Team notebook

September 2, 2023

Contents

1 algorithms

1.1 binarysearch

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

```cpp
template<class T, class F>
T first_true(T l, T r, F&& f) {
    l--, r++;
    while(r - l > 1) {
        T mid = midpoint(l, r); // l + (r-l)/2
        if(f(mid)) {
            l = mid;
        } else {
            r = mid;
        }
   }
    return r;
}

int main() {
}
```

2 combinatorics

2.1 binomial

```cpp
// Binomial coefficient modulo large prime

//First we precompute all factorials modulo m
factorial[0] = 1;
for (int i = 1; i <= MAXN; i++) {
    factorial[i] = factorial[i - 1] * i % m;
}

// And afterwards we can compute the binomial coefficient in O(log m)
    time.
long long binomial_coefficient(int n, int k) {
    return factorial[n] * inverse(factorial[k] * factorial[n - k] % m) %
        m;
}
```

3 data-structures

3.1 fenwicktree

```cpp
#include<bits/stdc++.h>

template <typename T>
struct Fenwick {
    int n;
    std::vector<T> a;

    Fenwick(int n = 0) {
        init(n);
    }

    void init(int n) {
        this->n = n;
        a.assign(n, T());
    }

    void add(int x, T v) {
        for (int i = x + 1; i <= n; i += i & -i) {
            a[i - 1] += v;
        }
    }

    T sum(int x) {
        auto ans = T();
        for (int i = x; i > 0; i -= i & -i) {
            ans += a[i - 1];
        }
        return ans;
    }

    T rangeSum(int l, int r) {
        return sum(r) - sum(l);
    }

    int kth(T k) {
        int x = 0;
        for (int i = 1 << std::__lg(n); i; i /= 2) {
            if (x + i <= n && k >= a[x + i - 1]) {
                x += i;
                k -= a[x - 1];
            }
        }
        return x;
    }
```

```
};

int main() {
    return 0;
}
```

