

**IUH** **Industrial University of Ho Chi Minh City**

**Team Notebook: IUH.OCEAN**

Member

*Dao Xuan Hoang Tuan*

*Lam Quang Phu*

*Nguyen Thi Thanh Hoa*

The 2022 ICPC Asia Ho Chi Minh City Regional Contest

December 09, 2022

**Mục lục**

1. Contest
2. Mathematics
3. Data structures
4. Numerical
5. Number theory
6. Combinatorial
7. Graph
8. Geometry
9. Strings
10. Various
11. **CONTEST**

**Kiểm tra phiên bản trình biên dịch GCC**

#include<iostream>

int main()

{

std::cout << \_\_cplusplus << '\n';

if (\_\_cplusplus == 201703L) std::cout << "C++17\n";

else if (\_\_cplusplus == 201402L) std::cout << "C++14\n";

else if (\_\_cplusplus == 201103L) std::cout << "C++11\n";

else if (\_\_cplusplus == 199711L) std::cout << "C++98\n";

else std::cout << "pre-standard C++\n";

}

**Template**

#include <bits/stdc++.h>

using namespace std;

#define Hello\_IUH\_Ocean ios\_base::sync\_with\_stdio(false);cin.tie(0);cout.tie(0);

#define print\_fix cout << fixed << setprecision(20);

#define show(x) cerr << #x << " -> " << x << endl;

#define show3 cerr << "\*\*\*" << endl;

#define show2 cerr << "\*\*" << endl;

#define show1 cerr << "\*" << endl;

#define all(v) v.begin(), v.end()

#define sz(t) (int) t.size()

#define pb push\_back

#define se second

#define fi first

#define el endl

#define ed '\n'

#define \_ " "

void debug\_out() {cout << '\n';}

template <typename Head, typename ...Tail>

void debug\_out(Head H, Tail ...T)

{

cout << H << ' ';

debug\_out(T...);

}

#define fix(...) cout << "[" << #\_\_VA\_ARGS\_\_ << "]: ", debug\_out(\_\_VA\_ARGS\_\_)

const long long N =1e6 + 7;

const long long Nn = 1e3+10;

const double PI = atan(1)\*4;

const int MOD = 1e9 + 7;

const long long INF = 1e9 + 7ll;

/\* --- you should drink a cup of milk tea before reading my code ---- \*/

int main()

{

Hello\_IUH\_Ocean

}

/\* Test case

\*/

/\* My code is very beautiful and artistic \*/

**Kinh nghiệm thi đấu**

Chiến thuật & lưu ý:

Try hard đến giây phút cuối cùng của contest

Một tiếng cuối chúng ta cần thời gian hơn là cần giảm thiểu penaty

Kiên trì viết test, nhận xét để ra được quy luật

Suy nghĩ theo hướng những tài nguyên mình có

Động viên các thành viên trong team giữ ý chí

Một bài nên có ít nhất hai người cùng giải và 1 fix bug

Nhớ mang từ điển

Trong những lúc khó khăn:

Cả team cùng giải bài

Suy nghĩ solution khác: Xử lý offline, 2 bên gần nhất, Truy hồi, chặt nhị phân, CTDL, đồ thị

Viết test để ra quy luật

Đề nói một đằng suy nghĩ một nẻo

Cố lên chỉ còn thiếu một trường hợp

Ăn bánh

Hãy biến thành tourist và chiến đâu

Bỏ làm câu khác

Trước khi submit:

Đối với những bài viết được trình sinh test, code trâu kiểm tra đáp án thì nên viết

Viết một số trường hợp thử nghiệm đơn giản nếu mẫu không đủ.

Tạo các trường hợp tối đa để kiểm tra thời gian và bộ nhớ

Kiểm tra tràn, kiểu dữ liệu

Wrong answer:

In code

Bạn có xóa tất cả cấu trúc dữ liệu giữa các trường hợp thử nghiệm không?

Thuật toán của bạn có thể xử lý toàn bộ phạm vi đầu vào không?

Đọc lại code một lần nữa.

Bạn có xử lý chính xác tất cả các trường hợp đặc biệt không? Bạn đã hiểu đúng vấn đề chưa?

Bất kỳ biến chưa được khởi tạo nào? Bất kỳ tràn? Nhầm lẫn giữa N và M, i và j, l và r, v.v.?

Bạn có chắc thuật toán của mình hoạt động không?

Bạn có chắc các chức năng STL bạn sử dụng hoạt động như bạn nghĩ không?

Thêm một số xác nhận, có thể gửi lại.

Tạo một số trường hợp thử nghiệm để chạy thuật toán của bạn.

Giải thích thuật toán của bạn cho đồng đội.

Yêu cầu đồng đội xem code của bạn.

Đi dạo một chút, e.g.

Vào nhà vệ sinh.

Định dạng đầu ra của bạn có đúng không? (bao gồm cả khoảng trắng)

Viết lại giải pháp của bạn từ đầu hoặc để một thành viên trong nhóm thực hiện.

Có thể sài trick assert để kiểm tra xem bạn sai chỗ nào

Runtime error:

Xem có tràn mảng không, có truy cập vùng nhớ ngoài không

Kiểm tra các test cơ sở chưa

Bất kỳ biến chưa được khởi tạo nào?

Bất kỳ khẳng định nào có thể thất bại?

Bất kỳ phép chia nào có thể cho 0? (ví dụ mod 0)

Bất kỳ đệ quy vô hạn nào có thể xảy ra? Con trỏ hoặc trình vòng lặp không hợp lệ?

