Team Reference

Pollard

```cpp
const int max_step = 4e5;

unsigned long long gcd(unsigned long long a, unsigned long long b){
    if (!a) return 1;
    while (a) swap(a, b%=a);
    return b;
}

unsigned long long get(unsigned long long a, unsigned long long b){
    if (a > b)
        return a-b;
    else
        return b-a;
}

unsigned long long pollard(unsigned long long n){
    unsigned long long x = (rand() + 1) % n, y = 1, g;
    int stage = 2, i = 0;
    g = gcd(get(x, y), n);
    while (g == 1){
        if (i == max_step)
            break;
        if (i == stage){
            y = x;
            stage <<= 1;
        }
        x = (x * (__int128)x + 1) % n;
        i++;
        g = gcd(get(x, y), n);
    }
    return g;
}
```

pragma

```
#pragma GCC optimize(''O3,no-stack-protector'')
#pragma GCC target(sse,sse2,sse4,ssse3,popcnt,abm,mmx,avx,tune=native)
```

Алгебра Pick

В + Г / 2 − 1 = AREA,
где В — количество целочисленных точек внутри многоугольника, а Г — количество целочисленных точек на границе многоугольника.

Newton

$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$

Catalan

$C_n = \sum_{k=0}^{n-1} C_k C_{n-1-k}$
$C_i = \frac{1}{n+1}\binom{2n}{n}$

Кол-во графов

$G_N := 2^{n(n-1)/2}$
Количество связных помеченных графов
$Conn_N = G_N - \frac{1}{N}\sum_{K=1}^{N-1} K\binom{N}{K} Conn_K G_{N-K}$

Количество помеченных графов с K компонентами связности
$D[N][K] = \sum_{S=1}^{N} \binom{N-1}{S-1} Conn_S D[N-S][K-1]$

Miller-Rabbin

```
a=a^t
FOR i = 1...s
    if a^2=1 && |a|!=1
        NOT PRIME
    a=a^2
return a==1 ? PRIME : NOT PRIME
```

Интегрирование по формуле Симпсона

$\int_a^b f(x)dx?$
$x_i := a + ih, i = 0 \dots 2n$
$h = \frac{b-a}{2n}$

$\int = (f(x_0) + 4f(x_1) + 2f(x_2) + 4f(x_3) + 2f(x_4) + \ldots + 4f(x_{2n-1}) + f(x_{2n}))\frac{h}{3}$
$O(n^4)$.

Простые числа

1009,1013;10007,10009;100003,100019
1000003,1000033;10000019,10000079
100000007,100000037
10000000019,10000000033
1000000000039,1000000000061

100000000000031,100000000000067
10000000000000061,10000000000000069
1000000000000000003,1000000000000000009