3.2 segtree

```
#include <bits/stdc++.h>

using namespace std;

struct segtree {
    using T = int;
    using F = int;

    T e() {
        return (int) 1e9;
    }

    F id() {
        return 0;
    }

    T op(T a, T b) {
        return min(a, b);
    }

    T mapping(F f, T x) {
        return f + x;
    }

    F composition(F f, F g) {
        return f + g;
    }

    int n;
    int size;
    int log_size;
    vector<T> node;
    vector<F> lazy;

    segtree() : segtree(0) {}
    segtree(int _n) {
        build(vector<T>(_n, e()));
    }
    segtree(const vector<T>& v) {
        build(v);
    }

    void build(const vector<T>& v) {
        n = (int) v.size();
        if (n <= 1) {
            log_size = 0;
        } else {
            log_size = 32 - __builtin_clz(n - 1);
        }
        size = 1 << log_size;
        node.resize(2 * size, e());
        lazy.resize(size, id());
        for (int i = 0; i < n; i++) {
            node[i + size] = v[i];
        }
        for (int i = size - 1; i > 0; i--) {
            pull(i);
        }
    }

    void push(int x) {
        node[2 * x] = mapping(lazy[x], node[2 * x]);
        node[2 * x + 1] = mapping(lazy[x], node[2 * x + 1]);
        if (2 * x < size) {
            lazy[2 * x] = composition(lazy[x], lazy[2 * x]);
            lazy[2 * x + 1] = composition(lazy[x], lazy[2 * x + 1]);
        }
        lazy[x] = id();
    }

    void pull(int x) {
        node[x] = op(node[2 * x], node[2 * x + 1]);
    }

    void set(int p, T v) {
        assert(0 <= p && p < n);
        p += size;
        for (int i = log_size; i >= 1; i--) {
            push(p >> i);
        }
```

```
        node[p] = v;
        for (int i = 1; i <= log_size; i++) {
            pull(p >> i);
        }
    }

    T get(int p) {
        assert(0 <= p && p < n);
        p += size;
        for (int i = log_size; i >= 1; i--) {
            push(p >> i);
        }
        return node[p];
    }

    T get(int l, int r) {
        assert(0 <= l && l <= r && r <= n);
        l += size;
        r += size;
        for (int i = log_size; i >= 1; i--) {
            if (((l >> i) << i) != l) {
                push(l >> i);
            }
            if (((r >> i) << i) != r) {
                push((r - 1) >> i);
            }
        }
        T vl = e();
        T vr = e();
        while (l < r) {
            if (l & 1) {
                vl = op(vl, node[l++]);
            }
            if (r & 1) {
                vr = op(node[--r], vr);
            }
            l >>= 1;
            r >>= 1;
        }
        return op(vl, vr);
    }

    void apply(int p, F f) {
        assert(0 <= p && p < n);
        p += size;
```

```
        for (int i = log_size; i >= 1; i--) {
            push(p >> i);
        }
        node[p] = mapping(f, node[p]);
        for (int i = 1; i <= log_size; i++) {
            pull(p >> i);
        }
    }

    void apply(int l, int r, F f) {
        assert(0 <= l && l <= r && r <= n);
        l += size;
        r += size;
        for (int i = log_size; i >= 1; i--) {
            if (((l >> i) << i) != l) {
                push(l >> i);
            }
            if (((r >> i) << i) != r) {
                push((r - 1) >> i);
            }
        }
        int ll = l;
        int rr = r;
        while (l < r) {
            if (l & 1) {
                node[l] = mapping(f, node[l]);
                if (l < size) {
                    lazy[l] = composition(f, lazy[l]);
                }
                l++;
            }
            if (r & 1) {
                r--;
                node[r] = mapping(f, node[r]);
                if (l < size) {
                    lazy[r] = composition(f, lazy[r]);
                }
            }
            l >>= 1;
            r >>= 1;
        }
        l = ll;
        r = rr;
        for (int i = 1; i <= log_size; i++) {
            if (((l >> i) << i) != l) {
```

```
                pull(l >> i);
            }
            if (((r >> i) << i) != r) {
                pull((r - 1) >> i);
            }
        }
    }
};

int main() {
    return 0;
}
```

3.3 simplebittree

```
#include <bits/stdc++.h>

using namespace std;

#define LSOne(S) ((S) & -(S))
typedef vector<int> vi;

// Simple version
class Bitree {

private:
    vi ft;
public:
    Bitree(int m) { ft.assign(m+1, 0); }

    int rsq(int j) {
        int sum = 0;
        for(; j; j -= LSOne(j))
            sum += ft[j];
        return sum;
    }

    int rsq(int i, int j) {
        return rsq(j) - rsq(i-1);
    }

    void update(int i, int v) {
        for(; i < (int)ft.size(); i += LSOne(i))
            ft[i] += v;
    }

};

// implementation
int main() {
    Bitree bt(10);
    bt.update(1, 0);
    bt.update(2, 1);
    bt.update(3, 0);
    bt.update(4, 1);
    bt.update(5, 2);
    bt.update(6, 3);
    bt.update(7, 2);
    bt.update(8, 1);
    bt.update(9, 1);
    cout<<"rsq(1): "<<bt.rsq(1)<<"\n";
    cout<<"rsq(2): "<<bt.rsq(2)<<"\n";
    cout<<"rsq(3): "<<bt.rsq(3)<<"\n";
    cout<<"rsq(4): "<<bt.rsq(4)<<"\n";
    cout<<"rsq(5): "<<bt.rsq(5)<<"\n";
    cout<<"rsq(6): "<<bt.rsq(6)<<"\n";
    cout<<"rsq(7): "<<bt.rsq(7)<<"\n";
    cout<<"rsq(8): "<<bt.rsq(8)<<"\n";
    cout<<"rsq(9): "<<bt.rsq(9)<<"\n";
    cout<<LSOne(8)<<"\n";
}
```