Bạn đang sử dụng quá nhiều bộ nhớ? Gỡ lỗi bằng cách gửi lại (ví dụ: tín hiệu được ánh xạ lại, xem Khác nhau).

Time litmit exceeded:

Bạn có bất kỳ vòng lặp vô hạn nào có thể không?

Độ phức tạp của thuật toán của bạn là gì?

Bạn đang sao chép rất nhiều dữ liệu không cần thiết?

Đầu vào và đầu ra lớn cỡ nào? Tránh vectơ, map. (sử dụng arrays/unordered\_map)

Đồng đội của bạn nghĩ gì về thuật toán của bạn?

Memory limit exceeded:

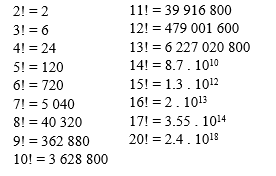
Dung lượng bộ nhớ tối đa mà thuật toán của bạn cần là bao nhiêu?

Bạn có xóa tất cả cấu trúc dữ liệu giữa các trường hợp thử nghiệm không?

1. **MATHEMATICS**

**Các phép toán:**

Ảnh có chứa văn bản, biên lai

Mô tả được tạo tự động

**Phương trình:**

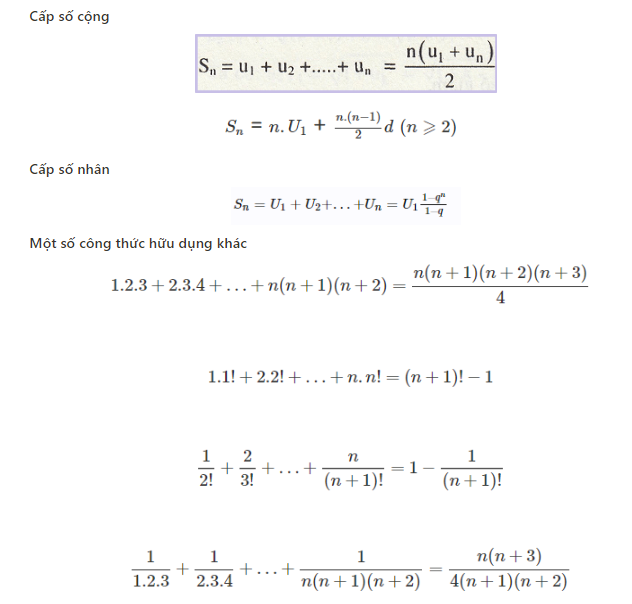
Delta > 0 (2 nghiệm): 

Delta = 0 (1 nghiệm): 

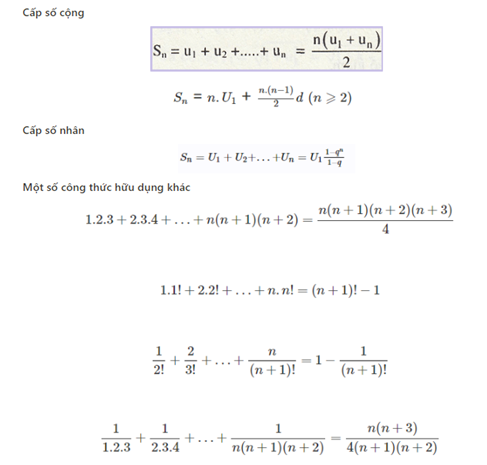
**Sums**

**Ảnh có chứa văn bản

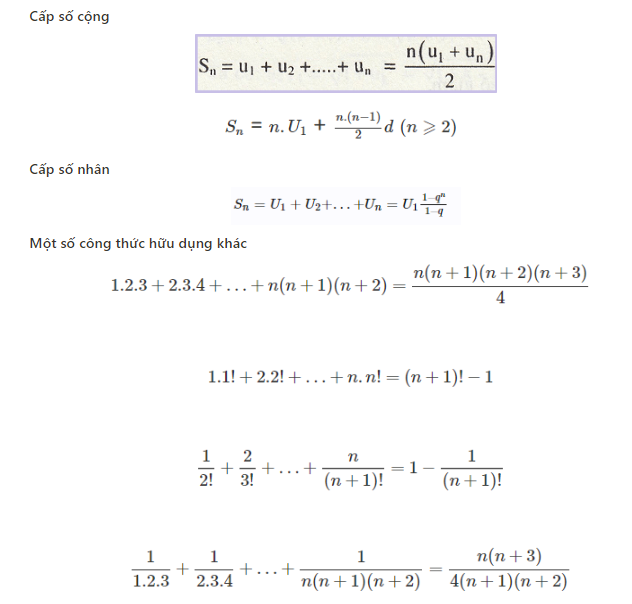
Mô tả được tạo tự động**



**Cấp số cộng**

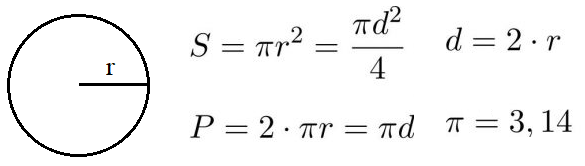


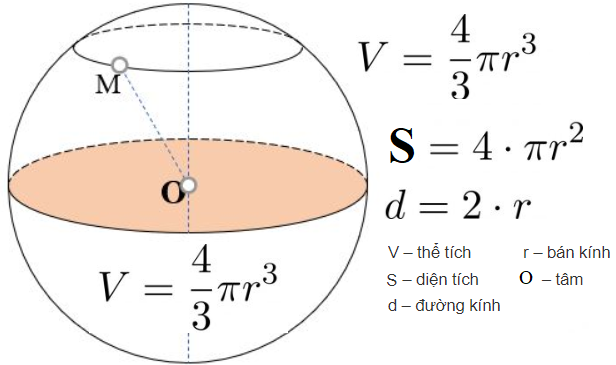
**Cấp số nhân**



**Công thức hình học**

Số Pi được biểu diễn với 50 chữ số thập phân: 3,14159 26535 89793 23846 26433 83279 50288 41971 69399 37510.





