Contents

1 Setup

1.1 Remap Escape

```
setxkbmap -option caps:swapescape   # X11
gsettings set org.gnome.desktop.input-sources xkb-options "['caps:swapescape']"   # Wayland
```

1.2 vimrc

```
se nu rnu cin ts=4 sw=4 | sy on
inoremap {<CR> {<CR>}<Esc>O
nnoremap j gj
nnoremap k gk
colo evening
:bad input.txt
:let @# = 'input.txt'
ca Hash w !cpp -dD -P -fpreprocessed \| tr -d "[:space:]" \| md5sum \| cut -c-6
```

1.3 default code [b7320]

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

#ifdef MIKU
string dbmc = "\033[1;38;2;57;197;187m", dbrs = "\033[0m";
#define debug(x...) cerr << dbmc << "[" << #x << "] : ", dout(x)
void dout() { cerr << dbrs << endl; }
template <typename T, typename ...U>
void dout(T t, U ...u) { cerr << t << (sizeof...(u) ? ", " : ""); dout(u...); }
#else
#define debug(...) 39
#endif

#define fs first
#define sc second
#define FOR(i, j, k) for (int i = j, Z = k; i < Z; i++)
using ll = long long;
typedef pair<int, int> pii;

void miku() {
    int a, b;
    cin >> a >> b;
    debug(a, b);
    cout << "DECO*" << a * b << '\n';
}

int32_t main() {
    cin.tie(0) -> sync_with_stdio(false);
    cin.exceptions(cin.failbit);
    miku();
    return 0;
}
```

2 Graph

2.1 Block Cut Tree [a7ea1]

```
// [0, n): round points, [n, tr.size()): square points
// take care of cases when n=1
struct BlockCutTree { // 0 based
  vector<vector<int>> paths, tr;
  vector<int> idx, low, stk;
  int gid, dfn;
  int n;
  BlockCutTree(int _n) {
    n = _n;
    paths = tr = vector<vector<int>>(n);
    gid = 0;
    idx = low = vector<int>(n, -1);
    stk.clear();
  }
  void add_edge(int a, int b) {
    paths[a].push_back(b);
    paths[b].push_back(a);
  }
  void dfs(int now) {
    idx[now] = low[now] = ++dfn;
    stk.push_back(now);
    for(auto nxt:paths[now]) {
      if (idx[nxt] == -1) {
        dfs(nxt);
        low[now] = min(low[now], low[nxt]);
        if (low[nxt] == idx[now]) {
          tr.push_back({});
          int t = -1;
          do {
            t = stk.back();
            stk.pop_back();
            tr[gid+n].push_back(t);
            tr[t].push_back(gid+n);
          } while(t != nxt);
          tr[now].push_back(gid+n);
          tr[gid+n].push_back(now);
          gid ++;
        }
      }
      else low[now] = min(low[now], idx[nxt]);
    }
    return;
  }
  vector<vector<int>> solve() {
    dfn = 0;
    for(int i = 0;i<n;i++) {
      if (idx[i] == -1) dfs(i);
    }
    return tr;
  }
};
```

2.2 Dinic [9befe]

```
struct Dinic { // 0-indexed
  // watch out for e.f overflow
  struct E { int v, c, f; };
  vector<vector<int>> g;
  vector<int> p, d;
  vector<E> e;
  queue<int> q;
  void init(int n) {
    g.resize(n);
    p.resize(n);
    d.resize(n);
  }
  void ae(int u, int v, int cu, int cv = 0) {
    g[u].push_back(e.size());
    e.push_back(E{v, cu, 0});
    g[v].push_back(e.size());
    e.push_back(E{u, cv, 0});
  }
  bool bfs(int s, int t, int l) {
    fill(d.begin(), d.end(), -1);
    d[s] = 0;
    q.push(s);
    while (q.size()) {
      int u = q.front();
      q.pop();
      for (auto &ei : g[u]) {
        if (!((e[ei].c-e[ei].f) >> (30-l))) continue;
        int v = e[ei].v;
        if (d[v] == -1) {
          d[v] = d[u] + 1;
          q.push(v);
        }
      }
    }
    return d[t] != -1;
  }
  int dfs(int u, int t, int fl) {
    if (u == t) return fl;
    for (int &i = p[u]; i < g[u].size(); i++) {
      int ei = g[u][i];
      if (e[ei].f == e[ei].c || d[e[ei].v] != d[u] + 1) continue;
      if (int re = dfs(e[ei].v, t, min(fl, e[ei].c - e[ei].f))) {
        e[ei].f += re;
        e[ei ^ 1].f -= re;
        return re;
      }
    }
    return 0;
  }
  int flow(int s, int t) {
    int ans = 0;
    for (int l = 0; l < 31; l++) {
      while (bfs(s, t, l)) {
        fill(p.begin(), p.end(), 0);
        while (auto re = dfs(s, t, INT_MAX)) ans += re;
      }
    }
    return ans;
  }
  bool inscut(int k) {
    return d[k] != -1;
  }
};
```

2.3 Dominator Tree [3b89c]

```
struct DominatorTree{
  //1-indexed
  //not reachable from s -> not on tree
  int n;
  vector<vector<int>> G,rG;
  vector<int> pa,dfn,id;
  int dfnCnt;
  vector<int> semi,idom,best;
  vector<vector<int>> ret;
  void init(int _n){
    n=_n;
    G = rG = ret = vector<vector<int>>(n+1);
    pa = dfn = id = vector<int>(n+1,-1);
    dfnCnt = 0;
    semi = idom = best = vector<int>(n+1,-1);
  }
  void add_edge(int u,int v){
    G[u].push_back(v);
    rG[v].push_back(u);
  }
  void dfs(int u){
    id[dfn[u]=++dfnCnt]=u;
    for(auto v:G[u]) if(!dfn[v]){
      dfs(v),pa[dfn[v]]=dfn[u];
    }
  }
  int find(int y,int x){
    if(y<=x)return y;
    int tmp=find(pa[y],x);
    if(semi[best[y]]>semi[best[pa[y]]])
      best[y]=best[pa[y]];
    return pa[y]=tmp;
  }
  void tarjan(int root){
    dfnCnt=0;
    for(int i=1;i<=n;++i){
      dfn[i]=idom[i]=0;
      ret[i].clear();
      best[i]=semi[i]=i;
    }
    dfs(root);
    for(int i=dfnCnt;i>1;--i){
      int u=id[i];
      for(auto v:rG[u]) if(v=dfn[v]){
        find(v,i);
        semi[i]=min(semi[i],semi[best[v]]);
      }
      ret[semi[i]].push_back(i);
      for(auto v:ret[pa[i]]){
        find(v,pa[i]);
        idom[v] = semi[best[v]]==pa[i] ? pa[i] : best[v];
      }
      ret[pa[i]].clear();
    }
    for(int i=2; i<=dfnCnt; ++i){
      if(idom[i]!=semi[i]) idom[i]=idom[idom[i]];
      ret[id[idom[i]]].push_back(id[i]);
    }
  }
  vector<vector<int>> solve(int s){
    tarjan(s);
    return ret;
  }
};
```

2.4 Euler Tour [d857a]

```
struct EulerTour{
  // undirected graph,0-indexed,fails if doesn't exist
  // for directed graph, remove the g[b].push_back(pii(a,id)) line in add_edge
  // returns the order of edges
  vector<vector<pii>> g;
  vector<int> ptr;
  vector<bool> vis;
  vector<int> re;
  int n,ecnt;
  void init(int _n){
    n = _n;
    ecnt = 0;
    g = vector<vector<pii>>(n);
    ptr = vector<int>(n);
  }
  void add_edge(int a,int b,int id = -1){
    if(id == -1)id = ecnt;
    g[a].push_back(pii(b,id));
    g[b].push_back(pii(a,id));
    ecnt++;
  }
  void dfs(int now){
    for(int &i = ptr[now];i<g[now].size();i++){
      auto [to,eid] = g[now][i];
      if(vis[eid]) continue;
      vis[eid] = true;
      dfs(to);
      re.push_back(eid);
    }
    return;
  }
  vector<int> solve(int s){
    re.clear();
    vis = vector<bool>(ecnt,0);
    dfs(s);
    reverse(re.begin(),re.end());
    return re;
  }
};
```

2.5 Gomory Hu [3ab29]

