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

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

<u>Chiến thuật & lưu ý:</u>
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

<u>Trong những lúc khó khăn:</u>
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 đẳng 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

<u>Trước khi submit:</u>
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

<u>Wrong answer:</u>
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

<u>Runtime error:</u>
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).

<u>Time litmit exceeded:</u>
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?

<u>Memory limit exceeded:</u>
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?

2. MATHEMATICS

Các phép toán:

$2^5 = 32$
$2^6 = 64$
$2^7 = 128$
$2^8 = 256$
$2^9 = 512$
$2^{10} = 1024$
$2^{11} = 2048$
$2^{12} = 4096$
$2^{13} = 8\ 192$
$2^{14} = 16\ 384$
$2^{15} = 32\ 768$
$2^{16} = 65\ 536$
$2^{17} = 131\ 072$
$2^{18} = 262\ 144$
$2^{19} = 524\ 288$
$2^{20} = 1\ 048\ 576$
$2^{21} = 2\ 097\ 152$
$2^{22} = 4\ 194\ 304$
$2^{23} = 8\ 388\ 608$
$2^{24} = 16\ 777\ 216$
$2^{25} = 33\ 554\ 432$
$2^{26} = 67\ 108\ 864$
$2^{27} = 134\ 217\ 728$

$2! = 2$
$3! = 6$
$4! = 24$
$5! = 120$
$6! = 720$
$7! = 5\ 040$
$8! = 40\ 320$
$9! = 362\ 880$
$10! = 3\ 628\ 800$
$11! = 39\ 916\ 800$
$12! = 479\ 001\ 600$
$13! = 6\ 227\ 020\ 800$
$14! = 8.7 \cdot 10^{10}$
$15! = 1.3 \cdot 10^{12}$
$16! = 2 \cdot 10^{13}$
$17! = 3.55 \cdot 10^{14}$
$20! = 2.4 \cdot 10^{18}$

Phương trình:

Delta > 0 (2 nghiệm):

$$ax^2 + bx + c = 0 \Rightarrow x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Delta = 0 (1 nghiệm): $x = -b/2a.$

Sums

$$c^a + c^{a+1} + \cdots + c^b = \frac{c^{b+1} - c^a}{c - 1}, c \neq 1$$

$$1 + 2 + 3 + \cdots + n = \frac{n(n+1)}{2}$$

$$1^2 + 2^2 + 3^2 + \cdots + n^2 = \frac{n(2n+1)(n+1)}{6}$$

$$1^3 + 2^3 + 3^3 + \cdots + n^3 = \frac{n^2(n+1)^2}{4}$$

$$1^4 + 2^4 + 3^4 + \cdots + n^4 = \frac{n(n+1)(2n+1)(3n^2+3n-1)}{30}$$

$$1.2.3 + 2.3.4 + \ldots + n(n+1)(n+2) = \frac{n(n+1)(n+2)(n+3)}{4}$$

$$1.1! + 2.2! + \ldots + n.n! = (n+1)! - 1$$

$$\frac{1}{2!} + \frac{2}{3!} + \ldots + \frac{n}{(n+1)!} = 1 - \frac{1}{(n+1)!}$$

$$\frac{1}{1.2.3} + \frac{1}{2.3.4} + \ldots + \frac{1}{n(n+1)(n+2)} = \frac{n(n+3)}{4(n+1)(n+2)}$$

Cấp số cộng

$$S_n = u_1 + u_2 + \ldots + u_n = \frac{n(u_1 + u_n)}{2}$$

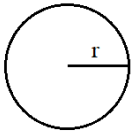
$$S_n = n.U_1 + \frac{n.(n-1)}{2} d \ (n \geqslant 2)$$

Cấp số nhân

$$S_n = U_1 + U_2 + \ldots + U_n = U_1 \frac{1 - q^n}{1 - q}$$

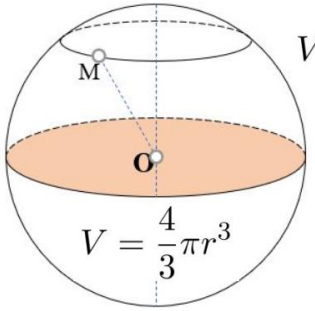
Công thức hình học

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



$$S = \pi r^2 = \frac{\pi d^2}{4} \quad d = 2 \cdot r$$

$$P = 2 \cdot \pi r = \pi d \quad \pi = 3,14$$



$$V = \frac{4}{3}\pi r^3$$

$$\mathbf{S} = 4 \cdot \pi r^2$$

$$d = 2 \cdot r$$

V – thể tích
S – diện tích
d – đường kính
r – bán kính
O – tâm

Bán chu vi: p = (a + b + c) / 2

Diện tích: $A = \sqrt{p(p-a)(p-b)(p-c)}$

Nội tiếp: Đường tròn: $R = \frac{abc}{4A}$ bán kính: $r = \frac{A}{p}$

Độ dài đường trung tuyến (chia tam giác thành hai tam giác có diện tích bằng nhau): $m_a = \frac{1}{2}\sqrt{2b^2 + 2c^2 - a^2}$