Bán chu vi: p = (a + b + c) / 2  
Diện tích: 

Nội tiếp: Đường tròn:  bán kính: 

Độ dài đường trung tuyến (chia tam giác thành hai tam giác có diện tích bằng nhau): 

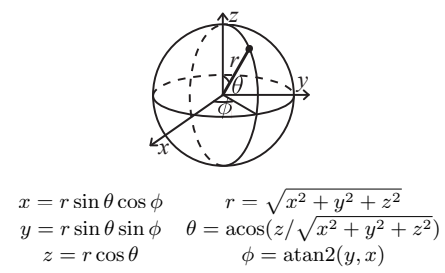
Độ dài đường phân giác (chia hai góc): 

Định luật Sin: 

**Tứ giác**: Với độ dài các cạnh a, b, c, d, đường chéo e, f, góc chéo θ, diện tích A và từ thông F = 

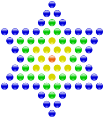
Đối với tứ giác nội tiếp có tổng các góc đối bằng 180◦



Tọa độ cầu: 

**Dãy số**

Fibonaci: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597, 2584, 4181, 6765, 10946, 17711, 28657, 46368,... F(n) = F(n-1) + F(n-2)

Star number:  1, 13, 37, 73, 121, 181, 253, 337, 433, 541, (là một CSC) 

**Bộ ba số Pythagoras**

**Ảnh có chứa văn bản

Mô tả được tạo tự động**

Trong đó m và n là hai số nguyên tố cùng nhau, có một số chẵn và một số lẻ, với m > n và k là số nguyên dương tùy ý

**Kiểm tra tam giác vuông (1018)**

**Ảnh có chứa văn bản

Mô tả được tạo tự động**

(c − a)(c − b)/2 là số chính phương thì là tam giác vuông

**Binmul**

long long MOD = 1e9 + 7;

long long binmul(long long a, long long b)

{

long long res = 0;

while(b > 0)

{

if(b & 1) res = (res % MOD + a % MOD) % MOD;

a = (a % MOD + a % MOD) % MOD;

b /= 2;

}

return res % MOD;

}

**Binpow**

long long binpow(long long a, long long b)

{

long long res = 1;

while (b > 0)

{

if (b & 1)

res = res \* a;

a = a \* a;

b >>= 1;

}

return res;

}

**Lũy thừa tránh tràn số**

int poww(int a, int b)

{

int ans = 1;

for(int i=1; i<=b; i++)

{

if(INF / a < ans) return INF;

ans \*= a;

}

return ans;

}

**GCD & LCM**

Lcm(a, b) = a\*b / gcd(a, b)

int gcd (int a, int b)

{

return b ? gcd (b, a % b) : a;

}

int gcd (int a, int b)

{

while (b) {

a %= b;

swap(a, b);

}

return a;

}

int lcm (int a, int b)

{

return a / gcd(a, b) \* b;

}

\_\_gcd(a, b);

lcm(a, b);

**Số nguyên tố & sàng nguyên tố**

Số các số nguyên tố từ 1 đến n xấp xỉ là n / ln(n)

Các số nguyên tố dưới 109 thì 2 số liên tiếp chỉ cách nhau ≤ 320 đơn vị

**Check Số nguyên tố O(n)**

bool isPrime[N+5];

void Sieve()

{

for(int i = 0; i <= N;++i)

{

isPrime[i] = true;

}

isPrime[0] = false;

isPrime[1] = false;

for(int i = 2; i \* i <= N; ++i)

{

if(isPrime[i] == true)

{

for(int j = i \* i; j <= N; j += i)

isPrime[j] = false;

}

}

}

**Phân tích một số ra ước nhỏ nhất của nó**

long long isprime[N+5];

void Sieve\_gcd(){

isprime[0] = 0;

isprime[1] = 1;

for(int i = 2; i\*i<N; i++)

{

if(isprime[i] == 0)

{

for(int j = i\*i; j <=N; j+=i)

{

if(isprime[j] == 0)

{

isprime[j] = i;

}

}

}

}

for (int i = 2; i < N; ++i)

{

if (isprime[i] == 0)

{

isprime[i] = i;

}

}

}

**Sàng để phân tích thừa số nguyên tố**

long long primeDiv[N+5];

void Sieve\_Fac()

{

for(int i=2; i \* i <= N; i++)

{

if(primeDiv[i] == 0)

{

for(int j=i\*i; j<=N; j+=i)

{

primeDiv[j] = i;

}

}

}

for(int i=2; i<=N; i++)

{

if(primeDiv[i] == 0) primeDiv[i] = i;

}

}

**Phân tích một số ra thành các thừa số nguyên tố**

//map<int, int> Fac;

//vector<int> Fac(N);

while(x > 1)

{

Fac[primeDiv[x]]++;

x /= primeDiv[x];

}

**Kiểm tra số nguyên tố nâng cao**

if (n <= 1)

return false;

if (n == 2 || n == 3)

return true;

if (n % 2 == 0 || n % 3 == 0)

return false;

for (int i = 5; 1ll\*i\*i <= n; i = i + 6)

if (n % i == 0 || n % (i + 2) == 0)

return false;

return true;

