n) { // a[0].. a[d]  a[n]
      if (n <= d) return a[n];
      p1[0] = p2[0] = 1;
      rep(i, 0, d+1) p1[i+1] = mul(p1[i], (n - i) % P);
      rep(i, 0, d+1) p2[i+1] = mul(p2[i], (n - d + i) % P);
      ll ans=0;
      rep(i, 0, d+1) {
         ll s1 = mul(p1[i], p2[d - i]);
         ll s2 = mul(ifac[i], ifac[d - i]);
         ll t = mul(mul(s1, s2), a[i]);
         ans = (d-i)&1 ? add(ans, -t) : add(ans, t);
      }
      return ans;
   }
   ll Polysum(ll n, ll *a, ll m) { // a[0].. a[m] \sum_{i=0}^{n-1} a[i]
      a[m+1] = calcn(m, a, m+1);
      rep(i, 1, m+2) a[i] = add(a[i-1], a[i]);
      return calcn(m+1, a, n-1);
   }
   ll qpolysum(ll R, ll n, ll *a, ll m) { // a[0].. a[m] \sum_{i=0}^{n-1} a[i]*R^i
      if (R == 1) return Polysum(n, a, m);
      a[m+1] = calcn(m, a, m+1);
      ll r = kpow(R, P - 2), p3 = 0, p4 = 0, c, ans;
      h[0][0] = 0; h[0][1] = 1;
      rep(i, 1, m+2) {
         h[i][0] = mul(h[i-1][0] + a[i-1], r);
         h[i][1] = mul(h[i-1][1], r);
      }
      rep(i, 0, m+2) {
         ll t = mul(ifac[i], ifac[m+1-i]);
         p3 = i & 1 ? add(p3, -mul(h[i][0], t)) : add(p3, mul(h[i][0], t));
         p4 = i & 1 ? add(p4, -mul(h[i][1], t)) : add(p4, mul(h[i][1], t));
      }
      c = mul(kpow(p4, P - 2), -p3);
      rep(i, 0, m+2) h[i][0] = add(h[i][0], h[i][1] * c);
      rep(i, 0 ,m+2) C[i] = h[i][0];
      ans = add(mul(calcn(m, C, n), kpow(R, n)), -c);
      return ans;
   }
};
```

7.28 prime

```
// time : O(n)
// low[] : optional
const int N = 1e6 + 6;
int low[N],cntp,p[N];
bool isp[N];
void getprime() {
   fill_n(isp + 2, N - 2, 1);
   rep(i, 2, N) {
      if (isp[i]) p[cntp++]=i;
      for (int j=0;j<cntp&&p[j]*i<N;j++){
         //low[p[j] * i] = p[j];
         isp[p[j] * i] = 0;
         if (i % p[j] == 0) break;
      }
   }
}

// 优化版欧拉筛法 bitset 需要 O2
const int N = 3e7 + 6, M = 2e6 + 6;
// int low[N],
bitset<N / 3 + 1> isp;
int cntp,p[M];
void getprime(int N) {
   cntp = 2;p[0] = 2;p[1] = 3;
   for (int i = 5, k = 1; i <= N; (k & 1) ? i+=2 : i+=4 , k++){
      if (!isp[k]) {
         p[cntp++]=i;
         // low[i] = i;
      }
      for (int j = 2; j < cntp && p[j] * i <= N; j++) {
         // low[p[j] * i] = p[j];
         isp[p[j] * i / 3] = 1;
         if (i % p[j] == 0) break;
      }
   }
}

// 优化埃氏筛法空间最小可以不存质数
const int N = 3e8 + 6, M = 2e7 + 6;
int cntp,p[M];
bitset<N / 3 + 1> bit;

void getprime(int n){
   int i, j;
   cntp = 2; p[0] = 2; p[1] = 3;
   for(i = 5, j = 1; i * i <= n; (j & 1) ? i += 2 : i += 4 , j++) {
      if(bit[j] == 0) {
         p[cntp++]=i;
         for(int j = i * i; j <= n ; j += i)
            if(j % 2 != 0 && j % 3 != 0) bit[j / 3] = 1;
      }
```

```
   }
   for( ; i <= n; (j&1) ? i+=2 : i+=4 , j++) if(bit[j] == 0) p[cntp++]=i;
}
```

7.29 划分数

```
const int N = 1e6 + 5, P = 998244353;
int n, f[N], fv[N];
inline int add(int a, int b) { if((a += b) >= P) a -= P; return a < 0 ? a + P : a;}
void init(int n) {
   f[0] = f[1] = 1;
   int m = sqrt(n) + 1;
   rep(i, 1, m+1) fv[i] = i * (3 * i - 1) / 2;
   rep(i, 2, n+1) {
      for(int j = 1; fv[j] <= i; j++) {
         f[i] = add(f[i], j & 1 ? f[i - fv[j]] : -f[i - fv[j]]);
         if (fv[j] + j <= i) f[i] = add(f[i], j & 1 ? f[i - fv[j] - j] : -f[i - fv[j] -
         j]);
      }
   }
}
```

7.30 原根

```
ll Pow(ll x, ll k, ll p) {
   ll ret = 1;
   for (; k; k >>= 1, x = x*x%p) if (k & 1) ret = ret*x%p;
   return ret;
}
struct Euler {
   vector<ll> c;
   inline bool check_g(ll g, ll p) {
      rep(i, 0, sz(c)) if (Pow(g, c[i], p) == 1) return 0;
      return 1;
   }
   inline ll getRoot(ll p) {
      c.clear();
      ll tmp = p - 1, g;
      for (ll k = 2; k*k <= tmp; ++k) if (tmp % k == 0) {
         c.pb(k);
         while (tmp % k == 0) tmp /= k;
      }
      if (tmp != 1) c.pb(tmp);
      rep(i, 0, sz(c)) c[i] = (p - 1) / c[i];
      for (g = 1; !check_g(g, p); ++g);
      return g;
   }
};
```

7.31 原根_合数

```
ll kpow(ll x, ll k, ll p) {
   ll ret = 1;
   for (; k; k >>= 1, x = x*x%p) if (k & 1) ret = ret*x%p;
   return ret;
}
struct Euler {
   vector<ll> P, A; ll phi, g;
   inline bool check_g(ll g, ll p) {
      rep(i, 0, sz(P)) if (kpow(g, P[i], p) == 1) return 0;
      return 1;
   }
   inline void factor(ll m) {
      P.clear(), A.clear();
      for (ll k = 2; k*k <= m; ++k) if (m%k == 0) {
         int cnt = 0;
         while (m%k == 0) m /= k, cnt++;
         P.pb(k), A.pb(cnt);
      }
      if (m > 1) P.pb(m), A.pb(1);
   }
   inline bool check(ll m) {
      //if (m==1 || m==2 || m==4) return 1;
      factor(m);
      if (sz(P) > 2 || sz(P) == 1 && P[0] == 2) return 0;
      if (sz(P) == 1) return 1;
      if (P[0] != 2 || P[0] == 2 && A[0] > 1) return 0;
      return 1;
   }
   inline ll getRoot(ll p) {
      if (p == 1 || p == 2 || p == 4) return phi = p + 1 >> 1, p – 1;
      if (!check(p)) return –1;
      phi = p;
      for (auto t : P) phi = phi / t*(t – 1);
      factor(phi);
      for (auto &t : P) t = phi / t;
      for (g = 1; __gcd(g, p) != 1 || !check_g(g, p); ++g);
      return g;
   }
   inline vector<ll> getAllRoot(ll p) {
      vector<ll> ret; ll g = getRoot(p);
      if (g == –1) return ret;
      rep(i, 0, phi) if (__gcd((ll)i, phi) == 1) ret.pb(kpow(g, i, p));
      sort(all(ret)); return ret;
   }
};
```

7.32 离散对数