Độ dài đường phân giác (chia hai góc):

$$s_a = \sqrt{bc\left[1 - \left(\frac{a}{b+c}\right)^2\right]}$$

Định luật Sin: $\frac{\sin\alpha}{a} = \frac{\sin\beta}{b} = \frac{\sin\gamma}{c} = \frac{1}{2R}$

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 =

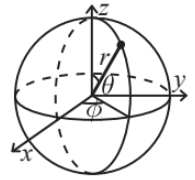
$$F = b^2 + d^2 - a^2 - c^2$$

$$4A = 2ef \cdot \sin\theta = F\tan\theta = \sqrt{4e^2f^2 - F^2}$$

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

$$ef = ac + bd, \text{ and } A = \sqrt{(p-a)(p-b)(p-c)(p-d)}.$$

Tọa độ cầu:



$$x = r\sin\theta\cos\phi \qquad r = \sqrt{x^2 + y^2 + z^2}$$
$$y = r\sin\theta\sin\phi \qquad \theta = \operatorname{acos}(z/\sqrt{x^2 + y^2 + z^2})$$
$$z = r\cos\theta \qquad \phi = \operatorname{atan2}(y, x)$$

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) $S_n = 6n(n-1) + 1$

Bộ ba số Pythagoras

$$a = k(m^2 - n^2)$$
$$b = k2mn$$
$$c = k(m^2 + n^2)$$

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 (10^{18})

Trong một bộ ba Pythagoras nguyên thủy, ký hiệu:

Hai cạnh góc vuông:

$m^2 - n^2$ và $2mn$ là 2 cạnh góc vuông a,b; trong đó $2mn$ là cạnh góc vuông chẵn.

$c = m^2 + n^2$ là cạnh huyền.

- Mối liên hệ khác giữa ba số trong bộ ba Pytagoras,

$$a + b = c + 2\sqrt{\frac{(c-a)(c-b)}{2}}$$

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

3. NUMBER THEORY

}

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 10^9 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 = 10^{10})

```
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;
}
```

Kiểm tra một số (108 chữ số) có chia hết cho một số

```
bool check(string &n, long long k)
{
   long long rem = 0;
   for (auto i : n)
      rem = ((rem * 10) + i - '0') % k;
   return rem == 0;
}
```

NCK (Tổ hợp)

```
ll nck(ll n, ll k) {

        for(int i=k+1; i<=n;i++) {
                int res=i;
                while(res>1) {
                        cnt[prime[res]]++;
                        res/=prime[res];
                }
        }
        for(int i=n-k; i>1;i--) {
                int res=i;
                while(res>1) {
                        cnt[prime[res]]--;
                        res/=prime[res];
                }
        }
        ll q=1;
        for(int i=2; i<=n; i++) {
                if(cnt[i]>0) {
                        ll ans=1, count =cnt[i], base= i;
                                while(count >=1) {

        if(count&1)

        (ans*=base)%=mod;

        (base*=base)%=mod;

        count>>=1;
                                }
                        q*=ans;
                        q%=mod;
                }
        }
        return q;
}
}
```

Modulo Giai thừa

```
ll giaiThua[N], nghichDao[N];
void TINHGIAITHUA() {
        giaiThua[0]=1;
        ll i;
  for (i = 1; i < N; i++) {
        if(i>18)
                giaiThua[i] = (i %mod* giaiThua[i -
1]%mod)%mod;
        else
         giaiThua[i] = (i * giaiThua[i - 1]);
  }
//   i--;
//   nghichDao[i] = binpow(giaiThua[i], phi(mod)-1, mod);
//   for (i--; i >= 0; i--)
//   {
//      nghichDao[i] = nghichDao[i + 1] * (i + 1) % mod;
//   }
}
```

Những số nguyên tố lớn : 1073676287, 68718952447, 274876858367, 4398042316799,...

Số catalan

$$C_n = \frac{1}{n+1}\binom{2n}{n} = \frac{(2n)!}{(n+1)!n!} = \prod_{k=2}^{n}\frac{n+k}{k} \text{ với } n \geq 0$$

Những số catalan đầu tiên với n = 0, 1, 2, 3, ...là:
1, 1, 2, 5, 14, 42, 132, 429, 1430, 4862, 16796, 58786, 208012, 742900, 2674440, 9694845, 35357670, 129644790, 477638700, 1767263190, 6564120420, 24466267020, 91482563640, 343059613650, 1289904147324, 4861946401452,...