3.4 ufds

```
#include <bits/stdc++.h>

using namespace std;

// 1-indexed
class Ufds {

private:
vector<int> ps, size;
int numSets;

public:
```

```
    Ufds(int N) {
        ps.assign(N+1, 0); iota(ps.begin(), ps.end(), 0);
        size.assign(N+1, 1);
        numSets = N;
    }

    int findSet(int i) {
        return ps[i] == i ? i : (ps[i] = findSet(ps[i]));
    }

    bool sameSet(int i, int j) {
        return findSet(i) == findSet(j);
    }

    int getSetSize(int i) { return size[findSet(i)]; }
    int getNumSets() { return numSets; }

    // unify two sets
    void unionSet(int i, int j) {
        if(sameSet(i, j)) return;

        int pi = findSet(i);
        int pj = findSet(j);

        if(size[pi] > size[pj]) swap(pi, pj);

        ps[pi] = pj;
        size[pj] += size[pi];

        --numSets;
    }

};

// implementation
int main() {
    Ufds uf(5);
    cout<<"Num of sets: "<<uf.getNumSets()<<"\n";
    uf.unionSet(1, 2);
    cout<<"Num of sets: "<<uf.getNumSets()<<"\n";
    uf.unionSet(3, 4);
    cout<<"Num of sets: "<<uf.getNumSets()<<"\n";
    uf.unionSet(5, 4);
    cout<<"Num of sets: "<<uf.getNumSets()<<"\n";
    cout<<uf.sameSet(1, 4)<<"\n";
    cout<<uf.sameSet(5, 4)<<"\n";

    for(int i=1; i<=5; i++) {
        printf("findSet(%d) = %d, size(%d) = %d\n",
                i, uf.findSet(i), i, uf.getSetSize(i));
    }
}
```

4 dynamic-programming

4.1 knapsack1

```
#include <bits/stdc++.h>

#define pb(x) push_back(x)
#define all(x) x.begin(),x.end()

using namespace std;
using ll = int64_t;

void solve() {
    int N, W;
    cin>>N>>W;
    vector<int> w(N+1), v(N+1);
    for(int i=1; i<=N; i++) {
        cin>>w[i]>>v[i];
    }

    vector<vector<ll>> dp(N+1, vector<ll>(W+1, 0));

    for(int i=1; i<=N; i++) {
        for(int j=0; j<=W; j++) {
            if(w[i] <= j)
                dp[i][j] = max(dp[i-1][j], v[i] + dp[i-1][max(j-w[i], 0)]);
            else
                dp[i][j] = dp[i-1][j];
        }
    }
    cout<<dp[N][W]<<"\n";
}
```

```
int main ()
{
    std::ios::sync_with_stdio(false);
    std::cin.tie(nullptr);

    int t = 1;
    while (t--) solve();
    return 0;
}
```

5 geometry

5.1 triangles

Let a, b, c be length of the three sides of a triangle.

$$p = (a + b + c) * 0.5$$

The inradius is defined by:

$$iR = \sqrt{\frac{(p-a)(p-b)(p-c)}{p}}$$

The radius of its circumcircle is given by the formula:

$$cR = \frac{abc}{\sqrt{(a+b+c)(a+b-c)(a+c-b)(b+c-a)}}$$

6 graphs

6.1 bfs

```
#include <bits/stdc++.h>

using namespace std;

int main() {
    vector<vector<int>> adj;
    int n; int s; // nodes and source

    queue<int> q;
    vector<bool> used(n);
    vector<int> d(n, 0), p(n); // distance and parents

    q.push(s);
    used[s] = true;
    p[s] = -1; // root
    d[s] = 0;
    while(!q.empty()) {
        int v = q.front();
        q.pop();
        for(auto u : adj[v]) {
            if(!used[u]) {
                q.push(u);
                used[u] = true;
                p[u] = v;
                d[u] = ++d[v];
            }
        }
    }

    // showing the shortest path
    int u;
    if(!used[u]) {
        cout<<"No path!\n";
    } else {
        vector<int> path;
        for(int v=u; v != -1; v=p[v])
            path.push_back(v);
        reverse(path.begin(), path.end());
        for(int v: path)
            cout<<v<<" ";
    }
    return 0;
}
```