```
ll kpow(ll a, ll b, ll P) {
   ll r = 1;
   for (; b; b >>= 1, a = a * a % P) if (b & 1) r = r * a % P;
   return r;
}

void ex_gcd(int a, int b, int &x, int &y) {
   b ? (ex_gcd(b, a % b, y, x), y –= a / b * x) : (x = 1, y = 0);
}
```

```
inline int Inv(int a, int P) {
   int x, y; ex_gcd(a, P, x, y);
   return x < 0 ? x + P : x;
}

struct BSGS {
   unordered_map<ll, int> M;
   ll bsgs(ll x, ll z, ll P) {
      if (x % P == 0) return –1;
      ll res = z % P, sa, t = 1, sq = sqrt(P); M.clear();
      rep(i, 0, sq + 1) { if (M.count(t)) break; M[t] = i, t = t * x % P; }
      t = P / sq, sa = Inv(kpow(x, sq, P), P);
      rep(i, 0, t + 1) if (M.count(res))
         return i * sq + M[res]; else res = res * sa % P;
      return –1;
   }
   ll ex_bsgs(ll x, ll z, ll P) { //x^y==z(mod P)
      ll t = 1 % P, w = 1, ans, c = 0; z %= P;
      rep(i, 0, 51) { if (t == z) return i; t = t * x % P; }
      for (t = __gcd(x, P); t != 1; t = __gcd(x, P)) {
         if (z % t) return –1;
         z /= t, P /= t, w = w * x / t % P, c++;
         if (z == w) return c;
      }
      z = z * Inv(w, P) % P, ans = bsgs(x, z, P);
      return ans + (ans != –1) * c;
   }
};
```

7.33 高次同余_合数

```
ll kpow(ll a, ll b, ll P) {
   ll r = 1; assert(b >= 0);
   for (; b; b >>= 1, a = a * a % P) if (b & 1) r = r * a % P;
   return r;
}
void ex_gcd(ll a, ll b, ll &x, ll &y) {
   b ? (ex_gcd(b, a % b, y, x), y –= a / b * x) : (x = 1, y = 0);
}
inline ll Inv(ll a, ll P) {
   ll x, y; ex_gcd(a, P, x, y);
   return x < 0 ? x + P : x;
}
struct BSGS {
   map<ll, int> M;
   ll bsgs(ll x, ll z, ll P) {
      if (x % P == 0) return –1;
      ll res = z % P, sa, t = 1, sq = sqrt(P); M.clear();
      rep(i, 0, sq + 1) { if (M.count(t)) break; M[t] = i, t = t * x % P; }
      t = P / sq, sa = Inv(kpow(x, sq, P), P);
      rep(i, 0, t + 1) if (M.count(res))
         return i * sq + M[res]; else res = res * sa % P;
      return –1;
   }
```

```cpp
  ll ex_bsgs(ll x, ll z, ll P) { //x^y==z(mod P)
    ll t = 1 % P, w = 1, ans, c = 0; z %= P;
    rep(i, 0, 51) { if (t == z) return i; t = t * x % P; }
    for (t = __gcd(x, P); t != 1; t = __gcd(x, P)) {
      if (z % t) return -1;
      z /= t, P /= t, w = w * x / t % P, c++;
      if (z == w) return c;
    }
    z = z * Inv(w, P) % P, ans = bsgs(x, z, P);
    return ans + (ans != -1) * c;
  }
};
struct CRT {
  ll M, R; static const int N = 55;
  ll a[N], mod[N];
  void exgcd(ll a, ll b, ll &x, ll &y) {
    if (b == 0) { x = 1; y = 0; return; }
    exgcd(b, a % b, y, x);
    y -= a / b * x;
  }
  ll Inv(ll a, ll mod) {
    ll x = 0, y = 0;
    exgcd(a, mod, x, y);
    x %= mod;
    return x < 0 ? x + mod : x;
  }
  ll solve(ll n) {
    M = mod[1], R = a[1];
    rep(i, 2, n + 1) {
      ll g = __gcd(M, mod[i]);
      ll inv = Inv(M / g, mod[i] / g);
      if ((a[i] - R) % g) return -1; // 无解
      R += inv * ((a[i] - R) / g) % (mod[i] / g) * M;
      M = M / g * mod[i];
      R = (R % M + M) % M; // 可能为 0 看是否需要是正整数
    }
    return R;
  }
} crt;

typedef vector<ll> vll;
typedef pair<ll, ll> pll;
struct Euler {
  vll P, A, _P, _A; ll phi, g, phi_phi; BSGS T;
  inline bool check_g(ll g, ll p) {
    rep(i, 0, sz(P)) if (kpow(g, P[i], p) == 1) return 0;
    return 1;
  }
  inline void factor(ll m, vll &P, vll &A) {
    P.clear(), A.clear();
    for (ll k = 2; k*k <= m; ++k) if (m%k == 0) {
      int cnt = 0;
      while (m%k == 0) m /= k, cnt++;
      P.pb(k), A.pb(cnt);
    }
```

```cpp
    if (m > 1) P.pb(m), A.pb(1);
  }
  inline void norm(ll &x, ll p) { x = (x%p + p) % p; }
  inline ll get_phi(ll p) {
    ll phi = p;
    //    for (auto t:P) phi=phi/t*(t-1);
    rep(i, 0, sz(P)) phi = phi / P[i] * (P[i] - 1);
    return phi;
  }
  inline bool check(ll m) {
    //if (m==1 || m==2 || m==4) return 1;
    factor(m, P, A);
    if (sz(P) > 2 || sz(P) == 1 && P[0] == 2) return 0;
    if (sz(P) == 1) return 1;
    if (P[0] != 2 || P[0] == 2 && A[0] > 1) return 0;
    return 1;
  }
  inline ll getRoot(ll p) {
    if (p == 1 || p == 2 || p == 4) return phi = p + 1 >> 1, phi_phi = 1, p - 1;
    if (!check(p)) return -1;
    phi = get_phi(p);
    factor(phi, P, A), phi_phi = get_phi(phi);
    //       for (auto &t:P) t=phi/t;
    rep(i, 0, sz(P)) P[i] = phi / P[i];
    for (g = 1; __gcd(g, p) != 1 || !check_g(g, p); ++g);
    return g;
  }
  // solve equation: ax=b(%p), gcd(a,p)!=1
  pll solve(ll a, ll b, ll p) {
    norm(a, p); norm(b, p); ll g = __gcd(a, p);
    if (b%g) return mp(-1, g);
    a /= g, b /= g, p /= g;
    return mp(kpow(a, phi_phi - 1, p)*b%p, g);//note that phi_phi
  }
  ll get_pow(ll p, int k) {
    ll ret = 1; assert(k >= 0);
    rep(i, 0, k) ret = ret*p;
    return ret;
  }
  // solve equation: x^a=b(%pp^k), pp is a prime
  pll solve_high(ll a, ll b, ll pp, int k) {
    assert(pp > 1), assert(k > 0);
    ll p = get_pow(pp, k); norm(b, p); ll t1, t2, t3;
    if (!a) return b == 1 ? mp(0, p) : mp(-1, 0ll);
    if (!b) return mp(!a, get_pow(pp, k - (k - 1) / a - 1));
    ll g = getRoot(p);
    if (g == -1) return mp(-1, 0);
    int cnt = 0; while (b%pp == 0) b /= pp, cnt++;
    if (cnt%a) return mp(-1, 0); bool ok = 0;
    if (cnt) t1 = get_pow(pp, cnt), t2 = get_pow(pp, cnt / a),
      t3 = t1 / t2, ok = 1, p /= t1;
    ll _b = T.ex_bsgs(g, b, p);
    if (_b == -1) return mp(-1, 0);
    ll _p = p / pp*(pp - 1);
    pair<ll, ll> t = solve(a, _b, _p);
```