```
// needs dinic
struct GomoryHuTree{//0-indexed
#define pii pair<int,int>
#define tiii tuple<int,int,int>
  vector<tiii> edges;
  vector<vector<pii>> tr;
  vector<int> p;
  int n;
  GomoryHuTree(int _n = 0){
    n = _n;
    p = vector<int>(_n,0);
    tr = vector<vector<pii>>(_n);
  }
  void add_edge(int a,int b,int c){
    edges.push_back(tiii(a,b,c));
  }
  vector<vector<pii>> make_tree(){
    fill(p.begin(),p.end(),0);
    tr = vector<vector<pii>>(n);
    for(int i = 1;i<p.size();i++){
      Dinic din(n);
      for(auto &[a,b,w]:edges){
        din.add_edge(a,b,w,w);
      }
      int w = din.flow(i,p[i]);
      tr[i].push_back(pii(p[i],w));
      tr[p[i]].push_back(pii(i,w));
      for(int j = i+1;j<n;j++){
        if(p[j] == p[i]&&din.inScut(j))p[j] = i;
      }
    }
    return tr;
  }
#undef pii
#undef tiii
};
```

2.6 Incremental SCC [d8b55]

```
struct IncrementalSCC{
#define pii pair<int,int>
#define fs first
#define sc second
#define tiii tuple<int,int,int>
  //if u == v : ans[i] = -1
  //if not connected : ans[i] = m
  //all 0-indexed
  int n;
  vector<int> ans;
  int m;
  vector<tiii> all;
  vector<int> SCC(int n,vector<vector<int>>& paths){
    vector<int> scc_id(n,-1),idx(n,-1),low(n,-1),st;
    int cnt = 0,gcnt = 0;
    function<void(int)> dfs = [&](int now)->void{
      low[now] = idx[now] = cnt++;
      st.push_back(now);
      for(auto nxt:paths[now]){
        if(scc_id[nxt] != -1)continue;
        if(idx[nxt] == -1){
          dfs(nxt);
          low[now] = min(low[now],low[nxt]);
        }
        else{
          low[now] = min(low[now],idx[nxt]);
        }
      }
      if(low[now] == idx[now]){
        int id = -1;
        while(id != now){
          id = st.back();
          st.pop_back();
          scc_id[id] = gcnt;
        }
        gcnt++;
      }
    };
    for(int i = 0;i<n;i++){
      if(scc_id[i] == -1)dfs(i);
    }
    return scc_id;
  }
  vector<int> mapping;
  void dc(int l,int r,vector<tiii> &edges){
    if(l == r){
      for(auto &[id,_,__]:edges)ans[id] = min(ans[id],l);
      return;
    }
    int mid = (l+r)>>1;
    int cnt = 0;
    for(auto &[t,u,v]:edges){
      if(mapping[u] == -1)mapping[u] = cnt++;
      if(mapping[v] == -1)mapping[v] = cnt++;
    }
    n = cnt;
    vector<vector<int>> paths(n);
    vector<int> vv;
    for(auto &[t,u,v]:edges){
      vv.push_back(u);
      vv.push_back(v);
      u = mapping[u],v = mapping[v];
      if(t<=mid)paths[u].push_back(v);
    }
    for(auto &i:vv)mapping[i] = -1;

    auto scc_id = SCC(n,paths);
    vector<tiii> vl,vr;
    for(auto &[t,u,v]:edges){
      if(scc_id[u] == scc_id[v]){
        ans[t] = min(ans[t],mid);
        vl.push_back(tiii(t,u,v));
      }
      else{
        u = scc_id[u],v = scc_id[v];
        vr.push_back(tiii(t,u,v));
      }
    }
    vector<tiii>().swap(edges);
    dc(l,mid,vl);
    dc(mid+1,r,vr);
    return;
  }
  void add_edge(int u,int v){
    all.push_back(tiii(all.size(),u,v));
  }
  vector<tiii> solve(){//[time,u,v]
    m = all.size();
    vector<tiii> ret(m);
    for(auto [t,u,v]:all)ret[t] = tiii(m,u,v);
    for(auto [t,u,v]:all)n = max({n,u,v});
    n++;
    ans = vector<int>(m,m);
    for(auto [t,u,v]:all){
      if(u == v)ans[t] = -1;
    }
    mapping = vector<int>(n,-1);
    dc(0,m,all);
    for(int i = 0;i<m;i++)get<0>(ret[i]) = ans[i];
    return ret;
  }
  IncrementalSCC(){
    ans.clear();
    n = m = 0;
  }
#undef tiii
#undef pii
#undef fs
#undef sc
};
```

2.7 KM [2bf04]

```
// Kuhn-Munkres : Bipartite matching with "maximum" weight
in O(n^3)
// NOTICE THAT match[y] = x
struct KM{
  const static int M = 500; // modify maximum number of
vertices
  int n;
  ll ans = 0;
  // 0-base
  vector<vector<ll>> w; // input weighted edges w[x][y]
  vector<int> match; // match[y] = x
  vector<ll> lx, ly, slack;
  bitset<M> visx, visy; // initialize with all zero

  // abbr
# define forx for(int x=0; x<n; x++)
# define fory for(int y=0; y<n; y++)
# define z match[y]

  bool dfs(int x){
    visx[x] = 1;
    fory{
      if(visy[y]) continue;
      ll d = lx[x]+ly[y]-w[x][y];
      if(!d){
        visy[y] = 1;
        if(z==-1 || (!visx[z] && dfs(z))){
          z = x;
          return 1;
        }
      }
      else if(d<slack[y]) slack[y] = d;
    }
    return 0;
  }

  bool augment(){
    fory if(!visy[y] && !slack[y]){
      visy[y] = 1;
      if(z==-1) return 1;
      else if(!visx[z] && dfs(z)){
        z = -1;
        return 1;
      }
    }
    return 0;
  }

  void relabel(){
    ll d = INT64_MAX;
    fory if(!visy[y]) d = min(d, slack[y]);
    forx if(visx[x]) lx[x] -= d;
    fory{
      if(visy[y]) ly[y] += d;
      else slack[y] -= d;
    }
  }

  KM(vector<vector<ll>> &W): n(W.size()), w(W) { // input
edges' weight
                          //initialize
    slack.resize(n);
    match.assign(n, -1);
    lx.assign(n, INT64_MIN);
    ly.assign(n, 0);
    forx fory lx[x] = max(lx[x], w[x][y]);
    //matching
    forx{
      visx.reset();
      visy.reset();
      visx[x] = 1;
      fory slack[y] = lx[x]+ly[y]-w[x][y];
      while(!augment()) relabel();
      visx.reset();
      visy.reset();
      dfs(x);
    }
    //summing
    forx ans += lx[x];
    fory ans += ly[y];
  }

# undef forx
# undef fory
# undef z
};
```

2.8 Max Clique [0de04]

```
constexpr size_t kN = 150; using bits = bitset<kN>;
#define _all(T) T.begin(),T.end()
struct MaxClique {
  bits G[kN], cs[kN];
  int ans, sol[kN], q, cur[kN], d[kN], n;
  void init(int _n) {
    n = _n;
    for (int i = 0; i < n; ++i) G[i].reset();
  }
  void add_edge(int u, int v) { G[u][v] = G[v][u] = 1; }
  void pre_dfs(vector<int> &v, int i, bits mask) {
    if (i < 4) {
      for (int x : v) d[x] = (int)(G[x] & mask).count();
      sort(_all(v), [&](int x, int y) {
          return d[x] > d[y]; });
    }
    vector<int> c(v.size());
    cs[1].reset(), cs[2].reset();
    int l = max(ans - q + 1, 1), r = 2, tp = 0, k;
    for (int p : v) {
      for (k = 1; (cs[k] & G[p]).any(); ++k);
      if (k >= r) cs[++r].reset();
      cs[k][p] = 1;
      if (k < l) v[tp++] = p;
    }
    for (k = l; k < r; ++k)
      for (auto p = cs[k]._Find_first();
          p < kN; p = cs[k]._Find_next(p))
        v[tp] = (int)p, c[tp] = k, ++tp;
    dfs(v, c, i + 1, mask);
  }
  void dfs(vector<int> &v, vector<int> &c,
      int i, bits mask) {
    while (!v.empty()) {
      int p = v.back(); v.pop_back(); mask[p] = 0;
      if (q + c.back() <= ans) return;
      cur[q++] = p;
      vector<int> nr;
      for (int x : v) if (G[p][x]) nr.push_back(x);
      if (!nr.empty()) pre_dfs(nr, i, mask & G[p]);
      else if (q > ans) ans = q, copy_n(cur, q, sol);
      c.pop_back(); --q;
    }
  }
  int solve() {
    vector<int> v(n); iota(_all(v), 0);
    ans = q = 0; pre_dfs(v, 0, bits(string(n, '1')));
    return ans; // sol[0 ~ ans-1]
  }
};
```

2.9 Minimum Cost Maximum Flow [80eed]