6.2 dfs

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

int n = 6;
vector<vector<int>> adj(n);
vector<bool> visited(n);
```

```cpp
int main() {
    vector<int> col(n);
    col[0] = 0;
    auto dfs = [&](int u, int p, auto&& dfs) -> void {
        for (int v : adj[u])
            if (v != p) {
                col[v] = col[u] ^ 1;
                dfs(v, u, dfs);
            }
    };
    dfs(0, -1, dfs);
}
```

6.3 dijkstra

```cpp
/**
 *   author: mralves
 *   created: 11-05-2023 21:24:59
**/
#include <bits/stdc++.h>

using namespace std;
using ll = int64_t;

const int INF = 1000000000;

vector<vector<pair<int, int>>> adj;

void dijkstra(int s, vector<int> &d, vector<int> &p) {
    int n = adj.size();
    d.assign(n, INF);
    p.assign(n, -1);
    vector<bool> u(n, false); // used
    d[s] = 0;

    for(int i=0; i<n; i++) {
        int v = -1;
        for(int j =0; j<n; j++) {
            if(!u[j] && (v == -1 || d[j] < d[v]))
                v = j;
        }

        if(d[v] == INF)
            break;

        u[v] = true;

        for(auto edge: adj[v]) {
            int to = edge.first;
            int len = edge.second;

            if(d[v] + len < d[to]) {
                d[to] = d[v] + len;
                p[to] = v;
            }
        }
    }
}

int main ()
{
    // simulation
    adj = {
        {},
        {{2, 6}, {4, 1}},
        {{1, 6}, {3, 5}, {4, 2}},
        {{2, 5}, {5, 5}},
        {{1, 1}, {2, 2}, {5, 1}},
        {{3, 5}, {4, 1}}
    };

    // 1 to 4
    int start = 1, end = 5;
    vector<int> d, p;
    dijkstra(1, d, p);

    cout<<d[end]<<"\n";

    vector<int> path;
    for(int i = end; i != -1; i = p[i]) {
        path.push_back(i);
    }

    reverse(path.begin(), path.end());

    for(auto x : path) {
        cout<<x<<" ";
    }
```

```cpp
    cout<<"\n";

    return 0;
}
```

6.4 kruskal

```cpp
//Just as in the simple version of the Kruskal algorithm, we sort all the
    edges of the graph in non-decreasing order of weights.
//Then put each vertex in its own tree (i.e. its set) via calls to the
    make_set function - it will take a total of $O(N)$.
//We iterate through all the edges (in sorted order) and for each edge
    determine whether the ends belong to different trees (with two
    find_set calls in $O(1)$ each).
//Finally, we need to perform the union of the two trees (sets), for
    which the DSU union_sets function will be called - also in $O(1)$. So
    we get the total time complexity of $O(M \log N + N + M)$ = $O(M \log
    N)$.

ivector<int> parent, rank;

void make_set(int v) {
    parent[v] = v;
    rank[v] = 0;
}

int find_set(int v) {
    if (v == parent[v])
        return v;
    return parent[v] = find_set(parent[v]);
}

void union_sets(int a, int b) {
    a = find_set(a);
    b = find_set(b);
    if (a != b) {
        if (rank[a] < rank[b])
            swap(a, b);
        parent[b] = a;
        if (rank[a] == rank[b])
            rank[a]++;
    }
}

struct Edge {
    int u, v, weight;
    bool operator<(Edge const& other) {
        return weight < other.weight;
    }
};

int n;
vector<Edge> edges;

int cost = 0;
vector<Edge> result;
parent.resize(n);
rank.resize(n);
for (int i = 0; i < n; i++)
    make_set(i);

sort(edges.begin(), edges.end());

for (Edge e : edges) {
    if (find_set(e.u) != find_set(e.v)) {
        cost += e.weight;
        result.push_back(e);
        union_sets(e.u, e.v);
    }
}
```

7 misc

7.1 alias

alias cmp="g++ -std=c++20 -Wall -Wshadow -fsanitize=address -D DEBUG"
alias pbcopy="xclip -selection clipboard"
alias pbpaste="xclip -selection clipboard -o"

7.2 functors

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