```
      if (t.fi == -1) return mp(-1, 0);
      ll _g = t.se, x = t.fi, ans = kpow(g, x, p), d = kpow(g, _p / _g, p), ret = _g;
      if (ok) ans *= t2, ret *= t3;
      return mp(ans, ret);
   }
   // solve equation: x^a=b(%p), p could not be a prime, but p must have a primitive
   root, that is 8 cannot divide p
   pll solve_high(ll a, ll b, ll p) {
      assert(p > 0); norm(b, p);
      if (p == 1) return mp(0, 1);
      factor(p, _P, _A); int tot = sz(_P); ll ret = 1, ans; pll tmp[32];
      rep(i, 0, tot) {
         tmp[i + 1] = solve_high(a, b, _P[i], _A[i]),
         crt.a[i + 1] = tmp[i + 1].fi,
         crt.mod[i + 1] = get_pow(_P[i], _A[i]),
         ret *= tmp[i + 1].se;
      }
      if (!ret) return mp(-1, 0);
      ans = crt.solve(tot);
      return mp(ans, ret);
   }
};
// 注: 返回 pair( 最小非负解 , [0,p) 中解的个数 )
```

7.34 高次同余_质数

```
ll kpow(ll a, ll b, ll P) {
   ll r = 1; assert(b >= 0);
   for (; b; b >>= 1, a = a * a % P) if (b & 1) r = r * a % P;
   return r;
}
void ex_gcd(ll a, ll b, ll &x, ll &y) {
   b ? (ex_gcd(b, a % b, y, x), y -= a / b * x) : (x = 1, y = 0);
}
inline ll Inv(ll a, ll P) {
   ll x, y; ex_gcd(a, P, x, y);
   return x < 0 ? x + P : x;
}
struct BSGS {
   map<ll, int> M;
   ll bsgs(ll x, ll z, ll P) {
      if (x % P == 0) return -1;
      ll res = z % P, sa, t = 1, sq = sqrt(P); M.clear();
      rep(i, 0, sq + 1) { if (M.count(t)) break; M[t] = i, t = t * x % P; }
      t = P / sq, sa = Inv(kpow(x, sq, P), P);
      rep(i, 0, t + 1) if (M.count(res))
         return i * sq + M[res]; else res = res * sa % P;
      return -1;
   }
   ll ex_bsgs(ll x, ll z, ll P) { //x^y==z(mod P)
      ll t = 1 % P, w = 1, ans, c = 0; z %= P;
      rep(i, 0, 51) { if (t == z) return i; t = t * x % P; }
      for (t = __gcd(x, P); t != 1; t = __gcd(x, P)) {
         if (z % t) return -1;
         z /= t, P /= t, w = w * x / t % P, c++;
         if (z == w) return c;
      }
      z = z * Inv(w, P) % P, ans = bsgs(x, z, P);
      return ans + (ans != -1) * c;
   }
};

typedef vector<ll> vll;
typedef pair<ll, ll> pll;
struct Euler {
   vll P, A; ll phi, g, phi_phi; BSGS T;
   inline bool check_g(ll g, ll p) {
      rep(i, 0, sz(P)) if (kpow(g, P[i], p) == 1) return 0;
      return 1;
   }
   inline void factor(ll m) {
      P.clear(), A.clear();
      for (ll k = 2; k*k <= m; ++k) if (m%k == 0) {
         int cnt = 0;
         while (m%k == 0) m /= k, cnt++;
         P.pb(k), A.pb(cnt);
      }
      if (m > 1) P.pb(m), A.pb(1);
   }
   inline void norm(ll &x, ll p) { x = (x%p + p) % p; }
   inline ll get_phi(ll p) {
      ll phi = p;
      for (auto t : P) phi = phi / t*(t - 1);
      //    rep(i,0,sz(P)) phi=phi/P[i]*(P[i]-1);
      return phi;
   }
   inline bool check(ll m) {
      //if (m==1 || m==2 || m==4) return 1;
      factor(m);
      if (sz(P) > 2 || sz(P) == 1 && P[0] == 2) return 0;
      if (sz(P) == 1) return 1;
      if (P[0] != 2 || P[0] == 2 && A[0] > 1) return 0;
      return 1;
   }
   inline ll getRoot(ll p) {
      if (p == 1 || p == 2 || p == 4) return phi = p + 1 >> 1, phi_phi = 1, p - 1;
      if (!check(p)) return -1;
      phi = p - 1;
      factor(phi), phi_phi = get_phi(phi);
      for (auto &t : P) t = phi / t;
      //    rep(i,0,sz(P)) P[i]=phi/P[i];
      for (g = 1; __gcd(g, p) != 1 || !check_g(g, p); ++g);
      return g;
   }
   // solve equation: ax=b(%p), gcd(a,p)!=1
   pll solve(ll a, ll b, ll p) {
      norm(a, p); norm(b, p); ll g = __gcd(a, p);
      if (b%g) return mp(-1, g);
      a /= g, b /= g, p /= g;
      return mp(kpow(a, phi_phi - 1, p)*b%p, g);//note that phi_phi
```

```
    }
    // solve equation: x^a=b(%p), p must be a prime
    vll solve_high(ll a, ll b, ll p) {
        vll ret; norm(b, p); assert(p > 0);
        if (!a == b) ret.pb(0);
        if (!b) return ret;
        ll g = getRoot(p);
        if (g == -1) return ret;
        ll _b = T.bsgs(g, b, p);
        if (_b == -1) return ret;
        ll _p = p - 1;
        pll t = solve(a, _b, _p);
        if (t.fi == -1) return ret;
        ll _g = t.se, x = t.fi, ans = kpow(g, x, p), d = kpow(g, _p / _g, p);
        ret.pb(ans);
        rep(i, 1, _g) ans = ans*d%p, ret.pb(ans);
        sort(all(ret));
        return ret;
    }
};
// 注 : 返回所有 [0,p) 中的非负整数解
```

8 Others

8.1 BitOperation

```
// 枚举子集
for(int i = x; i; (--i) & x) {
    //
}
// 统计子集的答案
rep(i, 0, n) {
    rep(j, 0, 1 << n) if(j >> i & 1) {
        upd(s[j], s[j ^ (1 << i)]);
    }
}
// 统计超集的答案
rep(i, 0, n) {
    for(int j = (1 << n) - 1; ~j; --j) if(!(j >> i & 1)) {
        upd(s[j], s[j | (1 << i)]);
    }
}
//
int __builtin_ffs (unsigned int x)
int __builtin_ffsl (unsigned long)
int __builtin_ffsll (unsigned long long)
Returns one plus the index of the least significant 1-bit of x, or if x is zero, returns
 zero.
//
int __builtin_clz (unsigned int x)
Returns the number of leading 0-bits in x, starting at the most significant bit position
. If x is 0, the result is undefined.
//
int __builtin_ctz (unsigned int x)
Returns the number of trailing 0-bits in x, starting at the least significant bit
position. If x is 0, the result is undefined.
//
int __builtin_popcount (unsigned int x)
Returns the number of 1-bits in x.
//
int __builtin_parity (unsigned int x)
Returns the parity of x, i.e. the number of 1-bits in x modulo 2.
```

8.2 Bitset

```
// Base
    b.any();   // has 1 ?
    b.none();  // all 0 ?
    b.count(); // cnt of 1

    b.set();   // all to 1
    b.reset(); // all to 0
    b.flip();  // all = 0 <-> 1

    b.set(p);  // b[p] = 1
    b.test(p); // b[p] is 1
    b.reset(p);// b[p] = 0
    b.flip(p); // b[p] = 0 <-> 1