```
#define T ll
const T inf = 1e12;
struct MCMF{//TC:O(VEF)
  struct E{
    int t,f;
    T c,w;
    E(int tt,T cap,T wei):t(tt),c(cap),w(wei),f(0){}
  };
  vector<E> e;
  vector<vector<int>> paths;
  vector<T> dis;
  vector<int> pre;
  vector<bool> inq;
  queue<int> q;
  int n;
  MCMF(int _n = 0){
    n = _n;
    paths = vector<vector<int>>(n);
    e.clear();
    pre = vector<int>(n);
    dis = vector<T>(n);
    inq = vector<bool>(n);
  }
  void add_edge(int a,int b,int c,int d){//from,to,cap,wei
    paths[a].push_back(e.size());
    e.push_back(E(b,c,d));
    paths[b].push_back(e.size());
    e.push_back(E(a,0,-d));
  }
  bool SPFA(int s,int t){
    fill(dis.begin(),dis.end(),inf);
    fill(pre.begin(),pre.end(),-1);
    dis[s] = 0;
    q.push(s);inq[s] = true;
    while(!q.empty()){
      auto now = q.front();q.pop();
      inq[now] = false;
      //assert(dis[now]>=0);
      for(auto &eid:paths[now]){
        if(e[eid].f == e[eid].c)continue;
        int nxt = e[eid].t;
        if(dis[nxt]>dis[now]+e[eid].w){
          pre[nxt] = eid;
          dis[nxt] = dis[now]+e[eid].w;
          if(!inq[nxt]){
            inq[nxt] = true;
            q.push(nxt);
          }
        }
      }
    }
    return dis[t] != inf;
  }
  T flow(int s,int t,int cnt = INT_MAX){//cnt is the number
of flows
    T ans = 0;
    while(cnt--&&SPFA(s,t)){
      ans += dis[t];
      int now = t;
      while(pre[now] != -1){
        int eid = pre[now];
        e[eid].f++;
        e[eid^1].f--;
        now = e[eid^1].t;
      }
    }
    return ans;
  }
};
#undef T
```

2.10 Minimum Cost Maximum Flow [b5b00]

```
struct Matching {
  queue<int> q; int ans, n;
  vector<int> fa, s, v, pre, match;
  int Find(int u) {
    return u == fa[u] ? u : fa[u] = Find(fa[u]); }
  int LCA(int x, int y) {
    static int tk = 0; tk++; x = Find(x); y = Find(y);
    for (;; swap(x, y)) if (x != n) {
      if (v[x] == tk) return x;
      v[x] = tk;
      x = Find(pre[match[x]]);
    }
  }
  void Blossom(int x, int y, int l) {
    for (; Find(x) != l; x = pre[y]) {
      pre[x] = y, y = match[x];
      if (s[y] == 1) q.push(y), s[y] = 0;
      for (int z: {x, y}) if (fa[z] == z) fa[z] = l;
    }
  }
  bool Bfs(auto &&g, int r) {
    iota(fa.begin(), fa.end(), 0); ranges::fill(s, -1);
    q = queue<int>(); q.push(r); s[r] = 0;
    for (; !q.empty(); q.pop()) {
      for (int x = q.front(); int u : g[x])
        if (s[u] == -1) {
          if (pre[u] = x, s[u] = 1, match[u] == n) {
            for (int a = u, b = x, last;
                b != n; a = last, b = pre[a])
              last = match[b], match[b] = a, match[a] = b;
            return true;
          }
          q.push(match[u]); s[match[u]] = 0;
        } else if (!s[u] && Find(u) != Find(x)) {
          int l = LCA(u, x);
          Blossom(x, u, l); Blossom(u, x, l);
        }
    }
    return false;
  }
  Matching(auto &&g) : ans(0), n(int(g.size())),
  fa(n+1), s(n+1), v(n+1), pre(n+1, n), match(n+1, n) {
    for (int x = 0; x < n; ++x)
      if (match[x] == n) ans += Bfs(g, x);
  } // match[x] == n means not matched
}; // test @ yosupo judge
```

3 Data Structure

3.1 Dynamic Convex Hull [98f67]

```
#define ll long long
// only works for integer coordinates!! maintain max

struct Line {
  mutable ll a, b, p;
  bool operator<(const Line &rhs) const { return a < rhs.a; }
  bool operator<(ll x) const { return p < x; }
};
struct CHT : multiset<Line, less<>> {
  static const ll kInf = 1e18;
  ll Div(ll a, ll b) { return a / b - ((a ^ b) < 0 && a % b); }
  bool isect(iterator x, iterator y) {
    if (y == end()) { x->p = kInf; return 0; }
    if (x->a == y->a) x->p = x->b > y->b ? kInf : -kInf;
    else x->p = Div(y->b - x->b, x->a - y->a);
    return x->p >= y->p;
  }
  void addline(ll a, ll b) {
    auto z = insert({a, b, 0}), y = z++, x = y;
    while (isect(y, z)) z = erase(z);
    if (x != begin() && isect(--x, y)) isect(x, y = erase(y));
    while ((y = x) != begin() && (--x)->p >= y->p) isect(x, erase(y));
  }
  ll query(ll x) {
    auto l = *lower_bound(x);
    return l.a * x + l.b;
  }
};
```

3.2 Link Cut Tree [08546]

```
#define ll long long
// 1-based, needs splay
// vertex add paths sum link-cut
struct LCT{
  Splay sp;
  void access(int x){
    sp.splay(x);
    sp.ch[x][1] =  0;
    sp.pull(x);
    while(sp.fa[x]){
      int u = sp.fa[x];
      sp.splay(u);
      sp.push(u);
      sp.ch[u][1] = x;
      sp.pull(u);
      sp.splay(x);
    }
  }
  void makeroot(int x){
    access(x);sp.splay(x);
    sp.rev[x] ^= 1;
  }
  void link(int u,int v){
    makeroot(u);
    sp.splay(u);
    sp.fa[u] = v;
  }
  void cut(int u,int v){
    makeroot(u);
    access(v);
    sp.splay(v);
    int lc = sp.ch[v][0];
    sp.fa[lc] = 0;
    sp.ch[v][0] = 0;
    sp.pull(v);
  }
  ll path_sum(int u,int v){
    makeroot(u);
    access(v);
    sp.splay(v);
    return sp.sum[v];
  }
  void addval(int u,int val){
    sp.splay(u);
    sp.val[u] += val;
    sp.pull(u);
    return;
  }
  int find(int p){
    access(p);
    return sp.get_sz(p,1);
  }
};
```

3.3 Li Chao [7c3a7]

```
// range add line get min
// can even be used if modifies aren't range modify
#define ll long long
const ll SZ = 8e6+10;
const ll inf = 3e18;
vector<ll> all; // coordinates are stored here
struct Line{
  ll m,b;
  Line(ll mm = 0,ll bb = 0):m(mm),b(bb){}
  ll operator()(ll k){
    return m*k+b;
  }
};
struct LiChao{
#define ls now*2+1
#define rs now*2+2
#define mid ((l+r)>>1)
  Line seg[SZ];
  LiChao(){
    fill(seg,seg+SZ,Line(0,inf));
  }
  void modify(int now,int l,int r,int s,int e,Line v){
    if(l == r){
      if(seg[now](all[l])>v(all[l]))swap(seg[now],v);
      return;
    }
    if(l>=s&&e>=r){
      if(seg[now](all[mid])>v(all[mid]))swap(seg[now],v);
      if(seg[now].m<v.m)modify(ls,l,mid,s,e,v);
      else modify(rs,mid+1,r,s,e,v);
    }
    else{
      if(mid>=s)modify(ls,l,mid,s,e,v);
      if(mid<e)modify(rs,mid+1,r,s,e,v);
    }
    return;
  }
  ll getval(int now,int l,int r,int p){
    if(l == r)return seg[now](all[p]);
    if(mid>=p)return min(seg[now]
(all[p]),getval(ls,l,mid,p));
    else return min(seg[now](all[p]),getval(rs,mid+1,r,p));
  }
  void add_line(int s,int e,Line v){
    modify(0,0,all.size()-1,s,e,v);
    return;
  }
  ll getmin(int p){
    return getval(0,0,all.size()-1,p);
  }
#undef ls
#undef rs
#undef mid
};
#undef ll long long
```

3.4 Splay [214ae]