**Kiểm tra số nguyên tố cực lớn**

vector<long long> snt; // Chua cac so nguyen to sau khi chay sang nguyen to truoc do

bool Ok\_prime(long long x)

{

for(auto it : snt)

{

if(1ll \* it \* it > x) return true;

if(x % it == 0) return false;

}

assert(false);

}

Kiểm tra số nguyên tố cực lớn trong O(log(n)) sử dụng Miller rabin

#include <bits/stdc++.h>

using namespace std;

uint64\_t pow(uint64\_t a, uint64\_t n, uint64\_t m) {

uint64\_t result = 1;

a = a % m;

while (n > 0) {

if (n & 1) result = result \* a % m;

n >>= 1;

a = a \* a % m;

}

return result;

}

pair<uint64\_t, uint64\_t> factor(uint64\_t n) {

uint64\_t s = 0;

while ((n & 1) == 0) {

s++;

n >>= 1;

}

return {s, n};

}

bool witness\_test(uint64\_t s, uint64\_t d, uint64\_t n, uint64\_t witness) {

if (n == witness) return true;

uint64\_t p = pow(witness, d, n);

if (p == 1) return true;

for (; s > 0; s--) {

if (p == n-1) return true;

p = p \* p % n;

}

return false;

}

bool miller(uint64\_t n) {

if (n < 2) return false;

if ((n & 1) == 0) return n == 2;

uint64\_t s, d;

tie(s, d) = factor(n-1);

return witness\_test(s, d, n, 2) && witness\_test(s, d, n, 7) && witness\_test(s, d, n, 1662803);

}

int main()

{

Hello\_i\_am\_Salmon

long long n; cin >> n;

if(miller(n)) cout << "La so nguyen to" << '\n';

else cout << "Khong phai so nguyen to" << '\n';

}

/\* Test case

\*/

/\* My code is very beautiful and artistic \*/

**Sàng nguyên tố trên đoạn**

long long L, R; cin >> L >> R;

long long cnt = 0;

vector<bool> isPrime(R - L + 1, true); // x là số nguyên tố khi và chỉ khi isPrime[x - l] == true

for (long long i = 2; i \* i <= R; ++i) {

for (long long j = max(i \* i, (L + i - 1) / i \* i); j <= R; j += i) {

isPrime[j - L] = false;

}

}

if (1 >= L) { // Xét riêng trường hợp số 1

isPrime[1 - L] = false;

}

**Đếm số lượng mũ nguyên tố của một số**

long long Count\_Fac(long long x)

{

long long ans = 0;

for(auto it : snt)

{

if(it \* it > x) break;

while(x % it == 0)

{

x /= it;

ans ++;

}

}

if(x > 1) ans++;

return ans;

}

**Tính tổng các số nguyên tố từ 1 → N (N = 1010)**

long long Sum\_prime(long long N) {

long long r = (long long) sqrt(N);

vector <long long> a(r + 1);

vector <long long> b(r + 1);

for (long long i = 1; i <= r; i++) {

a[i] = i \* (i + 1) / 2 - 1;

b[i] = (N/i) \* (N/i + 1) / 2 - 1;

}

for (long long p = 2; p <= r; p++)

if (a[p] > a[p - 1]) {

long long sp = a[p - 1];

long long p2 = p \* p;

long long to = min(r, N/p2);

for (long long i = 1; i <= to; i++) {

long long vp = i \* p;

if (vp <= r) vp = b[vp];

else vp = a[N / vp];

b[i] -= p \* (vp - sp);

}

for (long long v = r; v >= p2; v--)

a[v] -= p \* (a[v/p] - sp);

}

return b[1];

}

int main() {

cout << "Result 1: " << Sum\_prime(100000000000) << endl;

}

**Modulo**

#include <bits/stdc++.h>

using namespace std;

using ll = long long;

using ull = unsigned long long;

constexpr unsigned mod = 1000000007;

struct Modint{

unsigned num = 0;

constexpr Modint() noexcept {}

constexpr Modint(const Modint &x) noexcept : num(x.num){}

inline constexpr operator ll() const noexcept { return num; }

inline constexpr Modint& operator+=(Modint x) noexcept { num += x.num; if(num >= mod) num -= mod; return \*this; }

inline constexpr Modint& operator++() noexcept { if(num == mod - 1) num = 0; else num++; return \*this; }

inline constexpr Modint operator++(int) noexcept { Modint ans(\*this); operator++(); return ans; }

inline constexpr Modint operator-() const noexcept { return Modint(0) -= \*this; }

inline constexpr Modint operator-(Modint x) const noexcept { return Modint(\*this) -= x; }

inline constexpr Modint& operator-=(Modint x) noexcept { if(num < x.num) num += mod; num -= x.num; return \*this; }

inline constexpr Modint& operator--() noexcept { if(num == 0) num = mod - 1; else num--; return \*this; }

inline constexpr Modint operator--(int) noexcept { Modint ans(\*this); operator--(); return ans; }

inline constexpr Modint& operator\*=(Modint x) noexcept { num = ull(num) \* x.num % mod; return \*this; }

inline constexpr Modint& operator/=(Modint x) noexcept { return operator\*=(x.inv()); }

template<class T> constexpr Modint(T x) noexcept {

using U = typename conditional<sizeof(T) >= 4, T, int>::type;

U y = x; y %= U(mod); if(y < 0) y += mod; num = unsigned(y);