// Black tech
    // __builtin_ctz in bitst
    b._Find_first();

    // travel all 1
    for (int i = b._Find_first(); i < sz(b); i = b._Find_next(i));
```

8.3 ExpressionParse

```
// 二元运算左结合
vector<char> rpn, ch, sta;
// 定义运算符优先级
int pri(char ch) {
    if(ch == '(') return 0;
    // ...
    return -1;
}
char solve(string s) {
    // 中缀转后缀
    rpn.clear(); ch.clear();
    rep(i, 0, sz(s)) {
        char c = s[i];
        if(c == '(') { ch.pb(c);
        } else if(c == ')') {
            while(ch.back() != '(') rpn.pb(ch.back()), ch.pop_back();
            ch.pop_back();
        } else if(pri(c) > 0) {
            while(sz(ch) && pri(ch.back()) >= pri(c)) rpn.pb(ch.back()), ch.pop_back();
            ch.pb(c);
```

```
      } else { rpn.pb(c); }
   }
   reverse(all(ch)); rpn.insert(rpn.end(), all(ch));
   // 后缀表达式计算
   sta.clear();
   rep(i, 0, sz(rpn)) {
      char u = rpn[i];
      if(pri(u) > 0) {
         char b = sta.back(); sta.pop_back();
         sta[sz(sta) - 1] = calc(u, sta.back(), b);
      } else { sta.pb(u); }
   }
   return sta[0];
}
```

8.4 FastIO

```
// read untill EOF (xint)
struct FastIO {
    static const int S = 1310720;
    int wpos;
    char wbuf[S];
    bool ed;
    FastIO() : wpos(0), ed(0) { }
    inline int xchar() {
        static char buf[S];
        static int len = 0, pos = 0;
        if (pos == len) pos = 0, len = fread(buf, 1, S, stdin);
        if (pos == len) return -1;
        return buf[pos++];
    }
    inline int xint() {
        int c = xchar(), x = 0, s = 1;
        while (c <= 32) {
            if(!~c) return ed = 1;
            c = xchar();
        }
        if (c == '-') s = -1, c = xchar();
        for (; '0' <= c && c <= '9'; c = xchar()) x = x * 10 + c - '0';
        return x * s;
    }
    inline int xuint() {
        int c = xchar(), x = 0;
        while (c <= 32) c = xchar();
        for (; '0' <= c && c <= '9'; c = xchar()) x = x * 10 + c - '0';
        return x;
    }
    inline void xstring(char *s) {
        int c = xchar();
        while (c <= 32) c = xchar();
        for (; c >  32; c = xchar()) *s++ = c;
        *s = 0;
    }
    inline void wchar(int x) {
        if (wpos == S) fwrite(wbuf, 1, S, stdout), wpos = 0;
        wbuf[wpos++] = x;
    }
    inline void wint(int x) {
        if (x < 0) wchar('-'), x = -x;
        char s[24];
        int n = 0;
        while (x || !n) s[n++] = '0' + x % 10, x /= 10;
        while (n--) wchar(s[n]);
    }
    inline void wstring(const char *s) { while (*s) wchar(*s++); }
    ~FastIO() { if (wpos) fwrite(wbuf, 1, wpos, stdout), wpos = 0; }
} io;
```

8.5 FastMod

```
template<class T1, class T2>
struct FastD {
   const static int wb = sizeof(T1) * 8;
   int len; T1 m, x;
   FastD() = default;
   FastD(T1 n): m(n) {
      if (n == 1) x = 1, len = 0;
      else {
         if (wb == 32) len = 31 - __builtin_clz(n - 1) + wb;
         else len = 63 - __builtin_clzll(n - 1) + wb;
         x = ((T2(1) << len) + n - 1) / n;
      }
   }
   friend T1 operator / (const T1 &n, const FastD &d) { return T2(n) * d.x >> d.len; }
   friend T1 operator % (const T1 &n, const FastD &d) { return n - n / d * d.m; }
};

template<class T> // 只能用于奇数
struct ExactD {
   T t, i;
   ExactD() = default;
   ExactD(const T &n): t(T(-1) / n), i(mul_inv(n)) {}
   constexpr static T mul_inv(T n, int e = 6, T x = 1) {
      return !e ? x : mul_inv(n, e - 1, x * (2 - x * n));
   }
   friend T operator / (const T &n, const ExactD &d) { return n * d.i; }
   bool divide(const T &n) const { return n * i <= t; }
};

using FastDiv32 = FastD<uint32, uint64>;
using FastDiv64 = FastD<uint64, uint128>;
using ExactDiv32 = ExactD<uint32>;
using ExactDiv64 = ExactD<uint64>;
```

8.6 Rand

```
mt19937 gen(998244353);
typedef uniform_int_distribution<ll> RR;
ll rnd(ll l, ll r) { RR dis(l, r); return dis(gen); }
```

```
typedef uniform_real_distribution<db> RR;
db rnd(db l, db r) { RR dis(l, r); return dis(gen); }
```

8.7 RomanNumerals

```
const int rom[30] = {
   3000,2000,1000,900,800,700,600,500,400,300,200,100,
   90,80,70,60,50,40,30,20,10,
   9,8,7,6,5,4,3,2,1
};
string smb[30]={
   "MMM","MM","M",
   "CM","DCCC","DCC","DC","D","CD","CCC","CC","C",
   "XC","LXXX","LXX","LX","L","XL","XXX","XX","X",
   "IX","VIII","VII","VI","V","IV","III","II","I"
};
string toRoman(ll d) {
   string r;
   rep(i, 0, 30) if (d >= rom[i]) d -= rom[i], r += smb[i];
   return r;
}
```

8.8 Strtok

```
char s[111]; gets(s); vector<string> a;
for(char* p=strtok(s," .,()");p;p=strtok(NULL," .,()")) a.pb(p);
```

8.9 Time

```
clock_t st = clock(); CLOCKS_PER_SEC;
```

8.10 duipai

```
#!/bin/bash
while true; do
    ./gen > gen.in
    ./sol <gen.in >sol.out
    ./dp <gen.in >dp.out
    if diff sol.out dp.out; then
        printf "AC\n"
    else
        printf "Wa\n"
        exit 0
    fi
done

// sh duipai.sh
```

9 String

9.1 ACAutomaton

```
/*
 * [0,L) , N-1 is virtual , 0 is rt
 * init!!
 * addition: end[] end[c] |= end[fail[c]]
 */
struct Trie{
    static const int N = 101010 , M = 26;
    int ne[N][M] , fail[N] , fa[N] , rt , L;
    void ini(){ fill_n(ne[fail[0] = N-1],M,0);L = 0;rt = newnode();}
    int newnode(){ fill_n(ne[L],M,0); return L++; }
    void add(char *s){
        int p = rt;
        for(int i=0;s[i];++i){
            int c = s[i] - 'a';// modify
            if(!ne[p][c]) ne[p][c] = newnode() , fa[L-1] = p;
            p = ne[p][c];
        }
    }
    void Build(){
        vi v;v.pb(rt);
        rep(i,0,sz(v)){
            int c = v[i];
            rep(i,0,M) ne[c][i] ?
                v.pb(ne[c][i]) , fail[ne[c][i]] = ne[fail[c]][i] :
                ne[c][i] = ne[fail[c]][i];
        }
    }
};
```

9.2 DoublingArray