```
#define ll long long
const int SZ = 2e5+10;
//1-indexed,0 used for nullptr
//range reverse range sum
struct Splay{
#define ls ch[x][0]
#define rs ch[x][1]
#define p fa[x]
#define g fa[fa[x]]
  ll val[SZ];
  ll sum[SZ];
  int ch[SZ][2],fa[SZ],cnt,rev[SZ],sz[SZ];
  void pull(int x){
    if(!x)return;
    sum[x] = sum[ls]+sum[rs]+val[x];
    sz[x] = sz[ls]+sz[rs]+1;
    return;
  }
  void push(int x){
    if(!x)return;
    if(rev[x]){
      swap(ls,rs);
      rev[ls] ^= 1;
      rev[rs] ^= 1;
      rev[x] = 0;
    }
    pull(x);
    return;
  }
  Splay(){
    fill(sz+1,sz+SZ,1);
    return;
  }
  int newnode(){
    return ++cnt;
  }
  int dir(int x){//is ls or rs
    return ch[p][1] == x;
  }
  bool isroot(int x){//the || is for LCT
    return !p||ch[p][dir(x)] != x;
  }
  void rot(int x){ //g, p, x, here are _g, _p, _x
    int _p = p, _g = g, _x = x;
    push(_g); push(_p); push(_x);
    int d = dir(_x);
    if(!isroot(_p))ch[_g][dir(_p)] = _x;
    fa[ch[x][d^1]] = _p;
    ch[_p][d] = ch[_x][d^1];
    fa[_x] = _g;
    fa[_p] = _x;
    ch[_x][d^1] = _p;
    pull(_p);
    pull(_x);
    return;
  }
  void splay(int x){
    if(!x)return;
    while(!isroot(x)){
      push(g); push(p); push(x);
      if(!isroot(p)){
        rot(dir(p) == dir(x)? p: x);
      }
      rot(x);
    }
    push(x);
    return;
  }
  int get_sz(int x,int y){
    push(x);
    while(x&&sz[ls]+1 != y){
      if(sz[ls]>=y)x = ls;
      else{
        y -= sz[ls]+1;
        x = rs;
      }
      push(x);
    }
    return x;
  }
  void merge(int a,int b){
    if(!a||!b)return;
    splay(a);splay(b);
    a = get_sz(a,sz[a]);
    b = get_sz(b,1);
    splay(a);splay(b);
    ch[a][1] = b;
    fa[b] = a;
    pull(a);
    return;
  }
  pair<int,int> split(int a,int s){
    splay(a);
    if(!s)return make_pair(0,a);
    int b = get_sz(a,s);
    splay(b);
    pair<int,int> re;
    re.first = b;
    re.second = ch[b][1];
    fa[ch[b][1]] = 0;
    ch[b][1] = 0;
    pull(b);
    return re;
  }
#undef ls
#undef rs
#undef p
#undef g
};
```

3.5 Treap [ff400]

```
#define ll long long
// range reverse range add range sum
// need to push before using the info on node
struct node{
  int pri;
  int pl,pr;
  ll sum,tag,val;
  int sz;
  int rev;
  node(){
    pl = pr = sum = tag = 0;
    sz = 0;
    rev = 0;
    pri = rand();
  }
};

const int SZ = 2e5+10;
struct Treap{
  node nd[SZ];
  int cnt = 0;
  Treap(){
    cnt = 0;
  }
  int newnode(){
    cnt++;
    nd[cnt].sz = 1;
    return cnt;
  }
  void pull(int now){
    if(!now)return;
    nd[now].sz = nd[nd[now].pr].sz+nd[nd[now].pl].sz+1;
    ll ls =
nd[nd[now].pl].sum+nd[nd[now].pl].tag*nd[nd[now].pl].sz;
    ll rs =
nd[nd[now].pr].sum+nd[nd[now].pr].tag*nd[nd[now].pr].sz;
    nd[now].sum = nd[now].val+ls+rs;
    return;
  }
  void push(int now){
    if(!now)return;
    if(nd[now].rev){
      swap(nd[now].pl,nd[now].pr);
      if(nd[now].pl)nd[nd[now].pl].rev ^= 1;
      if(nd[now].pr)nd[nd[now].pr].rev ^= 1;
      nd[now].rev = 0;
    }
    int tl = nd[now].pl,tr = nd[now].pr;
    nd[now].val += nd[now].tag;
    if(tl)nd[tl].tag += nd[now].tag;
    if(tr)nd[tr].tag += nd[now].tag;
    nd[now].tag = 0;
    pull(now);
  }
  int merge(int a,int b){
    if(!a)return b;
    if(!b)return a;
    if(nd[a].pri>nd[b].pri){
      push(a);
      nd[a].pr = merge(nd[a].pr,b);
      pull(a);
      return a;
    }
    else{
      push(b);
      nd[b].pl = merge(a,nd[b].pl);
      pull(b);
      return b;
    }
  }
  void split(int now,int &a,int &b,int tar){
    if(!now){
      a = b = 0;
      return;
    }
    push(now);
    if(nd[nd[now].pl].sz+1<=tar){
      a = now;
      split(nd[now].pr,nd[a].pr,b,tar-
(nd[nd[now].pl].sz+1));
    }
    else{
      b = now;
      split(nd[now].pl,a,nd[b].pl,tar);
    }
    pull(a);
    pull(b);
    return;
  }
};
Treap T;
```

3.6 Quadrangle [1d61e]

```
struct QUADRANGLE {
  struct TUPLE {
    int l, r, id;
    TUPLE() {}
    TUPLE(int _l, int _r, int _id) : l(_l), r(_r), id(_id)
{}
  };
  int n, now;
  deque<TUPLE> dq;

  int calc_dp(int id, int i) {
    // ...
  }
  bool cmp(int cid, int pid, int i) {
    // ...
  }
  void init(int _n) {
    n = _n;
    now = 1;
    dq.clear();
  }
  void kill_head() {
    now++;
    if (dq.front().l == dq.front().r) dq.pop_front();
    else dq.front().l++;
  }
  void push(int id) {
    while (dq.size()) {
      TUPLE tl = dq.back();
      dq.pop_back();
      if (cmp(id, tl.id, tl.l)) {
        continue;
      }
      int l = tl.l, r = tl.r + 1;
      while (l + 1 < r) {
        int mid = (l + r) >> 1;
        (cmp(id, tl.id, mid) ? r : l) = mid;
      }
      dq.push_back(TUPLE(tl.l, l, tl.id));
      if (r <= n) dq.push_back(TUPLE(r, n, id));
      return;
    }
    dq.push_back(TUPLE(now, n, id));
  }
  int determine(int id) {
    return calc_dp(dq.front().id, id);
  }
};
```

3.7 SMAWK [f3776]

```
// For all 2x2 submatrix:
// If M[1][0] < M[1][1], M[0][0] < M[0][1]
// If M[1][0] == M[1][1], M[0][0] <= M[0][1]
// M[i][ans_i] is the best value in the i-th row
VI smawk(int N, int M, auto &&select) {
  auto dc = [&](auto self, const VI &r, const VI &c) {
    if (r.empty()) return VI{};
    const int n = (int)r.size(); VI ans(n), nr, nc;
    for (int i : c) {
      while (!nc.empty() &&
          select(r[nc.size() - 1], nc.back(), i))
        nc.pop_back();
      if (int(nc.size()) < n) nc.push_back(i);
    }
    for (int i = 1; i < n; i += 2) nr.push_back(r[i]);
    const auto na = self(self, nr, nc);
    for (int i = 1; i < n; i += 2) ans[i] = na[i >> 1];
    for (int i = 0, j = 0; i < n; i += 2) {
      ans[i] = nc[j];
      const int end = i + 1 == n ? nc.back() : ans[i + 1];
      while (nc[j] != end)
        if (select(r[i], ans[i], nc[++j])) ans[i] = nc[j];
    }
    return ans;
  };
  VI R(N), C(M); iota(all(R), 0), iota(all(C), 0);
  return dc(dc, R, C);
}
bool min_plus_conv_select(int r, int u, int v) {
  auto f = [](int i, int j) {
    if (0 <= i - j && i - j < n) return b[j] + a[i - j];
    return 2100000000 + (i - j);
  };
  return f(r, u) > f(r, v);
} // if f(r, v) is better than f(r, u), return true
```

4 Geometry

4.1 Point [eb195]