Числа для Фурье

- prime: $7340033 = 7{\cdot}2^{20} + 1; w : 5(w^{2^{20}} = 1)$
- prime: $13631489 = 13{\cdot}2^{20} + 1; w : 3(w^{2^{20}} = 1)$
- prime: $23068673 = 11{\cdot}2^{21} + 1; w : 38(w^{2^{21}} = 1)$
- prime: $69206017 = 33{\cdot}2^{21} + 1; w : 45(w^{2^{21}} = 1)$
- prime: $81788929 = 39{\cdot}2^{21} + 1; w : 94(w^{2^{21}} = 1)$
- prime: $104857601 = 25{\cdot}2^{22} + 1; w : 21(w^{2^{22}} = 1)$
- prime: $113246209 = 27{\cdot}2^{22} + 1; w : 66(w^{2^{22}} = 1)$
- prime: $138412033 = 33{\cdot}2^{22} + 1; w : 30(w^{2^{22}} = 1)$
- prime: $167772161 = 5{\cdot}2^{25} + 1; w : 17(w^{2^{25}} = 1)$
- prime: $469762049 = 7{\cdot}2^{26} + 1; w : 30(w^{2^{26}} = 1)$
- prime: $998244353 = 7{\cdot}17{\cdot}2^{23} + 1; w : 3^{7*17}$.

```
template <typename Info>
class DSU {
  public:
    DSU ( int n ) : jump (new int[n]), rank (new int [n]), info (new Info [n]) {
      for (int i = 0; i < n; i++) {
        jump[i] = i;
        rank[i] = 0;
      }
    }
    Info& operator [] ( int x ) {
      return info[get (x)];
    }
    void merge ( int a, int b, const Info &comment ) {
      a = get (a);
      b = get (b);
      if (rank[a] <= rank[b]) {
        jump[a] = b;
        rank[b] += rank[a] == rank[b];
        info[b] = comment;
      } else {
        jump[b] = a;
        info[a] = comment;
      }
    }
  private:
    int *jump, *rank;
    Info *info;

    int get ( int x ) {
      return jump[x] == x ? x : (jump[x] = get (jump[x]));
    }
};


struct Treap {
  int value, add;
  int source, target, height;
  int min_value, min_path;

  Treap *left, *right;

  Treap ( int _source, int _target, int _value ) : value (_value), add (0), source
  ↪ (_source), target (_target) {
    height = rand ();
    min_value = value, min_path = 0;
    left = right = 0;
  }

  Treap& operator += ( int sub ) {
    add += sub;
    return *this;
```

```
51    }
52
53    void push () {
54      if (!add)
55        return;
56      if (left) {
57        left->add += add;
58      }
59      if (right) {
60        right->add += add;
61      }
62      value += add;
63      min_value += add;
64      add = 0;
65    }
66
67    void recalc () {
68      min_value = value;
69      min_path = 0;
70      if (left && left->min_value + left->add < min_value) {
71        min_value = left->min_value + left->add;
72        min_path = -1;
73      }
74      if (right && right->min_value + right->add < min_value) {
75        min_value = right->min_value + right->add;
76        min_path = +1;
77      }
78    }
79  };
80
81  Treap* treap_merge ( Treap *x, Treap *y ) {
82    if (!x)
83      return y;
84    if (!y)
85      return x;
86    if (x->height < y->height) {
87      x->push ();
88      x->right = treap_merge (x->right, y);
89      x->recalc ();
90      return x;
91    } else {
92      y->push ();
93      y->left = treap_merge (x, y->left);
94      y->recalc ();
95      return y;
96    }
97  }
98
99  Treap* treap_getmin ( Treap *x, int &source, int &target, int &value ) {
100   assert (x);
101   x->push ();
```

```
  if (x->min_path == 0) {
    // memory leak, sorry
    source = x->source;
    target = x->target;
    value = x->value + x->add;
    return treap_merge (x->left, x->right);
  } else if (x->min_path == -1) {
    x->left = treap_getmin (x->left, source, target, value);
    value += x->add;
    x->recalc ();
    return x;
  } else if (x->min_path == +1) {