```
namespace Doubling{
   static const int N = 101010;
   // sa[0~n]: 排名第的后缀是以i sa[i] 开头
   // h[1~n]:S[sa[i-1]] 与 S[sa[i]] 的最长公共前缀长度为 h[i]
   int t[N] , wa[N] , wb[N] , sa[N] , h[N];
   void sort(int *x,int *y,int n,int m){
      rep(i,0,m) t[i] = 0;
      rep(i,0,n) t[x[y[i]]]++;
      rep(i,1,m) t[i] += t[i-1];
      per(i,0,n) sa[--t[x[y[i]]]] = y[i];
   }
   bool cmp(int *x,int a,int b,int d){ return x[a] == x[b] && x[a+d] == x[b+d]; }
   void da(int *s,int n,int m){
      int *x=wa,*y=wb;
      rep(i,0,n) x[i] = s[i] , y[i] = i;
      sort(x , y , n , m);
      for(int j=1,p=1;p<n;m=p,j<<=1){
         p = 0;rep(i,n-j,n) y[p++] = i;
         rep(i,0,n) if(sa[i] >= j) y[p++] = sa[i] - j;
         sort(x , y , n , m);
         swap(x , y);p = 1;x[sa[0]] = 0;
         rep(i,1,n) x[sa[i]] = cmp(y,sa[i],sa[i-1],j)?p-1:p++;
      }
   }
   void cal_h(int *s,int n,int *rk){
      int j,k=0;
      for(int i=1;i<=n;++i) rk[sa[i]] = i;
      for(int i=0;i<n;h[rk[i++]] = k) for(k&&--k,j=sa[rk[i]-1];s[i+k]==s[j+k];++k);
   }
}
// rank[0~n-1]: 以 i 开头的后缀排名 rank[i]
struct DA{ // [0,n] , in[n] = 0 , n load
   static const int N = 101010;
   int p[18][N] , rk[N] , in[N] , Log[N] , n;
   void Build(){
      Doubling::da(in,n+1,300);
      Doubling::cal_h(in,n,rk);
      Log[0] = -1;for(int i=1;i<=n;++i) Log[i] = Log[i-1] + (i==(i&(-i)));
      for(int i=1;i<=n;++i) p[0][i] = Doubling::h[i];
      for(int j=1;1<<j<=n;++j){
         int lim = n+1-(1<<j);
         for(int i=1;i<=lim;++i) p[j][i] = min(p[j-1][i] , p[j-1][i+(1<<j>>1)]);
      }
   }
   // 某两个后缀的最长公共前缀
   int lcp(int a,int b){
      a = rk[a] , b = rk[b];
      if(a > b) swap(a , b);++a;
      int t = Log[b-a+1];
      return min(p[t][a] , p[t][b-(1<<t)+1]);
   }
};
```

9.3 Exkmp

```
/*
 * s 串的每个后缀与 t 串的最长公共前缀
 * t:  a  b  a
 * nt: 0  0  1
 * s:  a  b  a  c  a  b  a
 * ns: 3  0  1  0  3  0  1
 */
void exkmp(char *s,int *z,char *t,int *p){
    int lens = strlen(s);
    int lent = strlen(t);
    p[0]=0;
    for(int i=0,x=0,y=0;i<lens;++i){
        z[i] = i <= y ? min(y-i,p[i-x]) : 0;
        while(i + z[i] < lens && z[i] < lent && s[i + z[i]] == t[z[i]]) ++z[i];
        if(y <= i + z[i]) x = i, y = i + z[i];
    }
}

void Exkmp(){
    scanf("%s%s",s,t);
    exkmp(t+1,nt+1,t,nt);
    exkmp(s,ns,t,nt);
}
```

9.4 Kmp

```
/*
t:  a  b  a
nt:-1 -1  0
s:  a  b  a  c  a  b  a
ns: 0  1  2 -1  0  1  2
*/
void kmp(char *s,int *ns,char *t,int *nt){
    int lens = strlen(s);
    int lent = strlen(t);
    nt[0] = -1;
    for(int i=0,j=-1;i<lens;++i){
        while(j >= 0 && s[i] != t[j + 1]) j = nt[j];
        if(s[i] == t[j + 1]) ++j;
        ns[i] = j;
        if(j + 1 == lent) j = nt[j];
    }
}
void KMP(){
    scanf("%s%s",s,t);
    kmp(t+1,nt+1,t,nt);
    kmp(s,ns,t,nt);
}
```

9.5 LyndonWord

```
// O(n) 分解为字典序非严格降的 lyndon word 分解唯一
vector<int> duval(char s[]){
  vector<int> ret;
  int n = strlen(s) + 1; // zero used here
  int start = 0, mid = 1, cur = 0;
  ret.push_back(0);
  for (int i = 0; i < n; ++i){
    if (s[i] == s[cur]){
      if (++cur == mid) cur = start;
    } else if (s[i] > s[cur]){
      mid = i + 1;
      cur = start;
    } else if (s[i] < s[cur]){
      int temp = mid - start;
      while (start + temp <= i){
        start += temp;
        ret.push_back(start);
      }
      i = cur = start;
      mid = start + 1;
    }
  }
  return ret;
}
/*
```

```
cbaabc
0 1 2 6
*/

// 生成字符集为 m , 长度不超过 n 的所有 lyndon word , 字符集从 a 开始
void lyndon_generate(int n, int m) {
  char z = 'a' + m - 1, s[1000];
  s[0] = 'a' - 1;
  for (int i = 1, x = 1; ; ++i) {
    s[x - 1]++; s[x] = 0;
    puts(s);
    if (strlen(s)==1 && s[0]=='a'+m-1) return;
    for (int j = x; j < n; ++j) s[j] = s[j - x];
    for (x = n; s[x - 1] == z; --x);
  }
}
/*
3 2
a
aab
ab
abb
b
*/
```

9.6 Manacher

```
/*
 * length of pa is two size of str
 * i: [0, n)     pa[i<<1] : odd string 整个回文长度为 2*pa[i<<1]-1
 * i: [0, n - 1) pa[i<<1|1] : even string 整个回文长度为 2*pa[i<<1]
 * N>2*n
*/
void Manacher(char *s,int n,int *pa){
    pa[0] = 1;
    for(int i=1,j=0;i<(n<<1)-1;++i){
        int p = i >> 1 , q = i - p , r = ((j + 1)>>1) + pa[j] - 1;
        pa[i] = r < q ? 0 : min(r - q + 1 , pa[(j<<1) - i]);
        while(0 <= p - pa[i] && q + pa[i] < n && s[p - pa[i]] == s[q + pa[i]]) pa[i]++;
        if(q + pa[i] - 1 > r) j = i;
    }
}
```

9.7 PAM

```
// [0,p) , 0(even) and 1(odd) is virtual , init!!
struct Palindromic_Tree {
   static const int N = ::N , M = 26;
   int ne[N][M] , fail[N] , len[N] , S[N] , last , n , p, cnt[N], las[N];
   int newnode(int l){
      fill(ne[p] , ne[p] + M , 0);
      len[p] = l;
      las[p] = n;
      cnt[p] = 0;
```

```
      return p++;
   }
   void ini(){
      p = 0;newnode(0);newnode(-1);
      S[n = last = 0] = -1;
      fail[0] = 1;
   }
   int get_fail(int x){
      while(S[n - len[x] - 1] != S[n]) x = fail[x];
      return x;
   }
   void add(int c){
      S[++n] = c;
      int cur = get_fail(last);
      if(!ne[cur][c]){
         int now = newnode(len[cur] + 2);
         fail[now] = ne[get_fail(fail[cur])][c];
         ne[cur][c] = now;
      }
      last = ne[cur][c];
      cnt[last]++;
   }
   void build() { for(int i = p - 1; ~i; --i) cnt[fail[i]] += cnt[i]; }
}pam;
```

9.8 SAIS