```
template<typename T = int>
struct Pt{
  T x,y;
  Pt (T xx = (T)(0),T yy = (T)(0)):x(xx),y(yy){}
  Pt operator+(Pt b)const{return Pt(x+b.x,y+b.y);}
  Pt operator-(Pt b)const{return Pt(x-b.x,y-b.y);}
  T operator*(Pt b)const{return x*b.x+y*b.y;}
  T operator^(Pt b)const{return x*b.y-y*b.x;}
  T operator/(Pt b)const{return x*b.y-y*b.x;}
  auto operator<=>(const Pt& b) const = default; // since C+
+20

  friend int dir(Pt a,Pt b){//returns sign(a ^ b)
    auto re = a ^ b;
    return re<0?-1:re>0?1:0;
  }
  friend bool onseg(Pt x,Pt s,Pt e){
    if(((e-x)^(s-x)) != 0)return false;
    else if((s-x)*(e-x)>0)return false;
    return true;
  }
  friend int intersect(Pt s1,Pt e1,Pt s2,Pt e2){//returns 0
if doesn't intersect,1 if intersect,2 if on line
    if(onseg(s1,s2,e2)||onseg(e1,s2,e2)||onseg(s2,s1,e1)||
onseg(e2,s1,e1))return 2;
    if(dir(s1-s2,e2-s2)*dir(e1-s2,e2-s2)<0&&dir(s2-s1,e1-
s1)*dir(e2-s1,e1-s1)<0)return 1;
    return 0;
  }

};
```

4.2 Convex Hull [17050]

```
// needs Point.cpp
template<typename T = int>
struct ConvexHull{
  //returns in clockwise direction
  // returns strictly on convex hull
  vector<Pt<T>> solve(vector<Pt<T>> v){
    sort(v.begin(),v.end());
    vector<Pt<T>> u,d;
    for(auto &i:v){
      if (!u.empty() && u.back() == i) continue;
      while(u.size()>1&&((i-u.end()[-1])^(u.end()[-2]-
u.end()[-1]))>=0)u.pop_back();
      while(d.size()>1&&((i-d.end()[-1])^(d.end()[-2]-
d.end()[-1]))<=0)d.pop_back();
      u.push_back(i);
      d.push_back(i);
    }
    for(int i = 1;i+1<d.size();i++)u.push_back(d.end()[-1-
i]);
    return u;
  }
};
```

4.3 Tangent of Convex Hull [e8a26]

```
int ori(Pt a, Pt b, Pt c) {
  ll tmp = (b-a) ^ (c-a);
  return tmp>0?1:tmp<0?-1:0;
}

int cyc_tsearch(int n, auto pred) {
  if (n == 1) return 0;
  int l = 0, r = n; bool rv = pred(1, 0);
  while(r-l > 1) {
    int m = (l+r) / 2;
    if (pred(0, m) ? rv: pred(m, (m+1) % n)) r = m;
    else l = m;
  }
  return pred(l, r % n) ? l : r % n;
}

pii get_tangent(const vector<Pt> &cvx, Pt p) {
  auto gao = [&](int s) {
    return cyc_tsearch(cvx.size(), [&](int x, int y)
     { return ori(p, cvx[x], cvx[y]) == s; });
  };
  return pii(gao(1), gao(-1));
}
```

4.4 Point In Convex [f8664]

```
bool PointInConvex(const vector<pll> &C, pll p, bool strict
= true) {
  int a = 1, b = SZ(C) - 1, r = !strict;
  if (SZ(C) == 0) return false;
  if (SZ(C) < 3) return r && btw(C[0], C.back(), p);
  if (ori(C[0], C[a], C[b]) > 0) swap(a, b);
  if (ori(C[0], C[a], p) >= r || ori(C[0], C[b], p) <= -r)
    return false;
  while (abs(a - b) > 1) {
    int c = (a + b) / 2;
    (ori(C[0], C[c], p) > 0 ? b : a) = c;
  }
  return ori(C[a], C[b], p) < r;
}
```

4.5 Minkowski Sum [9db95]

```
// needs Point template
template <typename T>
vector<Pt<T>> minkowski(vector<Pt<T>> va,vector<Pt<T>> vb){
  deque<Pt<T>> a,b;
  for(auto &i:va)a.push_back(i);
  for(auto &i:vb)b.push_back(i);
  Pt head = *min_element(a.begin(),a.end());
  while(a[0].x != head.x||a[0].y != head.y){
    a.push_back(a[0]);
    a.pop_front();
  }
  head = *min_element(b.begin(),b.end());
  while(b[0].x != head.x||b[0].y != head.y){
    b.push_back(b[0]);
    b.pop_front();
  }
  a.push_back(a[0]);
  b.push_back(b[0]);
  int p1 = 0,p2 = 0;
  vector<Pt<T>> re;
  while(p1 < a.size()&&p2 < b.size()){
    //cerr<<a.size()<<','<<b.size()<<":"<<p1<<' '<<p2<<endl;
    int dir = 0;
    re.push_back(a[p1]+b[p2]);
    if(p1+1 == a.size())dir = 1;
    else if(p2+1 == b.size())dir = 0;
    else if(((a[p1+1]-a[p1])^(b[p2+1]-b[p2]))>0)dir = 0;
    else dir = 1;
    if(dir == 0)p1++;
    else p2++;
  }
  return re;
}
```

4.6 Half Plane Intersection [e35a6]

```
// please don't use with other geometry templates
#define iter(v) v.begin(), v.end()
#define SZ(v) int(v.size())
#define pb emplace_back
#define ff first
#define ss second

using ll = long long;
using pii = pair<int, int>;
using pll = pair<ll, ll>;

template<class A, class B>
ostream &operator<<(ostream &o, pair<A, B> p) {
  return o << '(' << p.ff << ',' << p.ss << ')';
}

#define temp template<class T>
#define ptt pair<T, T>
#define X ff
#define Y ss
using ld = long double;
using pdd = pair<ld, ld>;

temp ptt operator+(ptt a, ptt b) {
  return {a.X + b.X, a.Y + b.Y};
}
temp ptt operator-(ptt a, ptt b) {
  return {a.X - b.X, a.Y - b.Y};
}
temp ptt operator*(ptt v, T i) {
  return {v.X * i, v.Y * i};
}
temp ptt operator*(T i, ptt v) {
  return {v.X * i, v.Y * i};
}
temp ptt operator/(ptt v, T i) {
  return {v.X / i, v.Y / i};
}
temp T dot(ptt a, ptt b) {
  return a.X * b.X + a.Y * b.Y;
}
temp T cross(ptt a, ptt b) {
  return a.X * b.Y - a.Y * b.X;
}
temp T abs2(ptt a) {
  return dot(a, a);
}
temp ld abs(ptt a) {
  return sqrt(abs2(a));
}
temp int sgn(T v) {
  return v > 0 ? 1 : (v < 0 ? -1 : 0);
}
temp int ori(ptt a, ptt b, ptt c) {
  return sgn(cross(b - a, c - a));
}
// intersects Line(p1, p2), Line(p3, p4)
pdd intersect(pdd p1, pdd p2, pdd p3, pdd p4) {
  ld a123 = cross(p2 - p1, p3 - p1);
  ld a124 = cross(p2 - p1, p4 - p1);
  return (p4 * a123 - p3 * a124) / (a123 - a124);
}

int cmp(pll a, pll b, bool same = true) {
#define is_neg(k) (sgn(k.Y) < 0 || (sgn(k.Y) == 0 &&
sgn(k.X) < 0))
  int A = is_neg(a), B = is_neg(b);
  if (A != B) return A < B;
  if (sgn(cross(a, b)) == 0) return same ? abs2(a) <
abs2(b) : -1;
  return sgn(cross(a, b)) > 0;
}

using Line = pair<pll, pll>;
// cross(p - line.X, line.Y-line.X) <= 0 <-> p in half plane
// LHS when going from line.X to line.Y

pll area_pair(Line a, Line b) {
  return pll(cross(a.Y - a.X, b.X - a.X), cross(a.Y - a.X,
b.Y - a.X));
}
bool isin(Line l0, Line l1, Line l2) {
  auto [a02X, a02Y] = area_pair(l0, l2);
  auto [a12X, a12Y] = area_pair(l1, l2);
  if (a12X - a12Y < 0) a12X *= -1, a12Y *= -1;
  return (__int128) a02Y * a12X - (__int128) a02X * a12Y >
0;
}
vector<Line> HalfPlaneInter(vector<Line> arr) {
  sort(iter(arr), [&](Line a, Line b) -> int {
    if (cmp(a.Y - a.X, b.Y - b.X, 0) != -1)
    return cmp(a.Y - a.X, b.Y - b.X, 0);
    return ori(a.X, a.Y, b.Y) < 0;
  });
  deque<Line> dq(1, arr[0]);
  for (auto p : arr) {
    if (cmp(dq.back().Y - dq.back().X, p.Y - p.X, 0) == -1)
      continue;
    while (SZ(dq) >= 2 && !isin(p, dq[SZ(dq) - 2],
dq.back()))
      dq.pop_back();
    while (SZ(dq) >= 2 && !isin(p, dq[0], dq[1]))
      dq.pop_front();
    dq.pb(p);
  }
  while (SZ(dq) >= 3 && !isin(dq[0], dq[SZ(dq) - 2],
dq.back()))
    dq.pop_back();
  while (SZ(dq) >= 3 && !isin(dq.back(), dq[0], dq[1]))
    dq.pop_front();
  return vector<Line>(iter(dq));
}
```

4.7 Min Enclosing Circle [f416d]

