Notebook - Maratona de Programação

Prisioneiras de WA e WAstros

Contents

1 EstruturaDados

1.1 Venice-set

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

struct VeniceSet {
    multiset<int> St;
    int water_level = 0;

    void add(int x) { St.insert(x + water_level); }

    void remove(int x)
    {
        auto it = St.find(x + water_level);
        if (it != St.end()) {
            St.erase(it);
        }
        else {
            cout << "Element " << x
                 << " not found for removal." << endl;
        }
    }

    void updateAll(int x) { water_level += x; }

    int size() { return St.size(); }
};

int main()
{
    VeniceSet vs;

    // Add elements to the VeniceSet
    vs.add(10);
    vs.add(20);
    vs.add(30);

    // Print size of the set
    cout << "Size of the set: " << vs.size() << endl;

    // Decrease all by 5
    vs.updateAll(5);

    // Remove an element
    // This removes 5 (present height) + 5 (water level) = 10
    vs.remove(5);

    // Attempt to remove an element that does not exist
    vs.remove(40);

    // Print size of the set
    cout << "Size of the set: " << vs.size() << endl;

    return 0;
}
```

1.2 Union Find

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

using namespace std;

#define ff first;
#define ss second;
#define ii pair<int, int>
#define vi vector<int>
#define ll long long
#define ld long double
#define ios ios_base::sync_with_stdio(0); cin.tie(0); cout.tie(0);

vector<int> si(100001);
vector<int> dad(100001);
int can = 1;

int find_set(int v){
  if(v == dad[v])
    return v;
  return dad[v] = find_set(dad[v]);
}
void make_set(int v){
  dad[v] = v;
  si[v] =1;
}

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

int main(){
  ios;
  int n, m;
  cin >> n >> m;
  int aux = m;
  vector<vector<int>> xs(n+1);
  for(int i=1; i<n; i++){
   dad[i] = i;
  }
  set<int> ns;
  while(m--){
    int A, B; cin >> A>> B;
    if(xs[A].size() == 2 or xs[B].size() == 2)
      can = 0;
    else if(!ns.empty() and ns.count(A) and ns.count(B) and find_set(A) ==
    find_set(B))
      can = 0;
    else{
      ns.insert(A);
      ns.insert(B);
      xs[A].push_back(B);
      xs[B].push_back(A);
      union_sets(A, B);
    }

  }
  cout << (can ? "Yes" : "No") << endl;

  }
```

1.3 Ordered Set

```
// C++ program to demonstrate the
// ordered set in GNU C++
#include <iostream>
using namespace std;

// Header files, namespaces,
// macros as defined above
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace __gnu_pbds;

#define ordered_set tree<int, null_type,less<int>, rb_tree_tag,
    tree_order_statistics_node_update>

// To implement in multiset
// template<class T> using ordered_multiset = tree<T, null_type, less_equal<T
    >, rb_tree_tag, tree_order_statistics_node_update>;

//costum cmpare
//
//template<class T>
//struct custom_compare {
//    bool operator()(const T& a, const T& b) const {
//        if (a == b) return true; // Keep duplicates
//        return a > b;
//    }
//};
//
//template<class T> using ordered_multiset = tree<T, null_type, custom_compare
    <T>, rb_tree_tag, tree_order_statistics_node_update>;

int main()
{
    // Ordered set declared with name o_set
    ordered_set o_set;

    // insert function to insert in
    // ordered set same as SET STL
    o_set.insert(5);

    // Finding the second smallest element
    // in the set using * because
    //  find_by_order returns an iterator
    cout << *(o_set.find_by_order(1))
         << endl;

    // Finding the number of elements
    // strictly less than k=4
    cout << o_set.order_of_key(4)
         << endl;

    // Finding the count of elements less
    // than or equal to 4 i.e. strictly less
    // than 5 if integers are present
    cout << o_set.order_of_key(5)
         << endl;

    // removing in a multiset
    //  auto it = ss.find_by_order(ss.order_of_key(2)); // Find iterator to
    the element 2
   // if (it != ss.end()) {
   //      ss.erase(it); // Erase the found element O(log n)
                   //
    // Deleting 2 from the set if it exists
    if (o_set.find(2) != o_set.end())
        o_set.erase(o_set.find(2));

    // Now after deleting 2 from the set
    // Finding the second smallest element in the set
    cout << *(o_set.find_by_order(1))
         << endl;

    // Finding the number of
    // elements strictly less than k=4
    cout << o_set.order_of_key(4)
         << endl;

    return 0;
}
```

