#include "bits/stdc++.h"

using namespace std;

using ll = long long;

using ld = long double;

string el = "\n";

const ll INF = 1e18;

const ll MOD = 1e9+7;

// const ll MOD = 998244353;

void ip(bool INTERACTIVE) {

#if LOCAL

if (!INTERACTIVE)

freopen("input.txt", "r", stdin);

#endif

}

void solve(ll tc) {

}

int main() {

ios::sync\_with\_stdio(0);

ll T=1, i=0; cin.tie(0); ip(0);

// cin >> T;

#define fast i<T

// #define fast cin.peek() != EOF

while (fast) {solve(i+1);i++;} return 0;

}

vector<ll> dist, prv;

vector<vector<pair<ll, ll>>> adj; ll n, m;

void dijkstra(ll src)

{

dist.resize(n, INF);

prv.resize(n, -1);

priority\_queue<pair<ll, ll>, vector<pair<ll, ll>>, greater<>> pq;

dist[src] = 0;

pq.emplace(0, src);

while (pq.size()) {

auto [d, u] = pq.top(); pq.pop();

if (d > dist[u]) continue;

if (u == n - 1) break;

for (auto& [v, w] : adj[u])

if (dist[v] > d + w) {

dist[v] = d + w;

prv[v] = u;

pq.emplace(dist[v], v);

}

}

}

void solve() {

cin >> n >> m;

adj.assign(n, {});

for (ll u, v, w; m--; ) {

cin >> u >> v >> w;

adj[u - 1].emplace\_back(v - 1, w);

adj[v - 1].emplace\_back(u - 1, w);

}

dijkstra(0);

cout << dist[n - 1] << " " << prv[n - 1];

}

vector<bool> primes;

set<ll> facs;

void sieve(ll n) {

primes.assign(n+1, true);

primes[0] = primes[1] = false;

for (ll i = 2; i \* i <= n; ++i)

if (primes[i])

for (ll j = i \* i; j <= n; j += i)

primes[j] = false;

}

void factorize(ll n) {

for (ll i = 2; i \* i <= n; ++i)

if (primes[i] && n % i == 0) {

facs.insert(i);

while (n % i == 0) n /= i;

}

if (n > 1) facs.insert(n);

}

void solve() {

ll n; cin >> n;

sieve(n);

factorize(n);

for (auto &p : facs) cout << p << " ";

}

template<typename T>

void permu(const vector<T>& v) {

vector<T> a = v;

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

do {

for (auto& x : a) cout << x;

cout << '\n';

} while (next\_permutation(a.begin(), a.end()));

}

void solve() {

vector<char> v = {'a', 'b', 'c'};

permu(v);

}

const ll N = 1e6;

vector<ll> fact, divisors;

ll power(ll base, ll exp, ll mod) {

ll res = 1;

base %= mod;

while (exp > 0) {

if (exp & 1) res = (res \* base) % mod;

base = (base \* base) % mod;

exp >>= 1;

}

return res;

}

ll modInv(ll n, ll mod) {

return power(n, mod - 2, mod);

}

void preFact(ll max\_n) {

fact.resize(max\_n + 1);

fact[0] = 1;

for (ll i = 1; i <= max\_n; i++)

fact[i] = fact[i - 1] \* i % MOD;

}

ll nCr(ll n, ll r) {

if (r < 0 || r > n) return 0;

ll num = fact[n];

ll den = fact[r] \* fact[n - r] % MOD;

return num \* modInv(den, MOD) % MOD;

}

ll nPr(ll n, ll r) {

if (r < 0 || r > n) return 0;

ll num = fact[n];

ll den = fact[n - r];

return num \* modInv(den, MOD) % MOD;

}

void divs(ll n) {

vector<ll> larger;

for (ll i = 1; i \* i <= n; i++)

if (n % i == 0) {

divisors.emplace\_back(i);

if (i != n / i) divisors.push\_back(n / i);

}

}

void sort() {

vector<pair<ll, ll>> test = {

{3, 1}, {2, 2}, {1, 2}, {4, 0}

};

cout << "Original vector:\n";

for (auto& [f, s] : test)

cout << "{" << f << ", " << s << "}\n";

sort(test.begin(), test.end(), [](auto& a, auto& b) {

return (a.second < b.second) || (a.second == b.second && a.first < b.first);

});

cout << "\nSorted vector (by second, then by first):\n";

for (auto& [f, s] : test)

cout << "{" << f << ", " << s << "}\n";

}

ll binarysearch(const vector<ll>& arr, ll target) {

ll low = 0;

ll high = arr.size() - 1;

while (low <= high) {

ll mid = low + (high - low) / 2;

if (arr[mid] == target) return mid;

else if (arr[mid] < target) low = mid + 1;

else high = mid - 1;

}

return -1;

}

void guess() {

ll l = 1, r = 1000000;

while (l != r) {

ll mid = (l + r + 1) / 2;

printf("%d\n", mid);

fflush(stdout);

char response[3];

scanf("%s", response);

if (strcmp(response, "<") == 0) r = mid - 1;

else l = mid;

}

printf("! %d\n", l);

fflush(stdout);

}

ll gcd(ll a, ll b) {

a = abs(a);

b = abs(b);

if (b == 0) return a;

while (b != 0) {

a %= b;

swap(a, b);

}

return a;

}

ll gcd(const vector<ll>& nums) {

if (nums.empty()) return 0;

ll res = nums[0];

for (int i = 1; i < nums.size(); i++) {

res = gcd(res, nums[i]);

if (res == 1) break;

}

return res;

}

ll lcm(ll a, ll b) {

a = abs(a);

b = abs(b);

if (a == 0 || b == 0) return 0;

ll g = gcd(a, b);

a /= g;

return a \* b;

}

ll lcm(const vector<ll>& nums) {

if (nums.empty()) return 1;

ll res = nums[0];

for (int i = 1; i < nums.size(); i++) {

if (res == 0) break;

res = lcm(res, nums[i]);

}

return res;

}

ll log\_(ll base, ll target) {

if (base <= 1 || target < 1) return -1;

ll res = 0, cur = 1;

while (cur <= target / base) { cur \*= base; res++; }

return res;

}