```
typedef pair<int, int> pii;
#define ld double
#define pdd Pt<ld>

ld len(pdd k) {
  return sqrt(k*k);
}
pdd excenter(pdd p0, pdd p1, pdd p2) {
  p1 = p1-p0;
  p2 = p2-p0;
  ld x1 = p1.x, y1 = p1.y, x2 = p2.x, y2 = p2.y;
  ld m = 2.0 * (x1*y2-y1*x2);
  pdd center;
  center.x = (x1*x1*y2 - x2*x2*y1 + y1*y2*(y1-y2)) / m;
  center.y = (x1*x2*(x2-x1) - y1*y1*x2 + x1*y2*y2) / m;
  return center + p0;
}

pdd Minimum_Enclosing_Circle(vector<pdd> dots, ld &r) {
  mt19937 seed(time(0));
  shuffle(dots.begin(), dots.end(), seed);
  pdd cent;
  cent = dots[0], r = 0;
  for(int i = 1;i<dots.size();i++) {
    if (len(dots[i]-cent) > r) {
      cent = dots[i], r = 0;
      for(int j = 0;j<i;j++) {
        if (len(dots[j]-cent) > r) {
          cent = (dots[i]+dots[j]);
          cent.x /= 2, cent.y /= 2;
          r = len(dots[i]-cent);
          for(int k = 0;k<j;k++) {
            if(len(dots[k]-cent) > r) {
              cent = excenter(dots[i], dots[j], dots[k]);
              r = len(dots[k]-cent);
            }
          }
        }
      }
    }
  }
  return cent;
}
```

4.8 Tangent Of Two Circles [48eb6]

```
struct Cir{ pdd O; double R; };
vector<Line> go( const Cir& c1 , const Cir& c2 , int sign1 )
{
  // sign1 = 1 for outer tang, -1 for inter tang
  vector<Line> ret;
  double d_sq = abs2(c1.O - c2.O);
  if (sign(d_sq) == 0) return ret;
  double d = sqrt(d_sq);
  pdd v = (c2.O - c1.O) / d;
  double c = (c1.R - sign1 * c2.R) / d;
  if (c * c > 1) return ret;
  double h = sqrt(max(0.0, 1.0 - c * c));
  for (int sign2 = 1; sign2 >= -1; sign2 -= 2) {
    pdd n = pdd(v.X * c - sign2 * h * v.Y,
      v.Y * c + sign2 * h * v.X);
    pdd p1 = c1.O + n * c1.R;
    pdd p2 = c2.O + n * (c2.R * sign1);
    if (sign(p1.X - p2.X) == 0 and
        sign(p1.Y - p2.Y) == 0)
      p2 = p1 + perp(c2.O - c1.O);
    ret.pb(Line(p1, p2));
  }
  return ret;
}
```

4.9 Circle Cover [861b1]

```
const int N = 1021;
struct CircleCover {
  int C;
  Cir c[N];
  bool g[N][N], overlap[N][N];
  // Area[i] : area covered by at least i circles
  double Area[ N ];
  void init(int _C){ C = _C;}
  struct Teve {
    pdd p; double ang; int add;
    Teve() {}
    Teve(pdd _a, double _b, int _c):p(_a), ang(_b), add(_c)
{}
    bool operator<(const Teve &a)const
    {return ang < a.ang;}
  }eve[N * 2];
  // strict: x = 0, otherwise x = -1
  bool disjuct(Cir &a, Cir &b, int x)
```

```
 {return sign(abs(a.O - b.O) - a.R - b.R) > x;}
 bool contain(Cir &a, Cir &b, int x)
 {return sign(a.R - b.R - abs(a.O - b.O)) > x;}
 bool contain(int i, int j) {
  /* c[j] is non-strictly in c[i]. */
  return (sign(c[i].R - c[j].R) > 0 || (sign(c[i].R -
c[j].R) == 0 && i < j)) && contain(c[i], c[j], -1);
 }
 void solve(){
  fill_n(Area, C + 2, 0);
  for(int i = 0; i < C; ++i)
   for(int j = 0; j < C; ++j)
    overlap[i][j] = contain(i, j);
  for(int i = 0; i < C; ++i)
   for(int j = 0; j < C; ++j)
    g[i][j] = !(overlap[i][j] || overlap[j][i] ||
      disjuct(c[i], c[j], -1));
  for(int i = 0; i < C; ++i){
   int E = 0, cnt = 1;
   for(int j = 0; j < C; ++j)
    if(j != i && overlap[j][i])
     ++cnt;
   for(int j = 0; j < C; ++j)
    if(i != j && g[i][j]) {
     pdd aa, bb;
     CCinter(c[i], c[j], aa, bb);
     double A = atan2(aa.Y - c[i].O.Y, aa.X -
c[i].O.X);
     double B = atan2(bb.Y - c[i].O.Y, bb.X -
c[i].O.X);
     eve[E++] = Teve(bb, B, 1), eve[E++] = Teve(aa, A,
-1);
     if(B > A) ++cnt;
    }
   if(E == 0) Area[cnt] += pi * c[i].R * c[i].R;
   else{
    sort(eve, eve + E);
    eve[E] = eve[0];
    for(int j = 0; j < E; ++j){
     cnt += eve[j].add;
     Area[cnt] += cross(eve[j].p, eve[j + 1].p) * .5;
     double theta = eve[j + 1].ang - eve[j].ang;
     if (theta < 0) theta += 2. * pi;
     Area[cnt] += (theta - sin(theta)) * c[i].R *
c[i].R * .5;
    }
   }
  }
 }
};
```

5 String

5.1 Z Algorithm [7a5e2]

```
template <typename T>
struct Z_alg {
 void operator()(T a, int n, int *z) {
  z[0] = 0;
  int l = 0;
  for (int i = 1; i <= n; i++) {
   for (z[i] = max(0, min(z[i - l], l + z[l] - i)); i +
z[i] < n && a[i + z[i]] == a[z[i]]; z[i]++);
   if (i + z[i] > l + z[l]) l = i;
  }
 }
};
```

5.2 KMP [bb2a1]

```
template <typename T>
struct KMP {
 void operator()(T a, int n, int *pi) {
  pi[0] = -1, pi[1] = 0;
  for (int i = 1; i < n; i++) {
   int j = pi[i];
   while (j >= 0 && a[i] != a[j]) j = pi[j];
   pi[i + 1] = j + 1;
  }
 }
};
```

5.3 Aho Corasick [fa0e4]

```
// only construct the automaton
struct AC {
 const static int c0 = 'a';
 int nc, c[MXN], pi[MXN], p[MXN], ch[MXN][MXC];
 void init() {
  nc = 2;
  fill(ch[0], ch[0] + MXC, 1);
  fill(ch[1], ch[1] + MXC, -1);
 }
 int nn(int pp, char cc) {
  c[nc] = cc;
  p[nc] = pp;
  fill(ch[nc], ch[nc] + MXC, -1);
  return nc++;
 }
 int push(const string &s) {
  int u = 1;
  for (auto &i : s) {
   int e = i - c0;
   if (!~ch[u][e]) ch[u][e] = nn(u, i);
   u = ch[u][e];
  }
  return u;
 }
 void build() {
  queue<int> q;
  q.push(1);
  while (q.size()) {
   int u = q.front();
   q.pop();
   pi[u] = (u == 1 ? 0 : ch[pi[p[u]]][c[u] - c0]);
   FOR(e, 0, MXC) {
    if (!~ch[u][e]) ch[u][e] = ch[pi[u]][e];
    else q.push(ch[u][e]);
   }
  }
 }
};
```

5.4 Manacher [b6ad8]

```
template <typename T>
struct MANACHER {
 void operator()(T a, int n, int *mn) {
  int l = 0;
  mn[0] = 0;
  for (int i = 1; i < n; i++) {
   mn[i] = (l + mn[l] >= i ? min(mn[2 * l - i], l + mn[l]
- i) : 0);
   while (i - mn[i] - 1 >= 0 && i + mn[i] + 1 < n && a[i
- mn[i] - 1] == a[i + mn[i] + 1]) mn[i]++;
   if (i + mn[i] > l + mn[l]) l = i;
  }
 }
};
```

5.5 Suffix Array [a683f]