}

template<class T> inline constexpr Modint operator+(T x) const noexcept { return Modint(\*this) += x; }

template<class T> inline constexpr Modint& operator+=(T x) noexcept { return operator+=(Modint(x)); }

template<class T> inline constexpr Modint operator-(T x) const noexcept { return Modint(\*this) -= x; }

template<class T> inline constexpr Modint& operator-=(T x) noexcept { return operator-=(Modint(x)); }

template<class T> inline constexpr Modint operator\*(T x) const noexcept { return Modint(\*this) \*= x; }

template<class T> inline constexpr Modint& operator\*=(T x) noexcept { return operator\*=(Modint(x)); }

template<class T> inline constexpr Modint operator/(T x) const noexcept { return Modint(\*this) /= x; }

template<class T> inline constexpr Modint& operator/=(T x) noexcept { return operator/=(Modint(x)); }

inline constexpr Modint inv() const noexcept { ll x = 0, y = 0; extgcd(num, mod, x, y); return x; }

static inline constexpr ll extgcd(ll a, ll b, ll &x, ll &y) noexcept { ll g = a; x = 1; y = 0; if(b){ g = extgcd(b, a % b, y, x); y -= a / b \* x; } return g; }

inline constexpr Modint pow(ull x) const noexcept { Modint ans = 1, cnt = \*this; while(x){ if(x & 1) ans \*= cnt; cnt \*= cnt; x /= 2; } return ans; }

};

std::istream& operator>>(std::istream& is, Modint& x) noexcept { ll a; cin >> a; x = a; return is; }

inline constexpr Modint operator""\_M(ull x) noexcept { return Modint(x); }

std::vector<Modint> fac(1, 1), inv(1, 1);

inline void reserve(ll a){

if(fac.size() >= a) return;

if(a < fac.size() \* 2) a = fac.size() \* 2;

if(a >= mod) a = mod;

while(fac.size() < a) fac.push\_back(fac.back() \* Modint(fac.size()));

inv.resize(fac.size());

inv.back() = fac.back().inv();

for(ll i = inv.size() - 1; !inv[i - 1]; i--) inv[i - 1] = inv[i] \* i;

}

inline Modint fact(ll n){ if(n < 0) return 0; reserve(n + 1); return fac[n]; }

inline Modint perm(ll n, ll r){

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

if(n >> 24){ Modint ans = 1; for(ll i = 0; i < r; i++) ans \*= n--; return ans; }

reserve(n + 1); return fac[n] \* inv[n - r];

}

inline Modint comb(ll n, ll r){ if(r < 0 || n < r) return 0; reserve(r + 1); return perm(n, r) \* inv[r]; }

inline Modint Mcomb(ll n, ll r){ return comb(n + r - 1, n - 1); }

inline Modint catalan(ll n){ reserve(n \* 2 + 1); return fac[n \* 2] \* inv[n] \* inv[n + 1]; }

//　exawizards2019\_e

int main(){

Modint ans = 1;

ans /= 2;

Modint cnt = ans;

ll b, w;

cin >> b >> w;

for(int i = 0; i < b + w; i++){

cout << ans << '\n';

cnt /= 2;

if(i >= w - 1) ans += comb(i, w - 1) \* cnt;

if(i >= b - 1) ans -= comb(i, b - 1) \* cnt;

}

}

**Ước của một số**

**Đếm ước của một số O(sqrt(n))**

long long countDiv(long long n)

{

long long cnt = 0;

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

{

if(n % i == 0)

{

if(n / i == i) cnt++;

else cnt += 2;

}

}

return cnt;

}

**Lấy các ước của một số O(sqrt(n))**

vector<int> Div;

void countDiv(long long n)

{

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

{

if(n % i == 0)

{

if(n / i == i) Div.pb(i);

else

{

Div.pb(i);

Div.pb(n/i);

}

}

}

}

**Đếm ước nâng cao cho nhiều số và mỗi lần truy vấn O(1)**

long long cntDiv[N+5];

for(int i = 1; i\*i <= N; i++)

for(int j = i\*i; j <= N; j+=i)

cntDiv[j]++;

for(int i = 1; i <= N; i++) cntDiv[i]\*=2;

for(int i = 1; i\*i <= N; i++) cntDiv[i\*i]--;

**Xử lý sai số của Sqrt**

long long getSqrt(unsigned long long x)

{

long long temp = sqrt(x);

for(int i=-3; i<=3; i++)

{

if(temp + i >= 0 && (temp + i)\*(temp + i) == x)

{

return temp + i;

}

}

return -1;

}

**Nhân ma trận**

**Code 1**

#define R1 2 // number of rows in Matrix-1

#define C1 2 // number of columns in Matrix-1

#define R2 2 // number of rows in Matrix-2

#define C2 2 // number of columns in Matrix-2

void mulMat(int mat1[][C1], int mat2[][C2])

{

int rslt[R1][C2];

cout << "Multiplication of given two matrices is:\n";

for (int i = 0; i < R1; i++)

{

for (int j = 0; j < C2; j++)

{

rslt[i][j] = 0;

for (int k = 0; k < R2; k++)

{

rslt[i][j] += mat1[i][k] \* mat2[k][j];

}

cout << rslt[i][j] << "\t";

}

cout << endl;

}

}

int main()

{

Hello\_i\_am\_Salmon

// R1 = 4, C1 = 4 and R2 = 4, C2 = 4 (Update these

// values in MACROs)

int mat1[R1][C1] = { { 1, 1 },

{ 2, 2 } };

int mat2[R2][C2] = { { 1, 1 },

{ 2, 2 } };

mulMat(mat1, mat2);

return 0;

}

**Code 2**

#include <bits/stdc++.h>

using namespace std;