```
/*
 * Ensure that str[n] is the unique lexicographically smallest character in str.
 * time complexity: O(n)
 */
namespace SA {
   const static int N = 100000 + 10;
   int sa[N], rk[N], ht[N], s[N << 1], t[N << 1], p[N], cnt[N], cur[N];
#define pushS(x) sa[cur[s[x]]--] = x
#define pushL(x) sa[cur[s[x]]++] = x
#define inducedSort(v) std::fill_n(sa, n, -1); std::fill_n(cnt, m, 0);      \
   for (int i = 0; i < n; i++) cnt[s[i]]++;                                  \
   for (int i = 1; i < m; i++) cnt[i] += cnt[i-1];                           \
   for (int i = 0; i < m; i++) cur[i] = cnt[i]-1;                            \
   for (int i = n1-1; ~i; i--) pushS(v[i]);                                  \
   for (int i = 1; i < m; i++) cur[i] = cnt[i-1];                            \
   for (int i = 0; i < n; i++) if (sa[i] > 0 &&  t[sa[i]-1]) pushL(sa[i]-1); \
   for (int i = 0; i < m; i++) cur[i] = cnt[i]-1;                            \
   for (int i = n-1;  ~i; i--) if (sa[i] > 0 && !t[sa[i]-1]) pushS(sa[i]-1)
   void sais(int n, int m, int *s, int *t, int *p) {
      int n1 = t[n - 1] = 0, ch = rk[0] = -1, *s1 = s + n;
      for (int i = n - 2; ~i; i--) t[i] = s[i] == s[i + 1] ? t[i + 1] : s[i] > s[i + 1];
      for (int i = 1; i < n; i++) rk[i] = t[i - 1] && !t[i] ? (p[n1] = i, n1++) : -1;
      inducedSort(p);
      for (int i = 0, x, y; i < n; i++) if (~(x = rk[sa[i]])) {
         if (ch < 1 || p[x + 1] - p[x] != p[y + 1] - p[y]) ch++;
         else for (int j = p[x], k = p[y]; j <= p[x + 1]; j++, k++)
            if ((s[j] << 1 | t[j]) != (s[k] << 1 | t[k])) { ch++; break; }
         s1[y = x] = ch;
```

```
        }
        if (ch + 1 < n1) sais(n1, ch + 1, s1, t + n, p + n1);
        else for (int i = 0; i < n1; i++) sa[s1[i]] = i;
        for (int i = 0; i < n1; i++) s1[i] = p[sa[i]];
        inducedSort(s1);
    }
    template<typename T>
    int mapCharToInt(int n, const T *str) {
        int m = *max_element(str, str + n);
        std::fill_n(rk, m + 1, 0);
        for (int i = 0; i < n; i++) rk[str[i]] = 1;
        for (int i = 0; i < m; i++) rk[i + 1] += rk[i];
        for (int i = 0; i < n; i++) s[i] = rk[str[i]] - 1;
        return rk[m];
    }
    template<typename T>
    void suffixArray(int n, const T *str) {
        int m = mapCharToInt(++n, str);
        sais(n, m, s, t, p);
        for (int i = 0; i < n; i++) rk[sa[i]] = i;
        for (int i = 0, h = ht[0] = 0; i < n - 1; i++) {
            int j = sa[rk[i] - 1];
            while (i + h < n && j + h < n && s[i + h] == s[j + h]) h++;
            if (ht[rk[i]] = h) h--;
        }
    }
};
```

9.9 SAM

```
/*
 * [0,L] , 0 is virtual , 1 is rt , init!!
 * [l[par[s]] + 1, l[s]]
 */
struct SAM {
    static const int N = ::N << 1, M = 26;
    int par[N], l[N], ne[N][M], rt, last, L;
    void add(int c) {
        int p = last;
        /* ex
        if(ne[p][c] && l[ne[p][c]] == l[p] + 1) { last = ne[p][c]; return ; }
        */
        int np = ++L;
        fill(ne[np], ne[np] + M, 0);
        l[np] = l[p] + 1;
        last = np;
        while(p && !ne[p][c]) ne[p][c] = np, p = par[p];
        if(!p) par[np] = rt;
        else{
            int q = ne[p][c];
            if(l[q] == l[p] + 1) par[np] = q;
            else {
                int nq = ++L;
                l[nq] = l[p] + 1;
                copy(ne[q], ne[q] + M, ne[nq]);
                par[nq] = par[q];
                par[q] = par[np] = nq;
                while(p && ne[p][c] == q) ne[p][c] = nq, p = par[p];
            }
        }
    }
    void ini() {
        rt = last = L = 1;
        fill(ne[rt], ne[rt] + M, 0);
        l[0] = -1;
    }
};
// BucketSort
rep(i, 1, L + 1) ++cnt[l[i]];
rep(i, 1, L + 1) cnt[i] += cnt[i - 1];
rep(i, 1, L + 1) cur[cnt[l[i]]--] = i;
```

9.10 SA_trie

```
// trie 树点带字母，每个点到根的字符串排序， O(nlogn)
// C 为字符集大小，从 a 开始， M 为倍增深度
// 调用 Init 之后，取 sa[]
const int N = 5e5, M = 21, C = 26;
int n, fa[N]; char s[N];
struct SA {
    int R[N], RF[N], tmp[N], pos[N], tax[N], tp[N], sa[N], siz, n, pa[N][M];
    int h(int c) { return c - 'a' + 1; }
    void Qsort(int *sa, int *R, int *tp, int siz) {
        rep(i, 0, siz + 1) tax[i] = 0;
        rep(i, 1, n + 1) tax[R[tp[i]]]++;
        rep(i, 1, siz + 1) tax[i] += tax[i - 1];
        per(i, 1, n + 1) sa[tax[R[tp[i]]]--] = tp[i];
    }
    // s[] 表示字母点权，下标从 1 开始
    // fa[] 表示树上父节点编号，根为 1
    void Init(int _n, int fa[], char s[]) {
        n = _n, pa[1][0] = 0; rep(i, 2, n + 1) pa[i][0] = fa[i];
        rep(i, 2, n + 1) rep(j, 1, M) pa[i][j] = pa[pa[i][j - 1]][j - 1];
        rep(i, 1, n + 1) R[i] = h(s[i]), tp[i] = i;
        Qsort(sa, R, tp, C); rep(i, 1, n + 1) pos[sa[i]] = i;
        for (int w = 1, p = 0; w < n; w <<= 1, p++) {
            rep(i, 1, n + 1) RF[i] = pos[pa[i][p]];
            Qsort(tp, RF, sa, n);
            Qsort(sa, R, tp, R[sa[n]]);
            rep(i, 1, n + 1) tmp[i] = R[i]; R[sa[1]] = 1;
            rep(i, 2, n + 1) R[sa[i]] =
                (tmp[sa[i]] == tmp[sa[i - 1]] && tmp[pa[sa[i]][p]] == tmp[pa[sa[i - 1]][p]])
                 ?
                  R[sa[i - 1]] : R[sa[i - 1]] + 1;
            rep(i, 1, n + 1) pos[sa[i]] = i;
        }
    }
} T;
```

9.11 StrHash

```
struct STR {
   ll _B[N], h[N], base, P;
   inline void init_h(char st[], int n, ll _base = 163, ll _P = 1e9 + 7, char c = 'a') {
      base = _base, P = _P, h[0] = st[0] - c;
      rep(i, 1, n) h[i] = (h[i - 1] * base + st[i] - c) % P;
      if (_B[0] == 1) return; _B[0] = 1;
      rep(i, 1, N) _B[i] = _B[i - 1] * base %P;
   }
   inline ll H(int l, int r) {
      if (!l) return h[r] < 0 ? h[r] + P : h[r];
      ll ans = (h[r] - h[l - 1] * _B[r - l + 1]) % P;
      return ans < 0 ? ans + P : ans;
   }
};
```

9.12 StrHash_双哈希