2 Matematica

2.1 Permutation

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

int main()
{
    vector<int> A { 5, 3, 4, 1, 2 };

    sort(A.begin(), A.end());           // Primeira çãpermutao na ordem
    álexicogrfica

    do {
        for (size_t i = 0; i < A.size(); ++i)
            cout << A[i] << (i + 1 == A.size() ? '\n' : ' ');
    } while (next_permutation(A.begin(), A.end()));

    return 0;
}
```

```cpp
template<typename T>
long long permutations(const vector<T>& A)
{
    map<T, int> hist;

    for (auto a : A)
        ++hist[a];

    long long res = factorial(A.size());

    for (auto [a, ni] : hist)
        res /= factorial(ni);

    return res;
}
```

2.2 Primes

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

using namespace std;

const int MAX { 10000001 };

bool is_prime(int n)
{
    if (n < 2)
        return false;

    for (int i = 2; i < n; ++i)
        if (n % i == 0)
            return false;

    return true;
}

bool is_prime2(int n)
{
    if (n < 2)
        return false;

    if (n == 2)
        return true;

    if (n % 2 == 0)
        return false;

    for (int i = 3; i < n; i += 2)
        if (n % i == 0)
            return false;

    return true;
}

bool is_prime3(int n)
{
    if (n < 2)
        return false;

    if (n == 2)
        return true;

    if (n % 2 == 0)
        return false;

    for (int i = 3; i * i <= n; i += 2)
        if (n % i == 0)
            return false;

    return true;
}

vector<int> primes(int N)
{
    vector<int> ps;

    for (int i = 2; i <= N; ++i)
        if (is_prime3(i))
            ps.push_back(i);

    return ps;
}

vector<int> primes2(int N) {
    vector<int> ps;
    bitset<MAX> sieve;              // MAX deve ser maior do que N
    sieve.set();                    // Todos ãso "potencialmente" primos
    sieve[1] = false;               // 1 ãno é primo

    for (int i = 2; i <= N; ++i) {
        if (sieve[i]) {             // i é primo
            ps.push_back(i);

            for (int j = 2 * i; j <= N; j += i)
                sieve[j] = false;
        }
    }

    return ps;
}

vector<int> primes3(int N)
{
    bitset<MAX> sieve;                  // MAX deve ser maior do que N
    vector<int> ps { 2 };               // Os pares ãso tratados à parte
    sieve.set();                        // Todos ãso "potencialmente" primos

    for (int i = 3; i <= N; i += 2) {   // Apenas ímpares ãso verificados agora
        if (sieve[i]) {                 // i é primo
            ps.push_back(i);

            for (int j = 2 * i; j <= N; j += i)
                sieve[j] = false;
        }
    }
    return ps;
}

vector<long long> primes4(long long N)
{
    bitset<MAX> sieve;                    // MAX deve ser maior do que N
    vector<long long> ps { 2 };           // Os pares ãso tratados à parte
    sieve.set();                          // Todos ãso "potencialmente" primos

    for (long long i = 3; i <= N; i += 2) {   // Apenas ímpares ãso verificados
     agora
        if (sieve[i]) {                       // i é primo
            ps.push_back(i);

            for (long long j = i * i; j <= N; j += 2*i)   // úMltiplos ímpares
    >= i*i
                sieve[j] = false;
        }
    }

    return ps;
}

vector<long long> primes5(long long N)
{
    bitset<MAX> sieve;                // MAX deve ser maior do que N
    vector<long long> ps { 2, 3 };   // Pares e úmltiplos de 3 ãso tratados à
    parte
    sieve.set();                      // Todos ãso "potencialmente" primos

    // O incremento alterna entre saltos de 2 ou 4, evitando os úmltiplos de 3
    for (long long i = 5, step = 2; i <= N; i += step, step = 6 - step) {
        if (sieve[i]) {                               // i é primo
            ps.push_back(i);

            for (long long j = i * i; j <= N; j += 2*i)   // úMltiplos ímpares
    >= i*i
                sieve[j] = false;
        }
    }
    return ps;
}

int main()
{
    cout << "==== Testes de primalidade:\n\n";

    auto p = 999983;
    auto start = chrono::system_clock::now();
    auto ok = is_prime(p);
    auto end = chrono::system_clock::now();
    chrono::duration<double> t = end - start;

    cout.precision(15);
    cout << fixed;
    cout << "is_prime(" << p << ")  = " << ok << " (" << t.count() << " ms)\n"
    ;
```

```
    start = chrono::system_clock::now();
    ok = is_prime2(p);
    end = chrono::system_clock::now();
    t = end - start;

    cout << "is_prime2(" << p << ") = " << ok << " (" << t.count() << " ms)\n"
    ;

    start = chrono::system_clock::now();
    ok = is_prime3(p);
    end = chrono::system_clock::now();
    t = end - start;

    cout << "is_prime3(" << p << ") = " << ok << " (" << t.count() << " ms)\n"
    ;

    cout << "\n\n==== çãGerao de primos éat N:\n\n";
    auto N = 10000000;

    start = chrono::system_clock::now();
    auto ps = primes(N);
    end = chrono::system_clock::now();
    t = end - start;

    cout << "primes(" << N << ")  = " << ps.size() << " (" << t.count() << "
    ms)\n";

    start = chrono::system_clock::now();
    ps = primes2(N);
    end = chrono::system_clock::now();
    t = end - start;

    cout << "primes2(" << N << ") = " << ps.size() << " (" << t.count() << "
    ms)\n";

    start = chrono::system_clock::now();
    ps = primes3(N);
    end = chrono::system_clock::now();
    t = end - start;

    cout << "primes3(" << N << ") = " << ps.size() << " (" << t.count() << "
    ms)\n";

    long long M = N;
    start = chrono::system_clock::now();
    auto qs = primes4(M);
    end = chrono::system_clock::now();
    t = end - start;

    cout << "primes4(" << N << ") = " << qs.size() << " (" << t.count() << "
    ms)\n";

    start = chrono::system_clock::now();
    qs = primes5(M);
    end = chrono::system_clock::now();
    t = end - start;

    cout << "primes5(" << N << ") = " << qs.size() << " (" << t.count() << "
    ms)\n";

    return 0;
}
```