// using namespace \_\_gnu\_pbds;

const long long MOD = 998244353;

/\* --- you should drink a cup of milk tea before reading my code ---- \*/

struct MT

{

long long a[10][10];

};

long long cs[10][10] = {

{1, 4, 2, 0, 12, 12},

{1, 0, 0, 0, 0, 0},

{0, 1, 0, 0, 0, 0},

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

{0, 0, 0, 1, 0, 0},

{0, 0, 0, 0, 1, 0},

};

long long bd[10][10] = {

{12, 0, 0, 0, 0, 0},

{0, 0, 0, 0, 0, 0},

{0, 0, 0, 0, 0, 0},

{5, 0, 0, 0, 0, 0},

{1, 0, 0, 0, 0, 0},

{1, 0, 0, 0, 0, 0},

};

MT base;

MT dv;

void Setup()

{

for(int i=0; i<6; i++)

{

for(int j=0; j<6; j++)

{

base.a[i][j] = cs[i][j];

}

cout << endgame.a[0][0] << ed;

}

**Code 3**

struct MaTran

{

ll c[21][21] = {{}, {}};

};

MaTran operator \* (MaTran a, MaTran b)

{

MaTran res;

for (int i=0; i<n; i++)

for (int j=0; j<n; j++)

{

res.c[i][j] = 0;

for (int k=0; k<n; k++)

res.c[i][j] = (res.c[i][j]%mod+((a.c[i][k]%mod)\*(b.c[k][j]%mod)%mod))%mod;

}

return res;

}

MaTran operator + (MaTran a, MaTran b)

{

MaTran res;

for (int i=0; i< n; i++)

for (int j=0; j< n; j++)

{

res.c[i][j] = (a.c[i][j]%mod + b.c[i][j]%mod)%mod;

}

return res;

}

**Đổi cơ số giữa hệ nhị phân và thập phân**

**Thập phân sang nhị phân**

vector<bool> Binary;

int DectoBi(long long n)

{

int i = 0;

while(n > 0)

{

Binary.pb(n%2);

n /= 2;

i++;

}

reverse(all(Binary));

return i;

}

**Nhị phân sang thập phân**

long long BitoDec(string n)

{

long long dec = 0;

long long base = 1;

for(int i=sz(n)-1; i>=0; i--)

{

if(n[i] == '1')

{

dec += base;

}

base \*= 2;

}

return dec;

}

1. **DATA STRUCTURES**

**Vector**

// Khai bao

#include <vector>

vector<int> a;

vector<pair<int, int>> b;

vector<int> c = {1, 2, 3, 5};

vector<int> d(10);

vector<int> e(10, 0);

//Thao tac co ban

a.push\_back(1);

a.pop\_back(1);

a.insert(a.begin()+1, 2);

a.insert(a.begin()+1, 10, 3); // chen nhieu

a.erase(a.begin()+2);

a.back();

a.front();

a.clear();

a.empty();

a.size();

a.reverse();

a.resize();

**queue**

//Khai bao

#include <queue>

queue<int> Q;

queue<int> Q(10, 1);

//Thao tac

Q.push(1);

Q.size();

Q.empty();

Q.front();

Q.back();

Q.pop();

**Deque**

// Khai bao

#include <vector>

deque<int> a;

deque<pair<int, int>> b;

deque<int> c = {1, 2, 3, 5};

deque<int> d(10);

deque<int> e(10, 0);

//Thao tac co ban

a.push\_back(1);

a.push\_front();

a.pop\_back(1);

a.pop\_front();

a.insert(a.begin()+1, 2);

a.insert(a.begin()+1, 10, 3); // chen nhieu

a.erase(a.begin()+2);

a.back();

a.front();

a.clear();

a.empty();

a.size();

a.reverse();

a.resize();

**priority\_queue**

#include <queue>

priority\_queue<int> Q;

priority\_queue<int> Q(10, 1);

priority\_queue<int, vector<int>, greater<int>> Q;

//Thao tac

Q.push(1);

Q.size();

Q.empty();

Q.top()

Q.pop();

**Set**

//Khai bao

#include<set>

set<int> st;

//Thao tac

st.insert(1);

st.begin();

st.end();

st.clear();

st.count(1);

st.empty();

st.erase();

st.find();

st.lower\_bound();

st.size();

st.upper\_bound();

**map**

//Khai bao

#include <map>

map<int, int> mp;

//Thao tac

mp.begin();

mp.end();

mp.count();

mp.clear();

mp.empty();

mp.erase();

mp.find();

mp.insert();

mp.lower\_bound();

mp.size();

mp.upper\_bound();

**multiset**

// Khai bao

#include<set>

multiset<int> mst;

//Thao tac

mst.clear();

mst.count();

mst.begin();

mst.end();

mst.empty();

mst.erase();

mst.find();

mst.insert();

mst.lower\_bound();

mst.size();

mst.upper\_bound();

**unordered\_map**

//Khai bao

#include <unordered\_map>

unordered\_map<int, int> mp;

//Thao tac

mp.begin();

mp.end();

mp.count();

mp.clear();

mp.empty();

mp.erase();

mp.find();

mp.insert();

mp.size();

mp.reserve();

**Order set**

#include <iostream>

using namespace std;

#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>

o\_set.insert(5);

o\_set.insert(1);

o\_set.insert(2);

cout << \*(o\_set.find\_by\_order(1))

<< endl;

cout << o\_set.order\_of\_key(4)

<< endl;

cout << o\_set.order\_of\_key(5)

<< endl;

if (o\_set.find(2) != o\_set.end())

o\_set.erase(o\_set.find(2));

cout << \*(o\_set.find\_by\_order(1))