    x->right = treap_getmin (x->right, source, target, value);
    value += x->add;
    x->recalc ();
    return x;
  } else
    assert (0);
}

Treap* treap_add ( Treap *x, int add ) {
  if (!x)
    return 0;
  return &((*x) += add);
}


int main () {
  int n, m;
  while (scanf ("%d%d", &n, &m) == 2) {
    Treap * g[n + 1];
    for (int i = 0; i <= n; i++)
      g[i] = 0;
    for (int i = 1; i <= n; i++) {
      int a;
      assert (scanf ("%d", &a) == 1);
      g[i] = treap_merge (g[i], new Treap (i, 0, a));
    }
    n++;
    for (int i = 0; i < m; i++) {
      int a, b, c;
      assert (scanf ("%d%d%d", &a, &b, &c) == 3);
      g[b] = treap_merge (g[b], new Treap (b, a, c));
    }
    DSU <pair <int, Treap*> > dsu (n + 1);
    for (int i = 0; i < n; i++) {
      dsu[i] = make_pair (i, g[i]);
    }

    int ans = 0, k = n;
    int jump[2 * n], jump_from[2 * n], parent[2 * n], c[n];
```

```
153     vector <int> children[2 * n];
154     memset (c, 0, sizeof (c[0]) * n);
155     memset (parent, -1, sizeof (parent[0]) * 2 * n);
156     vector <int> finish;
157     for (int i = 0; i < n; i++) {
158       if (dsu[i].first == 0)
159         continue;
160       int u = i;
161       c[u] = 1;
162       while (true) {
163         int source, target, value;
164         dsu[u].second = treap_getmin (dsu[u].second, source, target, value);
165         if (dsu[target] == dsu[u])
166           continue;
167         treap_add (dsu[u].second, -value);
168         ans += value;
169         jump_from[dsu[u].first] = source;
170         jump[dsu[u].first] = target;
171         if (dsu[target].first == 0)
172           break;
173         if (!c[target]) {
174           c[target] = 1;
175           u = target;
176           continue;
177         }
178         assert (k < 2 * n);
179         int node = k++, t = target;
180         parent[dsu[u].first] = node;
181         children[node].push_back (dsu[u].first);
182         dsu[u].first = node;
183         Treap *v = dsu[u].second;
184         while (dsu[t].first != node) {
185           int next = jump[dsu[t].first];
186           parent[dsu[t].first] = node;
187           children[node].push_back (dsu[t].first);
188           v = treap_merge (v, dsu[t].second);
189           dsu.merge (u, t, make_pair (node, v));
190           t = next;
191         }
192       }
193       u = i;
194       while (dsu[u].first) {
195         int next = jump[dsu[u].first];
196         finish.push_back (dsu[u].first);
197         dsu.merge (u, 0, make_pair (0, (Treap *)0));
198         u = next;
199       }
200     }
201     bool ok[k];
202     int res[n];
203     memset (ok, 0, sizeof (ok[0]) * k);
```

```
204     memset (res, -1, sizeof (res[0]) * n);
205     function <void (int, int)> add_edge = [&ok, &parent, &res, &n] ( int a, int b )
    ↪   {
206       assert (0 <= a && a < n);
207       assert (0 <= b && b < n);
208       assert (res[a] == -1);
209       res[a] = b;
210       while (a != -1 && !ok[a]) {
211         ok[a] = true;
212         a = parent[a];
213       }
214     };
215     function <void (int)> reach = [&ok, &reach, &children, &jump, &jump_from,
    ↪   &add_edge]( int u ) {
216       if (!ok[u])
217         add_edge (jump_from[u], jump[u]);
218       for (auto x : children[u])
219         reach (x);
220     };
221     for (auto x : finish)
222        reach (x);
223     printf (''%d\n'', ans);