2.3 Arranjos

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

#define ll long long;

ll A(ll n, ll p){
    if(n < p)
        return 0;

    ll res = 1;

    for(ll i = n; i > p; --i){
        res*=i;
    }

    return res;
}

//long long ós aguenta 10!
//maior N! ou A^B

ll dp(int k, int a, int b){
    if(a < 0 || b < 0)
        return 0;

    if(k == 0)
        return 1;

    if(st[k][a][b] != -1)
        return st[k][a][b];

    auto res = dp(k-1, a-1, b) + dp(k-1, a, b-1);

    st[k][a][b] = res;
    return res;
}
```

2.4 Primos

```
//(N ** fi de p) % p == 1 sempre
// sistema reduzido de íresduo é os diferentes restos que deixam (7 vai ter t
    =6) - pega todos os restos
// únmeros coprimos - únmero que mdc entre eles é 1
// coprimos de 6 = 1,4,5


// TEOREMA DE FERMAT
// a^p é congruente a a(mod p) - a é inteiro e p é primo
//

// TEOREMA DE EULER
// a^fi de m é congruente a 1 mod m
// ós de primo o fi é -1
// fatora em primo e sabe que é -1
// fi de qulquer valor é = fi de primo 1 * fi de primo 2

// Fatoracao em primos
#define ll long long

ll phi(){

}

ll fatp(int x){
    map<int, int> m;

    for(int i = 2; i * i < x; i++){
       while(x%i == 0){
        x/=i;
        m[i]++;
       }



    }


}

// verificar se é primo
bool is_p(int n){
    if(n < 2)
        return false;

    if(n == 2)
        return true;

    if(n%2 == 0)
        return false;

    for(int i = 3; i * i <= n; i+=2){
        if(n%i == 0)
            return false;
    }
    return true;
}

// crivo
vector<long, long> primes(ll N){
    bitset<MAX> sieve;
    vector<long long> ps{2};
    sieve.set();

    for(ll i = 3; i<=N; i+=2){
        if(sieve[i]){
            ps.push_back(i);
            for(ll j = i * i; j<=N; j+=2*i){
                sieve[j] = false;
            }
        }
    }
    return ps;
}
```

2.5 Mod