```
struct Edge {
	int a, b, w;
};

struct cmp {
	bool operator()(const Edge &x, const Edge &y) const { return x.w <
	    y.w; }
};

int main() {
	int M = 4;
	set<Edge, cmp> v;
	for (int i = 0; i < M; ++i) {
		int a, b, w;
		cin >> a >> b >> w;
		v.insert({a, b, w});
	}
	for (Edge e : v) cout << e.a << " " << e.b << " " << e.w << "\n";
}
```

7.3 mathextra

8 Math Extra

8.1 Combinatorial formulas

$\sum_{k=0}^{n} k^2 = n(n+1)(2n+1)/6$
$\sum_{k=0}^{n} k^3 = n^2(n+1)^2/4$
$\sum_{k=0}^{n} k^4 = (6n^5 + 15n^4 + 10n^3 - n)/30$
$\sum_{k=0}^{n} k^5 = (2n^6 + 6n^5 + 5n^4 - n^2)/12$
$\sum_{k=0}^{n} x^k = (x^{n+1} - 1)/(x-1)$
$\sum_{k=0}^{n} kx^k = (x - (n+1)x^{n+1} + nx^{n+2})/(x-1)^2$
$\binom{n}{k} = \frac{n!}{(n-k)!k!}$
$\binom{n}{k} = \binom{n-1}{k} + \binom{n-1}{k-1}$
$\binom{n}{k} = \frac{n}{n-k}\binom{n-1}{k}$
$\binom{n}{k} = \frac{n-k+1}{k}\binom{n}{k-1}$
$\binom{n+1}{k} = \frac{n+1}{n-k+1}\binom{n}{k}$
$\binom{n}{k+1} = \frac{n-k}{k+1}\binom{n}{k}$
$\sum_{k=1}^{n} k\binom{n}{k} = n2^{n-1}$
$\sum_{k=1}^{n} k^2\binom{n}{k} = (n+n^2)2^{n-2}$
$\binom{m+n}{r} = \sum_{k=0}^{r}\binom{m}{k}\binom{n}{r-k}$
$\binom{n}{k} = \prod_{i=1}^{k}\frac{n-k+i}{i}$

8.2 Number theory identities

Lucas' Theorem: For non-negative integers m and n and a prime p,

$$\binom{m}{n} \equiv \prod_{i=0}^{k}\binom{m_i}{n_i} \pmod p,$$

where

$$m = m_k p^k + m_{k-1}p^{k-1} + \cdots + m_1 p + m_0$$

is the base p representation of m, and similarly for n.

8.3 Stirling Numbers of the second kind

Number of ways to partition a set of n numbers into k non-empty subsets.

$$\left\{ {n \atop k} \right\} = \frac{1}{k!}\sum_{j=0}^{k}(-1)^{(k-j)}\binom{k}{j}j^n$$

Recurrence relation:

$$\left\{ {0 \atop 0} \right\} = 1$$

$$\left\{ {n \atop 0} \right\} = \left\{ {0 \atop n} \right\} = 1$$

$$\left\{ {n+1 \atop k} \right\} = k\left\{ {n \atop k} \right\} + \left\{ {n \atop k-1} \right\}$$

8.4 Burnside's Lemma

Let G be a finite group that acts on a set X. For each g in G let X^g denote the set of elements in X that are fixed by g, which means $X^g = \{x \in X | g(x) = x\}$. Burnside's lemma assers the following formula for the number of orbits, denoted $|X/G|$:

$$|X/G| = \frac{1}{|G|}\sum_{g \in G}|X^g|$$

8.5 Numerical integration

RK4: to integrate $\dot{y} = f(t, y)$ with $y_0 = y(t_0)$, compute

$$\begin{aligned} k_1 &= f(t_n, y_n) \\ k_2 &= f(t_n + \frac{h}{2}, y_n + \frac{h}{2}k_1) \\ k_3 &= f(t_n + \frac{h}{2}, y_n + \frac{h}{2}k_2) \\ k_4 &= f(t_n + h, y_n + hk_3) \\ y_{n+1} &= y_n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4) \end{aligned}$$

8.6 template