224     for (int i = 1; i < n; i++)
225       printf (''%d%c'', res[i] ? res[i] : -1, ''\n ''[i < n - 1]);
226   }
227   return 0;
228 }
```

```
struct state {
    state() {
        std::fill(next, next + 26, -1);
    }

    int len = 0, link = -1;
    bool term = false;

    int next[26];
};

vector<state> st;
int last;

void sa_init() {
    last = 0;
    st.clear();
    st.resize(1);
}

void sa_extend (char c) {
    int cur = st.size();
    st.resize(st.size() + 1);

    st[cur].len = st[last].len + 1;
    int p;
    for (p=last; p!=-1 && st[p].next[c - 'a'] == -1; p=st[p].link)
        st[p].next[c - 'a'] = cur;
    if (p == -1)
        st[cur].link = 0;
    else {
        int q = st[p].next[c - 'a'];
        if (st[p].len + 1 == st[q].len)
            st[cur].link = q;
        else {
            int clone = st.size();
            st.resize(st.size() + 1);
            st[clone].len = st[p].len + 1;
            std::copy(st[q].next, st[q].next + 26, st[clone].next);
            st[clone].link = st[q].link;
            for (; p!=-1 && st[p].next[c - 'a']==q; p=st[p].link)
                st[p].next[c - 'a'] = clone;
            st[q].link = st[cur].link = clone;
        }
    }
    last = cur;
}

for(int v=last; v!=-1; v=st[v].link) // set termination flag.
    st[v].term = 1;
```

```
struct treap{
    map<char, int> go;
    int len, suff;
    long long sum_in;
    treap(){}
};

treap v[max_n * 4];
int last = 0;

int add_treap(int max_len){
    v[number].sum_in = 0;
    v[number].len = max_len;
    v[number].suff = -1;
    number++;
    return number - 1;
}

void add_char(char c){
    int cur = last;
    int new_treap = add_treap(v[cur].len + 1);
    last = new_treap;
    while (cur != -1){
        if (v[cur].go.count(c) == 0){
            v[cur].go[c] = new_treap;
            v[new_treap].sum_in += v[cur].sum_in;
            cur = v[cur].suff;
            if (cur == -1)
                v[new_treap].suff = 0;
        }else{
            int a = v[cur].go[c];
            if (v[a].len == v[cur].len + 1){
                v[new_treap].suff = a;
            }else{
                int b = add_treap(v[cur].len + 1);
                v[b].go = v[a].go;
                v[b].suff = v[a].suff;
                v[new_treap].suff = b;
                while (cur != -1 && v[cur].go.count(c) != 0 && v[cur].go[c] == a){
                    v[cur].go[c] = b;
                    v[a].sum_in -= v[cur].sum_in;
                    v[b].sum_in += v[cur].sum_in;
                    cur = v[cur].suff;
                }
                v[a].suff = b;
            }
            return;
        }
    }
}
```

```cpp
int k = sqrt((double)p) + 2;

for (int i = k; i >= 1; i--)
    mp[bin(b, (i * 1ll * k) % (p-1), p)] = i;

bool answered = false;
int ans = INT32_MAX;
for (int i = 0; i <= k; i++){
    int sum = (n * 1ll * bin(b, i, p)) % p;
    if (mp.count(sum) != 0){
        int an = mp[sum] * 1ll * k - i;
        if (an < p)
            ans = min(an, ans);
    }
}
```

```
fill(par, par + 301, -1);
fill(par2, par2 + 301, -1);

int ans = 0;
for (int v = 0; v < n; v++){
    memset(useda, false, sizeof(useda));
    memset(usedb, false, sizeof(usedb));
    useda[v] = true;
    for (int i = 0; i < n; i++)
        w[i] = make_pair(a[v][i] + row[v] + col[i], v);
    memset(prev, 0, sizeof(prev));
    int pos;
    while (true){