```
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;
}

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

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

int mod(int a, int m)
{
    return ((a % m) + m) % m;
}
```

2.6 Phandfp

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

using namespace std;

using ll = long long;

ll fp(ll a, ll b){

  if ( not b)
    return 1;

  ll pr = fp(a, b/2);

  return ~b & 1 ? pr * pr : pr * pr * a;

}


ll ph(ll x){

  if ( x == 1)
    return 1;

  map<int, int> m;

  for ( int i = 2; i * i <= x; i++)
    while ( x % i == 0){
      x/=i;
      m[i]++;
  }

  if (x and x != 1)
    m[x]++;


  ll res = 1;
  for ( auto [primo, potencia ] : m)
    res = (primo - 1) fp(primo, potencia - 1);

  return res;

}


int main(){
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);

  cout << ph(400) << endl;



}
```

2.7 Fast Exp

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

using namespace std;

long long fast_exp(long long a, int n)
```

```
{
    if (n == 1)
        return a;

    auto x = fast_exp(a, n / 2);

    return x * x * (n % 2 ? a : 1);
}

long long fast_exp_it(long long a, int n)
{
    long long res = 1, base = a;

    while (n)
    {
        if (n & 1)
            res *= base;

        base *= base;
        n >>= 1;
    }

    return res;
}

int main()
{
    long long a;
    int n;

    cin >> a >> n;

    cout << a << "^" << n << " = " << fast_exp(a, n) << '\n';

    return 0;
}
```

2.8 Modular

```
#define ll long long

int mod(int a, int m){
    return ((a%m) + m)%m;
}

ll add(){

}

ll mul(){

}
```

2.9 Polinomy-add

```
polynomial operator+(const polynomial& p, const polynomial& q)
{
    int N = degree(p), M = degree(q);
    polynomial r(max(N, M) + 1, 0);

    for (int i = 0; i <= N; ++i)
        r[i] += p[i];

    for (int i = 0; i <= M; ++i)
        r[i] += q[i];

    while (not r.empty() and r.back() == 0)
        r.pop_back();

    if (r.empty())
        r.push_back(0);

    return r;
}
```

2.10 Funcoes Multiplicativas

```
#define ll long long

ll number_of_divisors(int n, const vector<int>& primes){
    auto fs = factorization(n, primes);
    ll res = 1;

    for(auto [p, k] : fs)
        res*=(k+1);

    return res;
}

ll sum_of_divisors(int n, const vector<int>& primes){
    auto fs = factorization(n, primes);
    ll res = 1;

    for(auto [p, k] : fs){
        ll pk = p;

        while(k--){
            pk *= p;
        }

        res *= (pk-1)/(p-1);
    }
    return res;
}

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;
}
```

2.11 Gcd

```
int gcd(int a, int b, const vector<int>& primes)
{
    auto ps = factorization(a, primes);
    auto qs = factorization(b, primes);

    int res = 1;

    for (auto p : ps) {
        int k = min(ps.count(p) ? ps[p] : 0, qs.count(p) ? qs[p] : 0);

        while (k--)
            res *= p;
    }

    return res;
}
```

2.12 Fatorial

```
map<int, int> factorial_factorization(int n, const vector<int>& primes)
{
    map<int, int> fs;

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

        fs[p] = E(n, p);
    }

    return fs;
}
```

2.13 Permutacoes

```
#include <bits/stdc++.h>
#include <vector>
#define ll long long

template<typename T>
ll permutations(const vector<T>& A){
    map<T, int>hist;

    for(auto a: A)
        ++hist[a];

    ll res = factorial(A.size());

    for(auto [a, ni]: hist)
        res/= factorial(ni);
    return res;
}

int main(){
    vector<int> A {5, 3, 4, 1, 2};

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

    do{
        for(size_t i = 0; i<A.size(); ++i){
            cout << A[i] << (i+1 == A.size() ? '\n' : ' ');
        }
    } while (next_permutations(A.begin(), A.end()));
    return 0;
}
```

2.14 Polinomial-degree

```
int evaluate(const polynomial& p, int x)
{
    int y = 0, N = degree(p);

    for (int i = N; i >= 0; --i)
    {
        y *= x;
        y += p[i];
    }

    return y;
}
```

2.15 Mdc

```
#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;
}
```

2.16 Fatorization

```
#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;
}
```