```cpp
#include <bits/stdc++.h>

#define debug (x) cout << #x << " = " << x << endl
#define pb(x) push_back(x)
#define all(x) x.begin(),x.end()

using namespace std;
using ll = int64_t;
using ii = pair<int, int>;

ll ceil(ll a, ll b) {return a % b == 0 ? a / b : a / b + 1;}
vector<ii> dir4 = {{1, 0}, {0, 1}, {-1, 0}, {0, -1}};
vector<ii> dir8 = {{1, 0}, {1, 1}, {0, 1}, {-1, 1}, {-1, 0}, {-1, -1},
    {0, -1}, {1, -1}};

void solve() {

}

int main ()
{
    std::ios::sync_with_stdio(false);
    std::cin.tie(nullptr);

    int t = 1;
    cin>>t;
    while (t--) solve();
    return 0;
}
```

8.7 vim

let mapleader=" "
set number
set smartindent
set laststatus=2 " show status
set showcmd
syntax on " turn on colors
imap jk ¡Esc¿
set tabstop=4
set shiftwidth=4
set softtabstop=4
set expandtab
filetype plugin indent on

9 number-theory

9.1 binpow

```cpp
#include <bits/stdc++.h>

using namespace std;

const long long MOD = 998244353;

long long binpow(long long a, long long b) {
    a %= MOD;
    long long res = 1;
    while(b > 0) {
        if(b & 1)
            res = res * a % MOD;
        a = a * a % MOD;
        b >>= 1;
    }
    return res;
}

int main() {
```

```
    cout<<binpow(2,5);
    return 0;
}
```

9.2 extendedGCD

```
int gcd(int a, int b, int& x, int& y) {
    if (b == 0) {
        x = 1;
        y = 0;
        return a;
    }
    int x1, y1;
    int d = gcd(b, a % b, x1, y1);
    x = y1;
    y = x1 - y1 * (a / b);
    return d;
}
```

9.3 lcm

```
int lcm (int a, int b) {
    return a / gcd(a, b) * b;
}
```

9.4 modular-arithmetic

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

// Operation modulo something

// adition
long long add(long long a, long long b, long long m) {
    long long x = (a + b) % m;
    return x;
}
// subtraction
long long sub(long long a, long long b, long long m) {
    long long x = (a - b) % m;
    // sometimes x can be negative
    if (x < 0) x += m;
    return x;
}
// multiplication
long long multi(long long a, long long b, long long m) {
    long long x = (a * b) % m;
    return x;
}

// division
long long div(long long a, long long b, long long m) {
    // just works for prime m
    long long b_inverse = binpow(b, m-2);
    long long x = (a * b_inverse) % m;
    return x;
}
```

9.5 modularInverse

```
// Finding the Modular Inverse using Extended Euclidean algorithm

int x, y;
int g = extended_euclidean(a, m, x, y);
if (g != 1) {
    cout << "No solution!";
}
else {
    x = (x % m + m) % m;
    cout << x << endl;
}
```

9.6 primefact

```
#include <bits/stdc++.h>

using namespace std;

using ll = long long;
```

```
map<ll, ll> primeFact(ll N) {
    map<ll, ll> fact;
    for(ll i=2; i*i<=N; i++) {
        while(N % i == 0) {
            N /= i;
            fact[i]++;
        }
    }
    if(N > 1)
        fact[N]++;
    return fact;
}

int main() {

    map<ll, ll> factorials = primeFact(100);
    for(auto f : factorials) {
        cout<<f.first<<" "<<f.second<<"\n";
    }
    return 0;
}
```