        pair<pair<int, int>, int> p = make_pair(make_pair(1e9, 1e9), 1e9);
        for (int i = 0; i < n; i++)
            if (!usedb[i])
                p = min(p, make_pair(w[i], i));
        for (int i = 0; i < n; i++)
            if (!useda[i])
                row[i] += p.first.first;
        for (int i = 0; i < n; i++)
            if (!usedb[i]){
                col[i] -= p.first.first;
                w[i].first -= p.first.first;
            }
        ans += p.first.first;
        usedb[p.second] = true;
        prev[p.second] = p.first.second; //из второй в первую
        int x = par[p.second];
        if (x == -1){
            pos = p.second;
            break;
        }
        useda[x] = true;
        for (int j = 0; j < n; j++)
            w[j] = min(w[j], {a[x][j] + row[x] + col[j], x});

    }
    while (pos != -1){
        int nxt = par2[prev[pos]];
        par[pos] = prev[pos];
        par2[prev[pos]] = pos;
        pos = nxt;
    }
}
cout << ans << ''\n'';
for (int i = 0; i < n; i++)
    cout << par[i] + 1 << '' '' << i + 1 << ''\n'';
```

```cpp
struct edge{
    int from, to;
    int c, f, num;
    edge(int from, int to, int c, int num):from(from), to(to), c(c), f(0), num(num){}
    edge(){}
};

const int max_n =  600;

edge eds[150000];
int num = 0;
int it[max_n];
vector<int> gr[max_n];
int s, t;
vector<int> d(max_n);

bool bfs(int k){
    queue<int> q;
    q.push(s);
    fill(d.begin(), d.end(), -1);
    d[s] = 0;
    while (!q.empty()){
        int v = q.front();
        q.pop();
        for (int x : gr[v])
            if (d[eds[x].to] == -1 && eds[x].c - eds[x].f >= (1 << k)){
                d[eds[x].to] = d[v] + 1;
                q.push(eds[x].to);
            }
    }

    return (d[t] != -1);
}

int dfs(int v, int flow, int k){
    if (flow < (1 << k))
        return 0;
    if (v == t)
        return flow;
    for (; it[v] < gr[v].size(); it[v]++){
        if (d[v] + 1 != d[eds[gr[v][it[v]]].to])
            continue;
        int num = gr[v][it[v]];
        int res = dfs(eds[gr[v][it[v]]].to, min(flow, eds[gr[v][it[v]]].c - eds[gr[v][it[v]]].f), k);
        if (res){
            eds[num].f += res;
            eds[num ^ 1].f -= res;
            return res;
        }
```

```
    }
    return 0;
}

void add(int fr, int to, int c, int nm){
    gr[fr].push_back(num);
    eds[num++] = edge(fr, to, c, nm);
    gr[to].push_back(num);
    eds[num++] = edge(to, fr, 0, nm); //corrected c
}

int ans = 0;
    for (int k = 30; k >= 0; k--)
        while (bfs(k)){
            memset(it, 0, sizeof(it));
            while (int res = dfs(s, 1e9 + 500, k))
                ans += res;
        }


// decomposition

int path_num = 0;
vector<int> paths[550];
int flows[550];

int decomp(int v, int flow){
    if (flow < 1)
        return 0;
    if (v == t){
        path_num++;
        flows[path_num - 1] = flow;
        return flow;
    }
    for (int i = 0; i < gr[v].size(); i++){
        int num = gr[v][i];
        int res = decomp(eds[num].to, min(flow, eds[num].f));
        if (res){
            eds[num].f -= res;
            paths[path_num - 1].push_back(eds[num].num);
            return res;
        }
    }
    return 0;
}

while (decomp(s, 1e9 + 5));
```

```
long long ans = 0;
int mx = 2 * n + 2;

memset(upd, 0, sizeof(upd));
for (int i = 0; i < mx; i++)
    dist[i] = inf;
dist[st] = 0;
queue<int> q;
q.push(st);
upd[st] = 1;
while (!q.empty()){