Số dư trung hoa

$$M = m_1 \cdot m_2 \cdot \ \ldots \ \cdot m_k$$

$$x \equiv a_1 \cdot M_1 \cdot y_1 + a_2 \cdot M_2 \cdot y_2 + \ldots + a_k \cdot M_k \cdot y_k \pmod{M}$$

$$M_1 = \frac{M}{m_1},\ M_2 = \frac{M}{m_2},\ \ldots,\ M_k = \frac{M}{m_k}$$

$$y_1 = (M_1)^{-1} \pmod{m_1},\ y_2 = (M_2)^{-1} \pmod{m_2},\ \ldots,\ y_k = (M_k)^{-1} \pmod{m_k}$$

Trong đó

$(M_1)^{-1} \pmod{m_1}$ *là nghịch đảo theo mô-đun của* m_1 *với* $y_1 = (M_1)^{-1} \pmod{m_1} \Leftrightarrow y_1 M_1 = 1 \pmod{m_1}$

Đồng dư

x^2 đồng dư a mod p sẽ có 2 nghiệm or k có nghiệm nào và nó thỏa mãn 2 nghiệm khi mà a^[(p-1)/2] đồng dư 1 mod p

4. GEOMETRY

Diện tích đa giác

```
ll n; cin >> n;
ll x[n+1], y[n+1];
```

```
        for(int i=0; i<n; i++)                                                  }
  {                                                                            }
  cin >> x[i] >> y[i];
  }
  x[n] = x[0], y[n] = y[0];
  ll ans = 0;
  for(int i=0; i<n; i++)
  {
  ans += (x[i] * y[i+1] - y[i] * x[i+1]);
  }
  cout << abs(ans) << ed;
```

Đoạn thẳng giao nhau

```
struct toado
{
  ll x, y;
};
struct DT
{
  ll A, B, C;
};

bool check_inside(ll x, ll y, ll u1, ll v1, ll u2, ll v2)
{
  return (x >= min(u1, u2) && x <= max(u1, u2) && y >= min(v1, v2) && y <= max(v1, v2));
}

toado Cal_vector(toado A, toado B)
{
  toado ans;
  ans.x = B.x - A.x;
  ans.y = B.y - A.y;
  return ans;
}

ll Tich_co_huong(toado A, toado B)
{
  return A.x * B.y - A.y * B.x;
}

int main()
{
  Hello_i_am_Salmon
  ll t; cin >> t;
  while(t--)
  {
  toado A, B, C, D;
  cin >> A.x >> A.y >> B.x >> B.y >> C.x >> C.y >> D.x >> D.y;
  toado AB = Cal_vector(A, B);
  toado AC = Cal_vector(A, C);
  toado AD = Cal_vector(A, D);
  toado CD = Cal_vector(C, D);
  toado CA = Cal_vector(C, A);
  toado CB = Cal_vector(C, B);

  ll ans = 0;
  if(Tich_co_huong(AB, AC) == 0)
  if(check_inside(C.x, C.y, A.x, A.y, B.x, B.y)) ans = 1;
  if(Tich_co_huong(AB, AD) == 0)
  if(check_inside(D.x, D.y, A.x, A.y, B.x, B.y)) ans = 1;
  if(Tich_co_huong(CD, CA) == 0)
  if(check_inside(A.x, A.y, C.x, C.y, D.x, D.y)) ans = 1;
  if(Tich_co_huong(CD, CB) == 0)
  if(check_inside(B.x, B.y, C.x, C.y, D.x, D.y)) ans = 1;
  if(((Tich_co_huong(AB, AC) < 0 && Tich_co_huong(AB, AD) > 0) || (Tich_co_huong(AB,
AC) > 0 && Tich_co_huong(AB, AD) < 0)) && ((Tich_co_huong(CD, CA) < 0 &&
Tich_co_huong(CD, CB) > 0) || (Tich_co_huong(CD, CA) > 0 && Tich_co_huong(CD, CB) <
0))) ans = 1;
  cout << (ans ? "YES" : "NO") << ed;
```

Kiểm tra 3 điểm CCW, CW

```
ll x1, y1, x2, y2, x3, y3; cin >> x1 >> y1 >> x2 >> y2 >> x3 >> y3;
        ll x12 = x2 - x1;
        ll y12 = y2 - y1;
        ll x13 = x3 - x1;
        ll y13 = y3 - y1;
        ll tich_co_huong = x12 * y13 - x13 * y12;
        if(tich_co_huong == 0) cout << "TOUCH" << ed;
        else if(tich_co_huong > 0) cout << "LEFT" << ed;
        else cout << "RIGHT" << ed;
```