```
const int P = 1e9 + 7;
inline int upd(int a, int b) {
   if((a += b) >= P) a -= P;
   return a < 0 ? a + P : a;
}
inline int mul(int a, int b) {return 1ll * a * b % P; }
struct Int{
   int a, b;
   Int(int a = 0, int b = 0) : a(a), b(b) {}
   inline Int operator + (const Int &c) const { return Int(upd(a, c.a), upd(b, c.b)); }
   inline Int operator - (const Int &c) const { return Int(upd(a, -c.a), upd(b, -c.b));
    }
   inline Int operator * (const Int &c) const { return Int(mul(a, c.a), mul(b, c.b)); }
   inline bool operator == (const Int &c) const {return a == c.a && b == c.b;}
} _0 = Int(), _1 = Int(1, 1), B[N];
void init(int n){
   B[0] = _1; B[1] = Int(233, 241);
   rep(i, 2, n+1) B[i] = B[i-1] * B[1];
}
struct Str{
   Int a; int len;
   Str(Int a = _0, int len = 0) : a(a), len(len) {}
   Str(int x) {a = Int(x, x); len = 1;}
   inline Str operator + (const Str &c) const { return Str(a * B[c.len] + c.a, len + c.
   len); }
   // 减去一个前缀
   inline Str operator - (const Str &c) const { return Str(a - c.a * B[len - c.len], len
    - c.len); }
   inline bool operator == (const Str &c) const { return a == c.a && len == c.len;}
} ha[N], hb[N];
void init(vi &s, Str *ha) {
   rep(i, 0, sz(s)) ha[i] = i > 0 ? ha[i-1] + Str(s[i] + 1) : Str(s[0] + 1);
}
Str sub(Str *ha, int l, int r) {
   if (l > r) return Str();
   return l > 0 ? ha[r] - ha[l-1] : ha[r];
}
```

9.13 最小表示法

```
// 下标从 0 开始
// s[] 开两倍长度
int MINR(char s[],int L){
   rep(i,0,L) s[L+i]=s[i]; s[2*L]=0;
   int i=0, j=1;
   while(i<L && j<L){
      int k=0;
      while(s[i+k]==s[j+k] && k<L)++k;
      if(k==L)return min(i,j);
      // 最大改成 <
      if(s[i+k]>s[j+k])i=max(i+k+1,j+1);
      else j=max(j+k+1,i+1);
   }
   return min(i,j);
}
```

10 Tree

10.1 DsuOnTree

```
// id starts with 1
namespace QuerySubtree{
   static const int N = ::N;
   int sz[N] , wson[N] , par[N];
   void dfs(int c,int fa,vi g[]){
      sz[c]=1;par[c]=fa;int &s=wson[c]=0;
      for(auto t:g[c]) if(t!=fa) dfs(t,c,g),sz[c]+=sz[t],(sz[t]>=sz[s])&&(s=t);
   }
   void solve(int c,int fa,bool iswson,vi g[]){
      for(auto t : g[c]) if(t != wson[c] && t != fa) solve(t , c , false , g);
      if(wson[c]) solve(wson[c] , c , true , g);
      for(auto t : g[c]) if(t != wson[c] && t != fa) {
         // 将该子树的信息加入
      }
      // 将当前节点的信息加入
      // 查询
      if(!iswson) {
         // 如果当前子树是轻儿子，删除这棵子树的信息
      }
   }
   void solve(vi g[]){
      dfs(1,0,g);
      solve(1,0,false,g); // 如果输入是单组数据，改成 true 可以优化常数
   }
}
```

10.2 HeavyChain

```
// id starts with 1
struct HeavyChain{
   static const int N = ::N;
   int sz[N], wson[N], top[N], dep[N], id[N], _, par[N], who[N];
   void dfs(int c, int fa, vi g[]){
      sz[c] = 1;
      par[c] = fa;
      dep[c] = dep[fa] + 1;
      int &s = wson[c] = top[c] = 0;
      for(auto t : g[c]) if(t != fa) {
         dfs(t, c, g);
         sz[c] += sz[t];
         if(sz[t] >= sz[s]) s = t;
      }
   }
   void dfs2(int c, int fa, vi g[]){
      id[c] = ++_;
      who[_] = c;
      int s = wson[c];
      if(!top[c]) top[c] = c;
      if(s) top[s] = top[c], dfs2(s, c, g);
      for(auto t : g[c]) if(t != fa && t != s) dfs2(t, c, g);
   }
   void Query(int a, int b){
      int fa = top[a], fb = top[b];
      while(fa != fb){
         if(dep[fa] < dep[fb]) swap(a, b), swap(fa, fb);
         // Cal id[fa] .. id[a]
         a = par[fa]; fa = top[a];
      }
      if(dep[a] < dep[b]) swap(a, b);
      // Cal id[b] .. id[a]
      // b is lca
   }
   void Build(vi g[]){
      dfs(1, 0, g);
      _=0;
      dfs2(1, 0, g);
   }
}hc;
```

10.3 LCARMQ

```
// N is 2 size of tree , id of nodes start from 1
struct LCARMQ{
    static const int N = 101010 << 1;
    int a[20][N] , lft[N] , dep[N] , lg[N] , L;
    int rmin(int x,int y){return dep[x] < dep[y] ? x : y;}
    void add(int x){ a[0][L++] = x;}
    void dfs(int c,int fa,const vi g[]){
        lft[c]=L;add(c);
        for(auto t : g[c]) if(t!=fa) dep[t]=dep[c]+1,dfs(t,c,g),add(c);
    }
    void Build(const vi g[]){
        L = 0;dfs(1,0,g);dep[0] = -1;
        rep(i,2,L) lg[i]=lg[i>>1]+1;
        rep(i,1,20){
            int lim = L+1-(1<<i);
            rep(j,0,lim) a[i][j] = rmin(a[i-1][j] , a[i-1][j+(1<<i>>1)]);
        }
    }
    int lca(int x,int y){
        x = lft[x] , y = lft[y];
        if(x > y) swap(x , y);
        int i = lg[y-x+1];
        return rmin(a[i][x] , a[i][y+1-(1<<i)]);
    }
};
```

10.4 LongChain

```
struct LongChain{
   static const int N = ::N;
   int wson[N] , top[N] , dep[N] , lg[N];
   int jump[N][20] , id[N] , who[N] , rwho[N] , _;
   void dfs(int c,int fa,vi g[]){
      dep[c]=1;int &s=wson[c]=top[c]=0;
      jump[c][0]=fa;rep(i,1,20) jump[c][i]=jump[jump[c][i-1]][i-1];
      for(auto t:g[c]) if(t!=fa)
         dfs(t,c,g),dep[c]=max(dep[t]+1,dep[c]),(dep[t]>=dep[s])&&(s=t);
   }
   void dfs2(int c,int fa,int rc,vi g[]){
      if(!top[c]) top[c]=c,rc=c;
      who[id[c]=++_]=c;rwho[_]=rc;
      int s=wson[c];
      if(s) top[s]=top[c],dfs2(s,c,jump[rc][0],g);
      for(auto t:g[c]) if(t!=fa&&t!=s) dfs2(t,c,t,g);
   }
   void Build(vi g[]){
      dfs(1,0,g);_=0;dfs2(1,0,1,g);
      rep(i,2,N) lg[i]=lg[i>>1]+1;
   }
   void solve(int c, int fa, vi g[]) {
      for(auto t : g[c]) if(t != fa) solve(t, c, g);
      if(wson[c]) {
         // upd c by wson[c], O(1) or O(log(n))
      } else {
         // c is leaf
      }
      for(auto t : g[c]) if(t != fa && t != wson[c]) {
         // brute force upd c by t
      }
      // 注意统计以 c 为起点的链的答案，注意深度的限制（两棵子树都要注意）
   }
   // kth_par should exist
   int kth_par(int x,int k){
      if(k==0) return x;
      int j0=1<<lg[k];
      int p0=jump[x][lg[k]];
```