```
int SA[MXN * 2], H[MXN], RA[MXN];
namespace SAIS {
  bool _t[MXN * 2];
  int _s[MXN * 2], _c[MXN * 2], x[MXN], _p[MXN], _q[MXN * 2];
  void pre(int *sa, int *c, int n, int z) {
    fill_n(sa, n, 0);
    copy_n(c, z, x);
  }
  void induce(int *sa, int *c, int *s, bool *t, int n, int z) {
    copy_n(c, z - 1, x + 1);
    FOR(i, 0, n) {
      if (sa[i] && !t[sa[i] - 1]) {
        sa[x[s[sa[i] - 1]]++] = sa[i] - 1;
      }
    }
    copy_n(c, z, x);
    for (int i = n - 1; i >= 0; i--) {
      if (sa[i] && t[sa[i] - 1]) {
        sa[--x[s[sa[i] - 1]]] = sa[i] - 1;
      }
    }
  }
  void sais(int *s, int *sa, int *p, int *q, bool *t, int *c, int n, int z) {
    bool uniq = t[n - 1] = true;
    int nn = 0, nmxz = -1, *nsa = sa + n, *ns = s + n, last = -1;
    fill_n(c, z, 0);
    FOR(i, 0, n) uniq &= ++c[s[i]] < 2;
    partial_sum(c, c + z, c);
    if (uniq) {
      FOR(i, 0, n) sa[--c[s[i]]] = i;
      return;
    }
    for (int i = n - 2; i >= 0; i--) {
      t[i] = (s[i] == s[i + 1] ? t[i + 1] : s[i] < s[i + 1]);
    }
    pre(sa, c, n, z);
    FOR(i, 1, n) {
      if (t[i] && !t[i - 1]) {
        sa[--x[s[i]]] = p[q[i] = nn++] = i;
      }
    }
    induce(sa, c, s, t, n, z);
    FOR(i, 0, n) {
      if (sa[i] && t[sa[i]] && !t[sa[i] - 1]) {
        bool neq = last < 0 || !equal(s + sa[i], s + p[q[sa[i]] + 1], s + last);
        ns[q[last = sa[i]]] = nmxz += neq;
      }
    }
    sais(ns, nsa, p + nn, q + n, t + n, c + z, nn, nmxz + 1);
    pre(sa, c, n, z);
    for (int i = nn - 1; i >= 0; i--) {
      sa[--x[s[p[nsa[i]]]]] = p[nsa[i]];
    }
    induce(sa, c, s, t, n, z);
  }
  void mkhei(int n) {
    for (int i = 0, j = 0; i < n; i++) {
      if (RA[i]) {
        for (; i + j < n && SA[RA[i] - 1] + j < n && _s[i + j] == _s[SA[RA[i] - 1] + j]; ++j);
        H[RA[i]] = j, j = max(0, j - 1);
      }
    }
  }
  void build(int *s, int n, int mxc) {
    copy_n(s, n, _s), _s[n] = 0;
    sais(_s, SA, _p, _q, _t, _c, n + 1, mxc);
    copy_n(SA + 1, n, SA);
    FOR(i, 0, n) RA[SA[i]] = i;
    mkhei(n);
    copy(H + 1, H + n, H);
  }
}
```

5.6 SAM [47371]

```
struct SAM {
  static const int MXND = 1000005, MXC = 33, C0 = 'a';
  int tot, rt, lst, pi[MXND], mx[MXND];
  int nxt[MXND][MXC], cnt[MXND], in[MXND];
  int newNode() {
    int res = ++tot;
    fill(nxt[res], nxt[res] + MXC, 0);
    pi[res] = mx[res] = cnt[res] = in[res] = 0;
    return res;
  }
  void init() {
    tot = 0;
    rt = newNode();
    pi[rt] = 0, mx[rt] = 0;
    lst = rt;
  }
  void push(int c) {
    int p = lst;
    int np = newNode();
    mx[np] = mx[p] + 1;
    for (; p && nxt[p][c] == 0; p = pi[p])
      nxt[p][c] = np;
    if (p == 0) pi[np] = rt;
    else {
      int q = nxt[p][c];
      if (mx[p] + 1 == mx[q]) pi[np] = q;
      else {
        int nq = newNode();
        mx[nq] = mx[p] + 1;
        for (int i = 0; i < MXC; i++)
          nxt[nq][i] = nxt[q][i];
        pi[nq] = pi[q];
        pi[q] = nq;
        pi[np] = nq;
        for (; p && nxt[p][c] == q; p = pi[p])
          nxt[p][c] = nq;
      }
    }
    lst = np, cnt[np] = 1;
  }
  void push(char *str) {
    for (int i = 0; str[i]; i++)
      push(str[i] - C0 + 1);
  }
  void count() {
    for (int i = 1; i <= tot; ++i)
      ++in[pi[i]];
    queue<int> q;
    for (int i = 1; i <= tot; ++i)
      if (!in[i]) q.push(i);
    while (!q.empty()) {
      int u = q.front();
      q.pop();
      cnt[pi[u]] += cnt[u];
      if (!--in[pi[u]])
      q.push(pi[u]);
    }
  }
} sam;
```

5.7 eertree [ca7d7]

```
#define pb emplace_back
struct eertree {
  const static int MXC = 26, C0 = 'a';
  struct nd {
    int nxt[MXC], pi, len;
    int cnt, num; // optional
    nd(int l = 0) : pi(0), len(l), cnt(0), num(0) {
      fill(nxt, nxt + MXC, 0);
    }
  };
  vector<nd> v;
  vector<char> s;
  int lst, n;
  eertree() : v(2), lst(1), n(0) {
    v[0].pi = 1, v[1].len = -1, s.pb(-1);
  }
  void clear() {
    v.clear(), s.clear(), lst = 1, n = 0;
    v.pb(0), v.pb(-1);
    v[0].pi = 1, s.pb(-1);
  }
  int get_fail(int x) {
    while (s[n - v[x].len - 1] != s[n])
      x = v[x].pi;
    return x;
  }
  void add(int c) {
    s.push_back(c -= 'a'), ++n;
    int cur = get_fail(lst);
    if (!v[cur].nxt[c]) {
      int now = v.size();
      v.pb(v[cur].len + 2);
      v[now].pi =
        v[get_fail(v[cur].pi)].nxt[c];
      v[cur].nxt[c] = now;
      v[now].num = v[v[now].pi].num + 1;
    }
    lst = v[cur].nxt[c], ++v[lst].cnt;
  }
  void count() {
    for (auto i = v.rbegin(); i != v.rend(); i++) {
      v[i -> pi].cnt += i -> cnt;
    }
  }
  inline int size() {
    return v.size() - 2;
  }
};
```

5.8 minimal rotation [7b1de]

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

6 Math

6.1 Chinese Remainder Theorem [6fdd6]

```
using lll = __int128_t;

struct ICRT {
  lll p1, p2, p3;
  lll c1, c2, c3;
  ICRT() {}
  ICRT(lll _p1, lll _p2, lll _p3) : p1(_p1), p2(_p2),
p3(_p3) {
    auto POW = [&](lll a, lll b, lll mod) -> lll {
      lll ans = 1;
      while (b) {
        if (b & 1) ans = ans * a % mod;
        b >>= 1;
        a = a * a % mod;
      }
      return ans;
    };
    c1 = POW(p2 * p3 % p1, p1 - 2, p1) * p2 * p3;
    c2 = POW(p3 * p1 % p2, p2 - 2, p2) * p3 * p1;
    c3 = POW(p1 * p2 % p3, p3 - 2, p3) * p1 * p2;
  }
  lll operator()(int r1, int r2, int r3) {
    return (c1 * r1 + c2 * r2 + c3 * r3) % (p1 * p2 * p3);
  }
};

ICRT icrt(998244353, 104857601, 167772161);
```

6.2 Euclid [ffed2]

```
struct euclid{
  ll x, y, g;
  void ec(ll a, ll b){
    // minimum integer solution of "ax+by=g, x>0"
    if(!b) return void((x=1, y=0, g=a));
    ec(b, a%b);
    swap(x, y);
    y -= a/b*x+a/g;
    x += b/g;
  }
  inline euclid(ll a, ll b){
    ec(abs(a), abs(b));
    if(b<0) y = -y;
    if(a<0) x = -x;
  }
};
```

6.3 FFT [70ff0]