<< endl;

cout << o\_set.order\_of\_key(4)

<< endl;

return 0;

}

(int chính là kiểu dữ liệu)

Hoặc có thể sử dụng template này

template<class T>using index\_set=tree<T, null\_type, less<T>, rb\_tree\_tag, tree\_order\_statistics\_node\_update> ;

index\_set<pair<int, int>> a;

**Segment tree**

#include <bits/stdc++.h>

using namespace std;

const int inf = 1e9 + 7;

const int maxN = 1e5 + 7;

int n, q;

int a[maxN];

int st[4 \* maxN]; // Lí do sử dụng kích thước mảng là 4 \* maxN sẽ được giải thích ở phần sau

// Thủ tục xây dựng cây phân đoạn

void build(int id, int l, int r) {

// Đoạn chỉ gồm 1 phần tử, không có nút con

if (l == r) {

st[id] = a[l];

return;

}

// Gọi đệ quy để xử lý các nút con của nút id

int mid = l + r >> 1; // (l + r) / 2

build(2 \* id, l, mid);

build(2 \* id + 1, mid + 1, r);

// Cập nhật lại giá trị min của đoạn [l, r] theo 2 nút con

st[id] = min(st[2 \* id], st[2 \* id + 1]);

}

// Thủ tục cập nhật

void update(int id, int l, int r, int i, int val) {

// i nằm ngoài đoạn [l, r], ta bỏ qua nút id

if (l > i || r < i) return;

// Đoạn chỉ gồm 1 phần tử, không có nút con

if (l == r) {

st[id] = val;

return;

}

// Gọi đệ quy để xử lý các nút con của nút id

int mid = l + r >> 1; // (l + r) / 2

update(2 \* id, l, mid, i, val);

update(2 \* id + 1, mid + 1, r, i, val);

// Cập nhật lại giá trị min của đoạn [l, r] theo 2 nút con

st[id] = min(st[2 \* id], st[2 \* id + 1]);

}

// Hàm lấy giá trị

int get(int id, int l, int r, int u, int v) {

// Đoạn [u, v] không giao với đoạn [l, r], ta bỏ qua đoạn này

if (l > v || r < u) return inf;

/\* Đoạn [l, r] nằm hoàn toàn trong đoạn [u, v] mà ta đang truy vấn,

ta trả lại thông tin lưu ở nút id \*/

if (l >= u && r <= v) return st[id];

// Gọi đệ quy với các nút con của nút id

int mid = l + r >> 1; // (l + r) / 2

int get1 = get(2 \* id, l, mid, u, v);

int get2 = get(2 \* id + 1, mid + 1, r, u, v);

// Trả ra giá trị nhỏ nhất theo 2 nút con

return min(get1, get2);

}

int main() {

cin >> n;

for (int i = 1; i <= n; ++i) cin >> a[i];

build(1, 1, n);

cin >> q;

while (q--) {

int type, x, y;

cin >> type >> x >> y;

if (type == 1) update(1, 1, n, x, y); // Gán giá trị y cho phần tử ở vị trí x

else cout << get(1, 1, n, x, y) << '\n'; // In ra giá trị nhỏ nhất trong đoạn [x, y]

}

}

**Segment tree với multiset**

#include <bits/stdc++.h>

using namespace std;

const int inf = 1e9 + 7;

const int maxN = 1e5 + 7;

int n, m;

int a[maxN];

multiset <int> st[4 \* maxN];

void build(int id, int l, int r) {

if (l == r) {

st[id].insert(a[l]);

return;

}

int mid = l + r >> 1;

build(2 \* id, l, mid);

build(2 \* id + 1, mid + 1, r);

st[id] = st[2 \* id + 1];

for (auto x : st[2 \* id]) st[id].insert(x);

}

void update(int id, int l, int r, int i, int old, int val) {

if (l > i || r < i) return;

if (l == r) {

st[id].clear();

st[id].insert(val);

return;

}

int mid = l + r >> 1;

update(2 \* id, l, mid, i, old, val);

update(2 \* id + 1, mid + 1, r, i, old, val);

st[id].erase(st[id].find(old));

st[id].insert(val);

}

int get(int id, int l, int r, int u, int v, int k) {

if (l > v || r < u) return inf;

if (l >= u && r <= v) {

auto it = st[id].lower\_bound(k);

if (it == st[id].end()) return inf;

return \*it;

}

int mid = l + r >> 1;

int get1 = get(2 \* id, l, mid, u, v, k);

int get2 = get(2 \* id + 1, mid + 1, r, u, v, k);

return min(get1, get2);

}

int main() {

cin >> n >> m;

for (int i = 1; i <= n; ++i) cin >> a[i];

build(1, 1, n);

while (m--){

int type, l, r, k;

cin >> type;

if (type == 1) {

cin >> l >> k;

update(1, 1, n, l, a[l], k);

a[l] = k;

}

else {

cin >> l >> r >> k;

int ans = get(1, 1, n, l, r, k);

cout << ((ans == inf) ? -1 : ans) << '\n';

}

}

}

**Segment tree offline**

long long n, q;

long long a[N];

long long st[4 \* N];

long long endgame[N];

vector<pair<pair<int, int>, int>> Query;

void Build(int id, int l, int r)

{

if(l == r)

{

st[id] = 1;

return;

}

int mid = (l + r) / 2;

Build(2 \* id, l, mid);

Build(2 \* id + 1, mid+1, r);

st[id] = st[2 \* id] + st[2 \* id + 1];

}

void Update(int id, int l, int r, int x, int val)