    int v = q.front();
    q.pop();
    if (upd[v]){
        for (int x : gr[v])        {
            edge &e = edges[x];
            if (e.c - e.f > 0 && dist[v] != inf && dist[e.to] > dist[v] + e.w) {
                dist[e.to] = dist[v] + e.w;
                if (!upd[e.to])
                    q.push(e.to);
                upd[e.to] = true;
                p[e.to] = x;
            }
        }
        upd[v] = false;
    }
}

for (int i = 0; i < k; i++){
    for (int i = 0; i < mx; i++)
        d[i] = inf;
    d[st] = 0;
    memset(used, false, sizeof(used));
    set<pair<int, int> > s;
    s.insert(make_pair(0, st));
    for (int i = 0; i < mx; i++){
        int x;
        while (!s.empty() && used[(s.begin() -> second)]){
            s.erase(s.begin());
        }
        if (s.empty())
            break;
        x = s.begin() -> second;
        used[x] = true;
        s.erase(s.begin());
        for (int i = 0; i < gr[x].size(); i++){
            edge &e = edges[gr[x][i]];
            if (!used[e.to] && e.c - e.f > 0){
                if (d[e.to] > d[x] + (e.c - e.f) * e.w + dist[x] - dist[e.to]){
                    d[e.to] = d[x] + (e.c - e.f) * e.w + dist[x] - dist[e.to];
                    p[e.to] = gr[x][i];
```

```
52                      s.insert(make_pair(d[e.to], e.to));
53                  }
54              }
55          }
56          dist[x] += d[x];
57      }
58      int pos = t;
59      while (pos != st){
60              int id = p[pos];
61              edges[id].f += 1;
62          edges[id ^ 1].f -= 1;
63          pos = edges[id].from;
64      }
65  }
```

```
string min_cyclic_shift (string s) {
        s += s;
        int n = (int) s.length();
        int i=0, ans=0;
        while (i < n/2) {
                ans = i;
                int j=i+1, k=i;
                while (j < n && s[k] <= s[j]) {
                        if (s[k] < s[j])
                                k = i;
                        else
                                ++k;
                        ++j;
                }
                while (i <= k)  i += j - k;
        }
        return s.substr (ans, n/2);
}
```

```
for(int i = 0; i<(1<<N); ++i)
        F[i] = A[i];
for(int i = 0;i < N; ++i) for(int mask = 0; mask < (1<<N); ++mask){
        if(mask & (1<<i))
                F[mask] += F[mask^(1<<i)];
}
```

```cpp
vector<int> getZ(string s){
    vector<int> z;
    z.resize(s.size(), 0);
    int l = 0, r = 0;
    for (int i = 1; i < s.size(); i++){
        if (i <= r)
            z[i] = min(r - i + 1, z[i - l]);
        while (i + z[i] < s.size() && s[z[i]] == s[i + z[i]])
            z[i]++;
        if (i + z[i] - 1 > r){
            r = i + z[i] - 1;
            l = i;
        }
    }
    return z;
}

vector<int> getP(string s){
    vector<int> p;
    p.resize(s.size(), 0);
    int k = 0;
    for (int i = 1; i < s.size(); i++){
        while (k > 0 && s[i] == s[k])
            k = p[k - 1];
        if (s[i] == s[k])
            k++;
        p[i] = k;
    }
    return p;
}

vector<int> getH(string s){
    vector<int> h;
    h.resize(s.size() + 1, 0);
    for (int i = 0; i < s.size(); i++)
        h[i + 1] = ((h[i] * 1ll * pow) + s[i] - 'a' + 1) % mod;
    return h;
}

int getHash(vector<int> &h, int l, int r){
    int res = (h[r + 1] - h[l] * p[r - l + 1]) % mod;
    if (res < 0)
        res += mod;
    return res;
}
```