```
using cd = complex<double>;
struct PolyF : public vector<cd> {
  static constexpr double PI = 3.14159265358979323;
  PolyF() : vector<cd>() {}
  PolyF(size_t sz) : vector<cd>(sz) {}
  void conv(size_t N, bool inv = 0) {
    assert(size() && N >= size());
    int LG = __lg(N);
    assert(N == (1 << LG));
    resize(N);
    vector<int> r(N);
    FOR(i, 1, N) {
      int i_ = i ^ (1 << __lg(i));
      r[i] = r[i_] << (__lg(i) - __lg(i_)) | 1;
      int j = r[i] << (LG - 1 - __lg(i));
      if (i < j) {
        std::swap(at(i), at(j));
      }
    }
    for (int w = 1; w < N; w <<= 1) {
      FOR(ok, 0, w) {
        double th = PI * ok / w * (inv ? -1 : 1);
        cd o(cos(th), sin(th));
        for (int s = 0; s < N; s += (w << 1)) {
          cd &L = at(s + ok), &R = at(s + ok + w);
          cd l = L, r = o * R;
          L = l + r;
          R = l - r;
        }
      }
    }
    if (inv) {
      FOR(i, 0, N) {
        at(i) /= N;
      }
    }
  }
};
```

6.4 FWT [b1077]

```
//     AND       OR        XOR
//  | 1  1 |  | 1  0 |  | 1  1 |
//  | 0  1 |  | 1  1 |  | 1 -1 |

struct FWT {
  // mod operations ADD, SUB, MUL, POW (if needed)
  void btf(int &L, int &R, bool inv) { // sample: XOR
    int l = L, r = R;
    L = ADD(l, r);
    R = SUB(l, r);
  }
  void operator()(int *a, int n, bool inv) {
    // sample: XOR
    for (int w = 1; w < n; w <<= 1) {
      FOR(i, 0, n) if (i & w) {
        btf(a[i - w], a[i], inv);
      }
    }
    if (inv) {
      int x = POW(n, mod - 2);
      FOR(i, 0, n) a[i] = MUL(a[i], x);
    }
  }
};
```

6.5 NTT [592be]

```
#define FOR(i, j, k) for (int i = j, Z = k; i < Z; i++)

struct NTT {
  const static int LG = 20;
  int mod;
  int o[(1 << LG) + 1];
  int ADD(int a, int b) {
    // help yourself
  }
  int SUB(int a, int b) {
    // help yourself
  }
  int MUL(int a, int b) {
    // help yourself
  }
  int POW(int a, int b) {
    // help yourself
  }
  NTT(int g, int gap, int _mod) {
    mod = _mod;
    o[0] = 1;
    int pp = POW(g, gap);
    FOR(i, 1, (1 << LG) + 1) o[i] = MUL(o[i - 1], pp);
  }
  void operator()(int *a, int n, bool inv) {
    auto REV = [&](int x) -> int {
      int ans = 0;
      for (int w = 1; w < n; w <<= 1) {
        ans = (ans << 1) | (x & 1);
        x >>= 1;
      }
      return ans;
    };
    FOR(i, 0, n) {
      int j = REV(i);
      if (i < j) swap(a[i], a[j]);
    }
    for (int w = 1; w < n; w <<= 1) {
      int owo = 1 << (LG - __lg(w) - 1), oid = 0;
      FOR(i, 0, w) {
        int omega = o[inv ? (1 << LG) - oid : oid];
        for (int s = 0; s < n; s += (w << 1)) {
          int &L = a[s + i], &R = a[s + w + i];
          int l = L, r = MUL(omega, R);
          L = ADD(l, r);
          R = SUB(l, r);
        }
        oid += owo;
      }
    }
    if (inv) {
      int x = POW(n, mod - 2);
      FOR(i, 0, n) a[i] = MUL(a[i], x);
    }
  }
};

NTT ntt1(3, 952, 998244353);
NTT ntt2(3, 100, 104857601);
NTT ntt3(3, 160, 167772161);

namespace POLY {
  const int MXM = 4 * MXN;
  int a[MXM], b[MXM];
  vector<int> VMUL(vector<int> v, vector<int> w, int m) {
    int N = 4 << __lg(m);
    fill(a, a + N, 0);
    fill(b, b + N, 0);
    int na = min((int) v.size(), m), nb = min((int)
w.size(), m);
    FOR(i, 0, na) a[i] = v[i];
    FOR(i, 0, nb) b[i] = w[i];
    ntt(a, N, false);
    ntt(b, N, false);
    FOR(i, 0, N) a[i] = MUL(a[i], b[i]);
    ntt(a, N, true);
    vector<int> ans;
    FOR(i, 0, m) ans.push_back(a[i]);
    return ans;
  }
}
```

6.6 Pollard Rho [b24d9]

```
// needs mad,mub,mul,pw with changable mod
//!!! use int128 for pw and mul

bool isprime(ll x) {
  if (x <= 2 || ~x & 1) return x == 2;
  auto witn = [&](ll a, int t) {
    for (ll a2; t-- && (a2 = mul(a, a, x)); a = a2)
      if (a2 == 1 && a != 1 && a != x - 1) return true;
    return a > 1;
  };
  int t = __builtin_ctzll(x-1); ll odd = (x-1) >> t;
  for (ll m:
      {2, 325, 9375, 28178, 450775, 9780504, 1795265022})
  if (witn(pw(m % x, odd, x), t)) return false;
  return true;
}

ll pollard_rho(ll n) {
  static mt19937_64 rnd(120821011);
  if (!(n & 1)) return 2;
  ll y = 2, z = y, c = rnd() % n, p = 1, i = 0, t;
  auto f = [&](ll x) {
    return mad(mul(x, x, n), c, n); };
  do {
    p = mul(mub(z = f(f(z)), y = f(y), n), p, n);
    if (++i &= 63) if (i == (i & -i)) t = gcd(p, n);
  } while (t == 1);
  return t == n ? pollard_rho(n) : t;
}

vector<ll> factorize(ll k){
  if(k == 1)return {};
  else if(isprime(k))return {k};
  else{
    vector<ll> re;
    function<void(ll)> dc = [&](ll k){
      if(isprime(k)){
        re.push_back(k);
        return;
      }
      ll x = pollard_rho(k);
      dc(x);dc(k/x);
    };
    dc(k);
    sort(re.begin(),re.end());
    return re;
  }
}
```

6.7 Floor Sum [713b0]

```
ll floor_sum(ll a, ll b, ll c, ll n) {
  // floor((a * x + b) / c) for x in [0, n]
  if (n < 0) return 0;
  if (a == 0) return b / c * (n + 1);
  if (a >= c || b >= c) return (n * (n + 1) / 2 * (a / c) + (b / c) * (n + 1)) + floor_sum(a % c, b % c, c, n);
  int m = (a * n + b) / c;
  return m * n - floor_sum(c, c - b - 1, a, m - 1);
}
```

7 Ideograph Advantage

7.1 3d Donut [1ff2e]

```
#include <stdio.h>
#include <math.h>
#include <stdint.h>
#include <string.h>
#include <unistd.h>

int main() {
  float A = 0, B = 0;
  float i, j;
  int k;
  float z[1760];
  char b[1760];
  printf("\x1b[2J");
  for(;;) {
    memset(b,32,1760);
    memset(z,0,7040);
    for(j=0; j < 6.28; j += 0.07) {
      for(i=0; i < 6.28; i += 0.02) {
        float c = sin(i);
        float d = cos(j);
        float e = sin(A);
        float f = sin(j);
        float g = cos(A);
        float h = d + 2;
        float D = 1 / (c * h * e + f * g + 5);
        float l = cos(i);
        float m = cos(B);
        float n = sin(B);
        float t = c * h * g - f * e;
        int x = 40 + 30 * D * (l * h * m - t * n);
        int y= 12 + 15 * D * (l * h * n + t * m);
        int o = x + 80 * y;
        int N = 8 * ((f * e - c * d * g) * m - c * d * e - f * g - l * d * n);
        if(22 > y && y > 0 && x > 0 && 80 > x && D > z[o]) {
          z[o] = D;
          b[o] = ".,-~:;=!*#$@"[N > 0 ? N : 0];
        }
      }
    }
    printf("\x1b[H");
    for(k = 0; k < 1761; k++) {
      putchar(k % 80 ? b[k] : 10);
      A += 0.00004;
      B += 0.00002;
    }
    usleep(30000);
  }
  return 0;
}
```

8 Notes

- NO PATH COMPRESSION on rollback dsu please