NeshesTeam - UT

BASE

_vimrc:

```
cd ~/Documents/Contest/

set sw=4
set ts=4

set si           " super indentation
set number       " line numbers
syntax on        " syntax highlighting
set cursorline   " highlight current line

set guifont=consolas:h11

set bs=2

set mouse=a      " mouse works normally
set gdefault     " global replacement

set fdm=indent           " folding method
set foldlevelstart=99    " at first all folds are open
```

Template:

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

#define pb push_back
#define mp make_pair
#define SQR(a) ((a) * (a))
#define SZ(x) ((int) (x).size())
#define ALL(x) (x).begin(), (x).end()
#define CLR(x, a) memset(x, a, sizeof x)
#define VAL(x) #x << " = " << (x) << "   "
#define FOREACH(i, x) for(__typeof((x).begin()) i = (x).begin(); i != (x).end(); i ++)
#define FOR(i, n) for (int i = 0; i < (n); i ++)
#define X first
#define Y second

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

const int MAXN = 1 * 1000 + 10;

int main () {
        ios::sync_with_stdio(false);
        return 0;
}
```

GRAPH

LCA:

```
void dfs(int x) {
        mark[x] = true;
        for (int i = 0; i < SZ(adj[x]); i ++) {
                int v = adj[x][i];
                if (!mark[v]) {
                        par[v][0] = x, h[v] = h[x] + 1;
                        dfs(v);
                }
        }
}

int get_parent(int x, int k) {
        for (int i = 0; i < MAXL; i ++)
                if ((1 << i) & k) x = par[x][i];
        return x;
}

int lca(int x, int y) {
        if (h[y] > h[x]) swap(x, y);
        x = get_parent(x, h[x] - h[y]);

        if (x == y) return x;

        for (int i = MAXL - 1; i >= 0; i--)
                if (par[x][i] != par[y][i])
                        x = par[x][i], y = par[y][i];
        return par[x][0];
}

for (int i = 1; i < MAXL; i ++)
        for (int j = 0; j < n; j ++)
                par[j][i] = par[par[j][i - 1]][i - 1];
```

Cut Edge:

```
void dfs(int x, int par, int len) {
        mark[x] = true, backEdge[x] = 1e9, h[x] = len;

        for (int i = 0; i < SZ(adj[x]); i ++) {
                int v = adj[x][i].X, idx = adj[x][i].Y;
                if (mark[v] && v != par) backEdge[x] = min(backEdge[x], h[v]);
                else if (!mark[v]) {
                        dfs(v, x, len + 1);
                        int tmp = backEdge[v];
                        if (tmp > h[x]) cut[idx] = true;
                        backEdge[x] = min(backEdge[x], tmp);
                }
        }
}
```

Cut Vertex:

```
void dfs(int x, int par, int dep) {
        mark[x] = true; h[x] = dep;
        edge[x] = 1e9;

        bool check = false;
        int cnt = 0;
        for (int i = 0; i < SZ(adj[x]); i ++) {
                int v = adj[x][i]; if (v == par) continue;
                if (mark[v]) edge[x] = min(edge[x], h[v]);
                else {
                        cnt ++;
                        dfs(v, x, dep + 1);
                        if (edge[v] >= dep) check = true;
                        edge[x] = min(edge[x], edge[v]);
                }
        }

        ans[x] = check;
        if (par == -1 && cnt < 2) ans[x] = false;
}
```

SCC:

```
vector <int> adj[N];
stack <int> S, P;
int mrk[N], ind, col[N], CL;

void dfs(int v) {
        mrk[v] = ++ind;
        S.push(v);
        P.push(v);
        for(int i = 0;i < Size(adj[v]);++i) {
                int u = adj[v][i];
                if(!mrk[u])
                        dfs(u);
                else
                        while(mrk[u] < mrk[S.top()])
                                S.pop();
        }
        if(S.top() == v) {
                mrk[v] = INF;
                col[v] = ++CL;
                while(P.top() != v) {
                        col[P.top()] = CL;
                        mrk[P.top()] = INF;
                        P.pop();
                }
                P.pop();
                S.pop();
        }
}

//main: for(int i = 1;i <= n;++i)
//                      if(!mrk[i]) dfs(i);
```

Matching:

```
bool dfs(int x) {
        if (mark[x]) return false;

        mark[x] = true;
        for (int i = 0; i < SZ(adj[x]); i ++) {
                int v = adj[x][i];
                if (match[1][v] == -1 || dfs(match[1][v])) {
                        match[0][x] = v;
                        match[1][v] = x;
                        return true;
                }
        }
        return false;
}

void bi_match() {
        CLR(match, -1);
        for (int i = 0; i < n; i ++) {
                CLR(mark, 0);
                bool check = false;
                for (int j = 0; j < n; j ++)
                        if (!mark[j] && match[0][j] == -1)
                                check |= dfs(j);
                if (!check) break;
        }
}
```

Bellman Ford:

```
bool bellman(int start) {
        FOR(i, n) d[i] = INF;
        d[start] = 0;

        FOR(i, n - 1) FOR(j, m) {
                int x = ex[j], y = ey[j]; double w = tw[j];
                d[y] = min(d[y], d[x] + w);
        }

        // check if graph has a negative cycle
        FOR(i, m) {
                int x = ex[i], y = ey[i]; double w = tw[i];
                if (d[y] > d[x] + w) return false;
        }

        return true;
}
```

Max Flow:

```
const int N = 100;
int viz[N], mat[N][N], network[N][N], parent[N];
bool anotherPath(int start, int end) {
        memset(viz, 0, sizeof viz);
        memset(parent, -1, sizeof parent);
        viz[start] = true;
        queue<int> q;
        q.push(start);
        while (!q.empty()) {
                int z = q.front(); q.pop();
                viz[z] = true;
                for (int i=0;i<N;i++) {
                        if (network[z][i] <= 0 || viz[i]) continue;
                        viz[i] = true;
                        parent[i] = z;
                        if (i == end) return true;
                        q.push(i);
                }
        }
        return false;
}
int maxflow(int start, int end) {
        memcpy(network, mat, sizeof(mat));
        int total = 0;
        while (anotherPath(start, end)) {
                int flow = network[parent[end]][end];
                int curr = end;
                while (parent[curr] >= 0) {
                        flow = min(flow, network[parent[curr]][curr]);
                        curr = parent[curr];
                }
                curr = end;
                while (parent[curr] >= 0) {
                        network[parent[curr]][curr]-=flow;
                        network[curr][parent[curr]]+=flow;
                        curr = parent[curr];
                }
                total += flow;
        }
        return total;
}
```

DSU:

```
int par[MAXN];

int father(int x) {
        return par[x] == -1 ? x : par[x] = father(par[x]);
}
void merge(int x, int y) {
        x = father(x);
        y = father(y);
        if (x != y) par[y] = x;
}

for (int i = 0; i < n; i ++) par[i] = -1;
```

Dijkstra:

```
void dij(int start) {
	for (int i = 0; i < MAXN; i ++) dis[i] = INF;

	CLR(mark, 0); s.clear();
	mark[start] = true;
	dis[start] = 0;
	s.insert(mp(0, start));

	while (SZ(s)) {
		int x = s.begin()->Y; s.erase(s.begin());

		for (int i = 0; i < SZ(adj[x]); i ++) {
			int v = adj[x][i].X, w = adj[x][i].Y;
			if (dis[v] > dis[x] + w) {
				if (mark[v]) s.erase(mp(dis[v], v));
				else mark[v] = true;

				dis[v] = dis[x] + w;
				s.insert(mp(dis[v], v));
			}
		}
	}
}
```

Prim:

```
const int N = 1000 * 100 + 5;
vector <pii> adj[N];
int ans, mrk[N];

void prim(int v) {
	int w;
	set <pii> st;
	st.insert(mp(0, v));
	while(!st.empty()) {
		v = st.begin()-> Y;
		w = st.begin()-> X;
		st.erase(st.begin());
		if(mrk[v]++) continue;
		ans += w;

		for(int i = 0;i < Size(adj[v]);++i)
			if(!mrk[adj[v][i].Y])
				st.insert(adj[v][i]);
	}
}
```

GEOMETRY

Intersection:

```
typedef long double ld;
typedef complex <ld> point;

const ld EPS = 1e-9;

ld operator ^(const point &a, const point &b) {
        return imag(a * conj(b));
}

ld operator |(const point &a, const point &b) {
        return real(a * conj(b));
}

bool in_between(const point &a, const point &b, const point &c) {
        if(norm(a - c) <= 2 * EPS || norm(b - c) <= 2 * EPS)
                return true;
        ld t = norm(a - b);
        return (abs((a - b) ^ (c - b)) <= EPS) && (norm(a - c) <= t) && (norm(b - c) <= t);
}

point intersection(const point &a, const point &b, const point &c, const point &d) {
        if(abs((b - a) ^ (d - c)) <= EPS) {
                if(in_between(a, b, c))
                        return c;
                if(in_between(a, b, d))
                        return d;
                //if(in_between(c, d, a))
                        return a;
        }
        ld s = ((c - a) ^ (d - c)) / ((b - a) ^ (d - c));
        return a + s * (b - a);
}
```

Convex Hull:

```
typedef complex <ld> point;

const ld EPS = 1e-12;
vector <point> vl;//convex hull vector
point p[N], O;//

ld operator ^(const point &a, const point &b) {
        return imag(a * conj(b));
}

ld operator |(const point &a, const point &b) {
        return real(a * conj(b));
}

bool cmp(const point &a, const point &b) {
        return (((a - O) ^ (b - O)) < (ld)0) || (((a - O) ^ (b - O)) == 0 && norm(a - O) < norm(b
- O));
}

inline bool chk(const point &a, const point &b, const point &c) {
        return (((b - c) ^ (a - c)) >= (ld)0);
}

void find_hull(int n) {
        int ind = 0;
        for(int i = 0;i < n;++i)
                if(p[i].Y < p[ind].Y || (p[i].Y == p[ind].Y && p[i].X < p[ind].X))
                        ind = i;
        swap(p[0], p[ind]);
        O = p[0];
        sort(p + 1, p + n, cmp);
        vl.push_back(p[0]);
        for(int i = 1;i < n;++i) {
                while(Size(vl) > 1 && chk(p[i], vl.back(), vl[Size(vl) - 2]))
                        vl.pop_back();
                vl.push_back(p[i]);
        }
        while(Size(vl) > 1 && chk(p[0], vl.back(), vl[Size(vl) - 2]))
                vl.pop_back();
}
```

DATA STRUCTURE

Fenwick1:

```
int fen[MAXN]; // 0-based, [)

void add(int x, int val = 1) {
        for (int i = x + 1; i < MAXN; i += i & (-i))
                fen[i] += val;
}

int get(int x) {
        int ans = 0;
        for (int i = x; i > 0; i -= i & (-i))
                ans += fen[i];
        return ans;
}

int sum(int x, int y) {
        return get(y) - get(x);
}
```

Fenwick2:

```
void add(int x, int val) {
        for (int i = x; i > 0; i -= i & (-i))
                fen[i] += val;
}

int get(int x) {
        int ans = 0;
        for (int i = x + 1; i < MAXN; i += i & (-i))
                ans += fen[i];
        return ans;
}

void update(int l, int r, int val) {
        add(r, +val);
        add(l, -val);
}
```

RMQ:

```
const int N = 1000 * 100 + 5, LOG = 20;

class RMQ{
        int f[LOG][N], Lgl[N], S;
public:
        RMQ() {
                for(int i = 1, p = 0;i < N;++i) {
                        if(i == 1 << (p + 1))
                                ++p;
                        Lgl[i] = p;
                }
        }
        void build(int a[], int n) {
                for(int i = 0;i < n;++i)
                        f[0][i] = a[i];

                for(int j = 1, p = 1; j < LOG; ++j, p *= 2)
                        for(int i = 0;i < n;++i) {
                                f[j][i] = f[j - 1][i];
                                if(i + p < n)
                                        f[j][i] = min(f[j - 1][i], f[j - 1][i + p]);
                        }
        }
        int find(int s, int e) {
                int l = Lgl[e - s + 1];
                return min(f[l][s], f[l][e + 1 - (1 << l)]);
        }
};
```

Segment Vector optimized:

```
// template <class T1>

class Segment{
        vector <int> Rb[4 * N], Rs[4 * N], Lb[4 * N], Ls[4 * N];
        int P[4 * N], Q[4 * N];
public:
        vector <ll> S[4 * N];

        void build(int x, int p, int q, ll a[]) {
                P[x] = p;
                Q[x] = q;
                for(int i = p;i <= q;++i)
                        S[x].push_back(a[i]);
                if(p == q)
                        return ;

                int m = (p + q) >> 1, r = x << 1, l = (x << 1) + 1, po = 0, qo = 0;
                build(x << 1, p, m, a);
                build((x << 1) + 1, m + 1, q, a);
                ///
                sort(S[x].begin(), S[x].end());