suf array + lcp

```
char *s; // входная строка
int n; // длина строки
const int maxlen = ...; // максимальная длина строки
const int alphabet = 256; // размер алфавита, <= maxlen
//
int p[maxlen], cnt[maxlen], c[maxlen];
memset (cnt, 0, alphabet * sizeof(int));
for (int i=0; i<n; ++i)
        ++cnt[s[i]];
for (int i=1; i<alphabet; ++i)
        cnt[i] += cnt[i-1];
for (int i=0; i<n; ++i)
        p[--cnt[s[i]]] = i;
c[p[0]] = 0;
int classes = 1;
for (int i=1; i<n; ++i) {
        if (s[p[i]] != s[p[i-1]])  ++classes;
        c[p[i]] = classes-1;
}
//
int pn[maxlen], cn[maxlen];
for (int h=0; (1<<h)<n; ++h) {
        for (int i=0; i<n; ++i) {
                pn[i] = p[i] - (1<<h);
                if (pn[i] < 0)  pn[i] += n;
        }
        memset (cnt, 0, classes * sizeof(int));
        for (int i=0; i<n; ++i)
                ++cnt[c[pn[i]]];
        for (int i=1; i<classes; ++i)
                cnt[i] += cnt[i-1];
        for (int i=n-1; i>=0; --i)
                p[--cnt[c[pn[i]]]] = pn[i];
        cn[p[0]] = 0;
        classes = 1;
        for (int i=1; i<n; ++i) {
                int mid1 = (p[i] + (1<<h)) % n,  mid2 = (p[i-1] + (1<<h)) % n;
                if (c[p[i]] != c[p[i-1]] || c[mid1] != c[mid2])
                        ++classes;
                cn[p[i]] = classes-1;
        }
        memcpy (c, cn, n * sizeof(int));
}
```

```
//p[i] -- prefix (id of first symbol) on i-th position of suff array (from 0)
for (int i = 0; i < n; i++)
    r[p[i]] = i;
int k = 0;
for (int j = 0; j < n; j++){
    int i = r[j];
    k--;
    if (k < 0 || i == n - 1)
        k = 0;
    if (i != n - 1)
        while (s[p[i] + k] == s[p[i + 1] + k])
            k++;
    lcp[i] = k;
}
for (int i = 0; i + 1 < n; i++)
    cout << lcp[i] << '' '';
```

```
int num = 0;
long long phi = n, nn = n;
for (long long x:primes){
    if (x*x>nn)
        break;
    if (nn % x == 0){
        while (nn % x == 0)
            nn /= x;
        phi -= phi/x;
        num++;
    }
}
if (nn != 1){
    phi -= phi/nn;
    num++;
}
if (!((num == 1 && n % 2 != 0) || n == 4 || n == 2 || (num == 2 && n % 2 == 0 && n
 ↪  % 4 != 0))){
    cout << ''-1\n'';
    continue;
}
vector<long long> v;
long long pp = phi;
for (long long x:primes){
    if (x*x>pp)
        break;
    if (pp % x == 0){
        while (pp % x == 0)
            pp /= x;
        v.push_back(x);
    }
}
if (pp != 1){
    v.push_back(pp);
}
while (true){
    long long a = primes[rand()%5000]%n;
    if (gcd(a, n) != 1)
        continue;
    bool bb = false;
    for (long long x:v)
        if (pow(a, phi/x) == 1){
            bb = true;
            break;
        }
    if (!bb){
        cout << a << ''\n'';
        break;
    }
}
```

```
int log = 20;
int N = 1 << log;

typedef complex<double> cd;

int rev[N];
cd root[N];

void init() {
    for (int i = 0; i != N; ++i)
        rev[i] = (rev[i >> 1] >> 1) + ((i & 1) << (maxlog - 1));

    const double pi = acos(-1);
    for (int k = 1; k < maxn; k *= 2) {
        cd tmp(pi / k);
        root[k] = {1, 0};
        for (int i = 1; i < k; i++)
            root[k + i] = (i & 1) ? root[(k + i) >> 1] * tmp : root[(k + i) >> 1];
    }
}

void fft(vector<cd>& a) {
    for (int i = 0; i != N; ++i)
        if (rev[i] < i)
            swap(res[rev[i]], res[i]);
    for (int k = 1; k < maxn; k *= 2)
        for (int i = 0; i < maxn; i += 2 * k)
            for (int j = 0; j != k; ++j) {
                cd tmp = root[k + j] * res[i + j + k];
                res[i + j + k] = res[i + j] - tmp;
                res[i + j]     = res[i + j] + tmp;
            }
}

void inv_fft(vector<cd>& a) {
    fft(a);
    reverse(a.begin() + 1, a.end());

    for (cd& elem: a)
        elem /= N;
}
```

```
void fast_fourier(vector<int>& a) { // AND-FFT.
    for (int k = 1; k < SZ(a); k *= 2)
        for (int start = 0; start < (1 << K); start += 2 * k) {
            for (int off = 0; off < k; ++off) {
                int a_val = a[start + off];
                int b_val = a[start + k + off];

                a[start + off] = b_val;
                a[start + k + off] = add(a_val, b_val);
```

```
            }
        }
}

void inverse_fast_fourier(vector<int>& a) {
    for (int k = 1; k < SZ(a); k *= 2)
        for (int start = 0; start < (1 << K); start += 2 * k) {
            for (int off = 0; off < k; ++off) {
                int a_val = a[start + off];
                int b_val = a[start + k + off];

                a[start + off] = sub(b_val, a_val);
                a[start + k + off] = a_val;
            }
        }
}
```

DM

Кол-во корневых деревьев:
$t(G) = \frac{1}{n}\lambda_2 \dots \lambda_n$ $(\lambda_1 = 0)$

Кол-во эйлеровых циклов:
$e(D) = t^-(D, x) \cdot \prod\limits_{y \in D} (outdeg(y) - 1)!$

Наличие совершенного паросочетания:
T – матрица с нулями на диагонали. Если есть ребро (i, j), то $a_{i,j} := x_{i,j}$, $a_{j,i} = -x_{i,j}$
$\det(T) = 0 \Leftrightarrow$ нет совершенного паросочетания.