{

if(l > x || r < x) return;

if(l == r)

{

st[id] = val;

return;

}

int mid = (l + r) / 2;

Update(2 \* id, l, mid, x, val);

Update(2 \* id + 1, mid + 1, r, x, val);

st[id] = st[2 \* id + 1] + st[2 \* id];

}

int Get(int id, int l, int r, int u, int v)

{

if(u > r || v < l) return 0;

if(u <= l && v >= r)

{

return st[id];

}

int mid = (l + r) / 2;

int get1 = Get(id \* 2, l, mid, u, v);

int get2 = Get(id \* 2 + 1, mid+1,r, u, v);

return get1+get2;

}

int main()

{

Hello\_i\_am\_Salmon

cin >> n;

for(int i=1; i<=n; i++) cin >> a[i];

Build(1, 1, n);

cin >> q;

for(int i=1; i<=q; i++)

{

int l, r; cin >> l >> r;

Query.pb({{r, l}, i});

}

sort(all(Query));

int indexQ = 0;

map<int, int> ok;

for(int i=1; i<=n; i++)

{

if(ok[a[i]] != 0)

{

Update(1, 1, n, ok[a[i]], 0);

}

ok[a[i]] = i;

while(Query[indexQ].fi.fi == i)

{

endgame[Query[indexQ].se] = Get(1, 1, n, Query[indexQ].fi.se, Query[indexQ].fi.fi);

indexQ++;

}

}

for(int i=1; i<=q; i++)

{

cout << endgame[i] << ed;

}

}

**Segment tree lazy**

int n, q, b, R, V;

const int N = 2e5+10;

int arr[N];

int st[4\*N];

int lazy[4\*N];

void update(int id, int l , int r, int u, int v, int val) {

if(l>v||r<u)

return;

if(l>=u&&r<=v) {

st[id]+=val;

lazy[id]+=val;

return;

}

int mid = (l+r)>>1;

update(id\*2, l , mid, u, v, val);

update(id\*2+1, mid+1 , r, u, v, val);

st[id]+=st[id\*2]+st[id\*2+1]+lazy[id];

}

int query(int id, int l, int r, int u, int v) {

if(l>v||r<u)

return 0;

if(l>=u&&r<=v) {

return st[id];

}

int mid = (l+r)>>1;

int t1=query(id\*2, l , mid, u, v);

int t2=query(id\*2+1, mid+1 , r, u, v);

return t1+t2+lazy[id];

}

pair<int,int> check(int l, int r, int val) {

int res=val;

while(l<r) {

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

int cur =query(1, 1, n, mid, mid);

if(cur>val) {

l=mid;

res=cur;

}

else

r=mid-1;

}

return make\_pair(l, res);

}

void solve() {

cin >> n >> q >> b;

for(int i=1; i<=q; i++) {

cin >> R >> V;

while(V>0) {

int val=query(1, 1, n, R, R);

pair<int,int> it=check(0, R, val);

int res=it.S;

int bonus=min(res-val, V/(R-it.F));

if(bonus!=0) {

update(1, 1, n, it.F+1, R, bonus);

V-=bonus\*(R-it.F);

}

else {

if(res==val&&(V/(R-it.F))!=0) {

update(1, 1, n, it.F+1, R, (V/(R-it.F)));

V-=(V/(R-it.F))\*(R-it.F);

}

else {

update(1, 1, n, it.F+1, it.F+V%(R-it.F), 1);

V-=V%(R-it.F);

}

}

}

}

for(int i=1; i<=n; i++) {

cout<<query(1, 1, n, i, i)+b<<" ";

}

}

**Lazy**

Cho dãy số A với N phần tử (N≤50,000). Bạn cần thực hiện 2 loại truy vấn:

Cộng tất cả các số trong đoạn [l,r] lên giá trị val.

In ra giá trị lớn nhất của các số trong đoạn [l,r]

struct Node {

int lazy; // giá trị T trong phân tích trên

int val; // giá trị lớn nhất.

} nodes[MAXN \* 4];

void down(int id) {

int t = nodes[id].lazy;

nodes[id\*2].lazy += t;

nodes[id\*2].val += t;

nodes[id\*2+1].lazy += t;

nodes[id\*2+1].val += t;

nodes[id].lazy = 0;

}

void update(int id, int l, int r, int u, int v, int val) {

if (v < l || r < u) {

return ;

}

if (u <= l && r <= v) {

// Khi cài đặt, ta LUÔN ĐẢM BẢO giá trị của nút được cập nhật ĐỒNG THỜI với

// giá trị lazy propagation. Như vậy sẽ tránh sai sót.

nodes[id].val += val;

nodes[id].lazy += val;

return ;

}

int mid = (l + r) / 2;

down(id); // đẩy giá trị lazy propagation xuống các con

update(id\*2, l, mid, u, v, val);

update(id\*2+1, mid+1, r, u, v, val);

nodes[id].val = max(nodes[id\*2].val, nodes[id\*2+1].val);

}

int get(int id, int l, int r, int u, int v) {

if (v < l || r < u) {

return -INFINITY;

}

if (u <= l && r <= v) {

return nodes[id].val;

}

int mid = (l + r) / 2;

down(id); // đẩy giá trị lazy propagation xuống các con

return max(get(id\*2, l, mid, u, v),

get(id\*2+1, mid+1, r, u, v));

// Trong các bài toán tổng quát, giá trị ở nút id có thể bị thay đổi (do ta đẩy lazy propagation

// xuống các con). Khi đó, ta cần cập nhật lại thông tin của nút id dựa trên thông tin của các con.

}