                for(int i = 0;i < Size(S[x]);++i) {
                        while(po < Size(S[r]) && S[r][po] < S[x][i])
                                ++po;
                        while(qo < Size(S[l]) && S[l][qo] < S[x][i])
                                ++qo;
                        Rb[x].push_back(po);
                        Lb[x].push_back(qo);
                }

                po = Size(S[r]) - 1;
                qo = Size(S[l]) - 1;
                for(int i = Size(S[x]) - 1;i > -1;--i) {
                        while(po > -1 && S[r][po] > S[x][i])
                                --po;
                        while(qo > -1 && S[l][qo] > S[x][i])
                                --qo;
                        Rs[x].push_back(po);
                        Ls[x].push_back(qo);
                }
                reverse(Rs[x].begin(), Rs[x].end());
                reverse(Ls[x].begin(), Ls[x].end());
        }
        int find(int x, int p, int q, int a, int b) {
                if(a == Size(S[x]) || b == -1 || a > b)
                        return 0;
                if(q < P[x] || Q[x] < p)
                        return 0;
                if(p <= P[x] && Q[x] <= q)
                        return b - a + 1;
                int tmp = find(x << 1, p, q, Rb[x][a], Rs[x][b]);
                tmp += find((x << 1) + 1, p, q, Lb[x][a], Ls[x][b]);
                return tmp;
        }
} T;
```

STRINGS

Hash:

```
const int MAXN = 100 * 1000 + 10, BASE = 701;

ll p[MAXN], hash[MAXN];

int main () {
        p[0] = 1;
        for (int i = 1; i < MAXN; i ++)
                p[i] = p[i - 1] * BASE;

        for (int i = 1; i <= SZ(s); i ++)
                hash[i] = hash[i - 1] * BASE + s[i - 1];

        // hash in [i, j], 1-based
        ll h = hash[j] - (hash[i - 1] * p[j - i + 1]);
}
```

KMP:

```
int f[M];
string s,t;
bool match[M];
void kmp() {
        f[0] = -1;
        int pos = -1;
        for (int i = 1; i <= SZ(t); i++) {
                while(pos != -1 && t[pos] != t[i - 1]) pos = f[pos];
                f[i] = ++pos;
        }
        pos = 0;
        for (int i = 0; i < SZ(s); i++) {
                while(pos != -1 && (pos == SZ(t) || s[i] != t[pos])) pos = f[pos];
                pos ++;
                if (pos == SZ(t)) match[i] = 1;
                else match[i] = 0;
        }
}
```

Suffix Array:

```
const int N = 1000 * 100 + 5;

namespace Suffix{
        int sa[N], rank[N], lcp[N], gap, S;
        bool cmp(int x, int y) {
                if(rank[x] != rank[y])
                        return rank[x] < rank[y];
                x += gap, y += gap;
                return (x < S && y < S)? rank[x] < rank[y]: x > y;
        }
        void sa_build(const string &s) {
                S = Size(s);
                int tmp[N] = {0};
                for(int i = 0;i < S;++i)
                        rank[i] = s[i],
                        sa[i] = i;
                for(gap = 1;;gap <<= 1) {
                        sort(sa, sa + S, cmp);
                        for(int i = 1;i < S;++i)
                                tmp[i] = tmp[i - 1] + cmp(sa[i - 1], sa[i]);
                        for(int i = 0;i < S;++i)
                                rank[sa[i]] = tmp[i];
                        if(tmp[S - 1] == S - 1)
                                break;
                }
        }
        void lcp_build() {
                for(int i = 0, k = 0;i < S;++i, --k)
                        if(rank[i] != S - 1) {
                                k = max(k, 0);
                                while(s[i + k] == s[sa[rank[i] + 1] + k])
                                        ++k;
                                lcp[rank[i]] = k;
                        }
                        else
                                k = 0;
        }
};
```

NUMBER THEORY

Phi:

```
void find_phi(int n) {
        phi[1] = 1;
        for(int i = 2;i < n;++i) {
                if(lp[i] == i)
                        phi[i] = i - 1;
                else {
                        phi[i] = phi[lp[i]] * phi[(i / lp[i])];
                        if(lp[i / lp[i]] == lp[i])
                                phi[i] *= lp[i], phi[i] /= (lp[i] - 1)
                }
        }
}
```

370:

```
bool mark[MAXN];

vector<int> dv[MAXN];
int n, m;

int f(int x) {
        int res = 0;
        for (int mask = 0; mask < (1 << SZ(dv[x])); mask ++) {
                int t = __builtin_popcount(mask), a = n - 1;
                for (int i = 0; i < SZ(dv[x]); i ++)
                        if (mask & (1 << i))
                                a /= dv[x][i];
                if (t & 1) res -= a;
                else res += a;
        }
        return res;
}

int main() {
        cin >> n >> m;
        for (int i = 2; i < n; i ++)
                if (!mark[i]) {
                        for (int j = i; j < m; j += i) {
                                mark[j] = true;
                                dv[j].pb(i);
                        }
                }
        ll ans = 2;
        for (int i = 1; i < m; i ++) ans += f(i);
        cout << ans << endl;
        return 0;
}
```

OTHER THINGS

Read Input:

```
inline int read() {
        bool minus = false;
        int result = 0;
        char ch;
        ch = getchar();
        while (true) {
                if (ch == '-') break;
                if (ch >= '0' && ch <= '9') break;
                ch = getchar();
        }
        if (ch == '-') minus = true; else result = ch-'0';
        while (true) {
                ch = getchar();
                if (ch < '0' || ch > '9') break;
                result = result*10 + (ch - '0');
        }
        if (minus)
                return -result;
        else
                return result;
}
```

LIS:

```
for (int i = 0; i <= n; i ++) c[i] = 1e9;

int ans = 0;
for (int i = 0; i < n; i ++) {
        int l = 0, r = i + 1;
        while (r - l > 1) {
                int mid = (l + r) / 2;
                if (c[mid] <= a[i]) l = mid;
                else r = mid;
        }
        ans = max(ans, l + 1);
        if (c[l + 1] > a[i]) c[l + 1] = a[i];
}
```

DFS non-returning:

```
void DFS(int v){
        stack <int> S;
        S.push(v);
        while(!S.empty()){
                while(!S.empty()  &&  mrk[S.top()]++) S.pop();
                if(S.empty()) break;
                v = S.top(), S.pop();
                for(int i = 0; i< Size(adj[v]); ++i)
                        if(!mrk[adj[v][i]])
                                S.push(v);
        }
}
```

Divide and Conquer Tree:

```
void dfs(int v, int p) {
        h[v] = 1;
        for(int u:adj[v])
                if(u != p && mrk[u]) {
                        dfs(u, v);
                        h[v] += h[u];
                }
}

void algo(int v, char c) {
        dfs(v, v);
        int S = h[v], p = v;
sign:
        for(int u:adj[v])
                if(mrk[u] && u != p && h[u] > S / 2) {
                        p = v;
                        v = u;
                        goto sign;
                }
        mrk[v] = 0;
        ans[v] = c;
        for(int u:adj[v])
                if(mrk[u])
                        algo(u, c + 1);
}
```

Divide and Conquer Optimization:

```
void solve(int x, int y, int s, int e) {
        if(x > y)
                return ;
        int m = (x + y) >> 1, t;
        dp[m][l] = INF;
        for(int i = s;i <= min(e, m);++i) {
                if(dp[i][l - 1] + F[i + 1][m] < dp[m][l]) {
                        dp[m][l] = dp[i][l - 1] + F[i + 1][m];
                        t = i;
                }
        }
        solve(x, m - 1, s, t);
        solve(m + 1, y, t, e);
}
```

Knuth Optimization:

```
for(int j = 2;j <= n;++j) {
        for(int i = n;i >= j;--i) {
                int M = INF;
                for(int s = u[i][j - 1];s <= min(u[i + 1][j], i);++s) {
                        if(dp[s - 1][j - 1] + F[s][i] < M) {
                                M = dp[s - 1][j - 1] + F[s][i];
                                u[i][j] = s;
                        }
                }
                dp[i][j] = M;
        }
}
```

Lower Envelop Otimization:

```
deque <pii> dq;
pii a[N], b[N];
ll dp[N];
bool mrk[N];

for(int i = 0;i < p;++i) {
        while(Size(dq) > 1 && (dp[i] - dq.back().Y) * (dq[Size(dq) - 2].X - b[i].Y) < (dp[i] -
dq[Size(dq) - 2].Y) * (dq.back().X - b[i].Y))
                dq.pop_back();
        dq.push_back(mp(b[i].Y, dp[i]));
        while(Size(dq) > 1 && dq[0].X * b[i].X + dq[0].Y >= dq[1].X * b[i].X + dq[1].Y)
                dq.pop_front();
        dp[i + 1] = dq[0].X * b[i].X + dq[0].Y;
}
```