9.7 sieve

```
#include <bits/stdc++.h>

using namespace std;

using ll = long long;
const ll MAX = 1e6;

// Sieve of Eratosthenes
// O(n log log(n))
vector<ll> sieve(ll MAX) {
    vector<bool> prime(MAX + 1, true);
    vector<ll> plist;
    for(ll i=2; i<=MAX; i++) {
        if(prime[i]) {
            plist.push_back(i);
            for(ll j=i*i; j<=MAX; j+=i) {
                prime[j] = false;
            }
        }
    }
    return plist;
}

int main() {

    vector<ll> plist = sieve(MAX);

    for(auto x: plist) {
        cout<<x<<" ";
    }
    cout<<"\n";
    return 0;
}
```

10 tep

10.1 addmul

```
long long add(long long a, long long b, long long m)
{
    auto r = (a + b) % m;

    return r < 0 ? r + m : r;
}

long long mul(long long a, long long b, long long m)
{
    auto r = (a * b) % m;

    return r < 0 ? r + m : r;
}

long long fast_exp_mod(long long a, long long n, long long m) {
    long long res = 1, base = a;

    while (n) {
        if (n & 1)
            res = mul(res, base, m);

        base = mul(base, base);
        n >>= 1;
```

```
    }

    return res;
}

// p is prime
long long inv(long long a, long long p) {
    return fast_exp_mod(a, p - 2, p);
}

//   assumido que (a, m) = 1
long long inverse(long long a, long long m)
{
    return fast_exp_mod(a, phi(m) - 1, m);
}

// find the inverse using extended gcd
int x, y;
int g = extended_euclidean(a, m, x, y);
if (g != 1) {
    cout << "No solution!";
}
else {
    x = (x % m + m) % m;
    cout << x << endl;
}
```

10.2 factorization

```
#include <bits/stdc++.h>

using namespace std;

map<long long, long long> factorization(long long n) {
    map<long long, long long> fs;

    for (long long d = 2, k = 0; d * d <= n; ++d, k = 0) {
        while (n % d == 0) {
            n /= d;
            ++k;
        }

        if (k) fs[d] = k;
    }

    if (n > 1) fs[n] = 1;

    return fs;
}

map<long long, long long> factorization(long long n, vector<long long>&
    primes)
{
    map<long long, long long> fs;

    for (auto p : primes)
    {
        if (p * p > n)
            break;

        long long k = 0;

        while (n % p == 0) {
            n /= p;
            ++k;
        }

        if (k)
            fs[p] = k;
    }

    if (n > 1)
        fs[n] = 1;

    return fs;
}

int main()
{
    long long n;
    cin >> n;

    auto fs = factorization(n);
    bool first = true;

    cout << n << " = ";
    for (auto [p, k] : fs)
    {
```

```
        if (not first)
            cout << " x ";

        cout << p << "^" << k;
        first = false;
    }

    cout << endl;

    return 0;
}
```

10.3 gcd

```
#include <bits/stdc++.h>

using namespace std;

long long gcd(long long a, long long b)
{
    return b ? gcd(b, a % b) : a;
}

long long ext_gcd(long long a, long long b, long long& x, long long& y)
{
    if (b == 0)
    {
        x = 1;
        y = 0;
        return a;
    }

    long long x1, y1;
    long long d = ext_gcd(b, a % b, x1, y1);

    x = y1;
    y = x1 - y1*(a/b);

    return d;
}

int main()
{
    long long a, b;
    cin >> a >> b;

    cout << "(" << a << ", " << b << ") = " << gcd(a, b) << '\n';

    long long x, y;
    auto d = ext_gcd(a, b, x, y);

    cout << d << " = (" << a << ")(" << x << ") + (" << b << ")(" << y <<
        ")\n";

    return 0;
}
```

10.4 phi

```
int phi(int n, const vector<int>& primes)
{
    if (n == 1)
        return 1;

    auto fs = factorization(n, primes);
    auto res = n;

    for (auto [p, k] : fs)
    {
        res /= p;
        res *= (p - 1);
    }

    return res;
}
```