```
        int j1=k-j0;
        int del=id[p0]-id[top[p0]];
        if(del>=j1) return who[id[p0]-j1];
        else return rwho[id[top[p0]]+j1-del];
    }
}hc;
```

10.5 MoOnTree_Path

```
// 不带修改莫队
// 带修改莫队：块大小 N^(2/3) 按照 l 所在块， r 所在块， time 排序
namespace MoOnTree {
    const int N = ::N, SZ = sqrt(N), M = 17;
    int cd; // starts from 1
    int dep[N], pre[N][M], st[N], ed[N], dfn[N << 1], B[N << 1], cnt[N];
    struct Node {
        int l, r, id, lca;
        Node(int id, int l, int r, int lca = 0) : id(id), l(l), r(r), lca(lca) {}
        bool operator < (const Node &c) const {
            if(B[l] != B[c.l]) return B[l] < B[c.l];
            return (r < c.r) ^ (B[l] & 1);
        }
    };
    vector<Node> nds;
    void dfs(int u, int fa, vi g[]) {
        dep[u] = dep[fa] + 1;
        pre[u][0] = fa;
        for(int i = 1; i < M && pre[u][i - 1]; ++i) pre[u][i] = pre[pre[u][i - 1]][i - 1];
        dfn[++cd] = u, st[u] = cd;
        for(auto v : g[u]) if(v != fa) dfs(v, u, g);
        dfn[++cd] = u, ed[u] = cd;
    }
    int lca(int x, int y) {
        if(dep[x] > dep[y]) swap(x, y);
        per(i, 0, M) if(dep[pre[y][i]] >= dep[x]) y = pre[y][i];
        per(i, 0, M) if(pre[x][i] != pre[y][i]) x = pre[x][i], y = pre[y][i];
        if(x == y) return x;
        return pre[x][0];
    }
    void adde(int u, int v, int id) {
        if(st[u] > st[v]) swap(u, v);
        int f = lca(u, v);
        if(f == u) { nds.pb(Node(id, st[u], st[v]));
        } else {
            int l = ed[u], r = st[v];
            if(l > r) swap(l, r);
            nds.pb(Node(id, l, r, f));
        }
    }
    // p is index in tree
    void add(int p) { }
    void sub(int p) { }
    void upd(int p, int c) {
        p = dfn[p];
        cnt[p] += c;
        (cnt[p] == 1) ? add(p) : sub(p);
    }
    void solve(vi g[]) {
        rep(i, 0, N << 1) B[i] = i / SZ;
        dfs(1, cd = 0, g);
        // adde(u, v)
        sort(all(nds));
        int l = 1, r = 0;
        for(auto &nd : nds) {
            while(r < nd.r) upd(++r, 1);
            while(l > nd.l) upd(--l, 1);
            while(r > nd.r) upd(r--, -1);
            while(l < nd.l) upd(l++, -1);
            if(nd.lca) upd(st[nd.lca], 1);
            // save ans
            if(nd.lca) upd(st[nd.lca], -1);
        }
    }
}
```

10.6 VTree

```
// sort !
namespace Vtree{
    const int N = ::N << 1;
    int tp[N], _, del[N], cntd, l[N], cntl;
    void solve(vi&v,LCARMQ&R){
        _ = cntd = 0; del[++cntd] = tp[_++] = v[0];
        rep(i, 1, sz(v)){
            int lca = R.lca(tp[_-1] , v[i]);
            cntl = 0; while(_ > 0 && R.dep[lca] < R.dep[tp[_-1]]) l[++cntl] = tp[--_];
            if(_ == 0 || lca != tp[_-1]) del[++cntd] = tp[_++] = lca;
            l[++cntl] = tp[_-1]; del[++cntd] = tp[_++] = v[i];
            rep(i, 2, cntl + 1) {
                int u = l[i], v = l[i - 1];
                // g[u].pb(v);
            }
        }
        per(i, 0, _ - 1) {
            int u = tp[i], v = tp[i + 1];
            // g[u].pb(v);
        }
        rep(i, 1, cntd + 1) {
            // del
        }
    }
}
```

10.7 点分树

```
// id starts from 1
namespace Centriod {
    const int N = ::N;
    bool vis[N]; int sz[N], par[N]; vi G[N];
    void dfssz(int c,int fa,int Sz,int &rt){
        sz[c] = 1;
        for(auto t : g[c]) if(!vis[t]&&t!=fa) dfssz(t,c,Sz,rt) , sz[c]+=sz[t];
        if(!rt && sz[c]*2>Sz) rt=c;
    }
    int dfs(int c){
        int rt=0;dfssz(c,0,0,rt);dfssz(c,0,sz[c],rt=0);
        vis[rt] = true;
        for(auto v : g[rt]) if(!vis[v]) {
            int t = dfs(v);
            G[rt].pb(t);
            par[t] = rt;
        }
        return rt;
    }
    void init() {
        fill_n(G + 1, n, vi());
        fill_n(par + 1, n, 0);
    }
};
```

10.8 点分治

```
// id starts from 1
namespace Centriod {
    const int N = ::N;
    bool vis[N]; int sz[N];
    void dfssz(int c,int fa,int Sz,int &rt){
        sz[c] = 1;
        for(auto t : g[c]) if(!vis[t]&&t!=fa) dfssz(t,c,Sz,rt) , sz[c]+=sz[t];
        if(!rt && sz[c]*2>Sz) rt=c;
    }
    void dfs(int c){
        int rt=0;dfssz(c,0,0,rt);dfssz(c,0,sz[c],rt=0);
        vis[rt] = true;
        /*
         * 注意计算以 rt 为起点的路径、只包含 rt 的路径
         * 注意 v != vis[rt]
         */
        for(auto t : g[rt]) if(!vis[t]) dfs(t);
    }
};
```

10.9 边分树

```
// init
namespace ET {
    const int N = ::N << 1;
    Gra g, T; int L, n, sz[N]; bool vis[N << 1];
    void dfssz(int u, int fa, int Sz, int &rt) {
        sz[u] = 1;
        for(int i = g.hd[u]; ~i; i = g.ne[i]) if(!vis[i] && g.to[i] != fa) {
            int v = g.to[i];
            dfssz(v, u, Sz, rt);
            sz[u] += sz[v];
            if(rt == -1 || max(sz[g.to[rt]], Sz - sz[g.to[rt]]) > max(sz[v], Sz - sz[v]))
            rt = i;
        }
    }
    void init(int n) { fill_n(vis, n << 1, 0); }
    int dfs(int u) {
        int I = 0; dfssz(u, 0, 0, I);
        if(sz[u] == n) { T.init(n); }
        if(sz[u] == 1) return u;
        dfssz(u, 0, sz[u], I = -1);
        vis[I] = vis[I ^ 1] = true;
        int _ = (I >> 1) + 1 + n, st = g.fr[I >> 1 << 1], ed = g.to[I >> 1 << 1];
        T._add(_, dfs(st));
        T._add(_, dfs(ed));
        return _;
    }
    void rebuild(int u, int fa, const Gra &G) {
        if(u == 1) L = n = ::n, g.init(n << 1);
        bool F = 0; int pre = u;
        for(int i = G.hd[u]; ~i; i = G.ne[i]) if(G.to[i] != fa) {
            if(F) {
                if(~G.ne[i]) {
                    g.add(pre, ++n, 0);
                    g.add(n, G.to[i], G.val[i]);
                    pre = n;
                } else {
                    g.add(pre, G.to[i], G.val[i]);
                }
            } else {
                g.add(u, G.to[i], G.val[i]);
                F = 1;
            }
            rebuild(G.to[i], u, G);
        }
    }
}
```