Góc giữa 2 vector

$$\cos(\vec{a},\vec{b}) = \frac{a_1b_1 + a_2b_2 + a_3b_3}{\sqrt{a_1^2 + a_2^2 + a_3^2}\sqrt{b_1^2 + b_2^2 + b_3^2}} \ ;$$

Convex hull

```
struct pt { // Kiểu điểm
  double x, y;
};

bool cmp (pt a, pt b)
{ // So sánh theo tọa độ x, trong trường hợp bằng nhau so sánh theo y
  return a.x < b.x || a.x == b.x && a.y < b.y;
}

bool cw (pt a, pt b, pt c) { // a -> b -> c đi theo thứ tự xuôi chiều kim đồng
hồ
  return a.x*(b.y-c.y)+b.x*(c.y-a.y)+c.x*(a.y-b.y) < 0;
}

bool ccw (pt a, pt b, pt c) { // a -> b -> c đi theo thứ tự ngược chiều kim
đồng hồ
  return a.x*(b.y-c.y)+b.x*(c.y-a.y)+c.x*(a.y-b.y) > 0;
}

void convex_hull (vector<pt> & a) {
  if (a.size() == 1) { // chỉ có 1 điểm
    return;
  }

  // Sắp xếp các điểm theo tọa độ x, nếu bằng nhau sắp xếp theo y
  sort (a.begin(), a.end(), &cmp);

  pt p1 = a[0],  p2 = a.back();

  vector<pt> up, down; // chuỗi trên và chuỗi dưới
  up.push_back (p1);
  down.push_back (p1);

  for (size_t i=1; i<a.size(); ++i) { // xét lần lượt các điểm
    // Thêm vào chuỗi trên
    if (i==a.size()-1 || cw (p1, a[i], p2)) {
      while (up.size()>=2 && !cw (up[up.size()-2], up[up.size()-1], a[i]))
        up.pop_back();
      up.push_back (a[i]);
    }

    // Thêm vào chuỗi dưới
    if (i==a.size()-1 || ccw (p1, a[i], p2)) {
      while (down.size()>=2 && !ccw (down[down.size()-2],
down[down.size()-1], a[i]))
        down.pop_back();
      down.push_back (a[i]);
    }
  }

  // Gộp 2 chuỗi trên và dưới để lấy bao lồi
  a.clear();
  for (size_t i=0; i<up.size(); ++i)
    a.push_back (up[i]);
  for (size_t i=down.size()-2; i>0; --i)
    a.push_back (down[i]);
}
```

Tích vô hướng giữa 2 vector

```
ll Tich_vo_huong(toado AB, toado AC)
{
        return AB.x * AC.y + AB.y * AC.x;
}
```

Tích có hướng giữa 2 vector

```
ll Tich_co_huong(toado AB, toado AC)
{
        return AB.x * AC.x - AB.y * AC.y;
}
```

Tính độ lớn của vector

```
ll Distance(toado A)
{
        return sqrt(A.x * A.x + A.y * A.y);
}
```

Tính khoảng cách từ điểm C đến đường thẳng /đoạn thẳng AB

```
double LinePointDist(toado AB, toado AC, toado BC, isSegment)
{
        double dist = abs(Tich_co_huong(AB, AC)) / Distance(AB);
        if(isSegment)
        {
                ll dot1 = Tich_vo_huong(BA, BC);
                if(dot1 < 0) return Distance(BC);
                ll dot2 = Tich_vo_huong(AB, AC);
                if(dot2 < 0) return Distance(AC);
        }
        return dist;
}
```

Mọi thứ trở nên phức tạp hơn khi ta muốn tìm khoảng cách từ một đoạn thẳng đến một điểm. Trong trường hợp này điểm gần nhất có thể 1 trong 2 đầu mút của đoạn thẳng thay vì một điểm nào đó trên đoạn thẳng.

Các giải quyết:

Tích vô hướng: Kiểm tra xem điểm gần nhất trên đường thẳng AB có phải A hoặc B không? bằng cách tính $\overrightarrow{BA} . \overrightarrow{BC}$ đối với B và $\overrightarrow{AB} . \overrightarrow{AC}$ đối với A nếu một trong 2 tích âm thì góc ở đó là góc tù nên điểm gần nhất sẽ là điểm tạo ra góc tù đó, nếu tất cả đều lớn hơn 0 thì điểm gần C nhất sẽ nằm giữa A và B

Giao điểm của 2 đường thẳng

```
double check = A1 * B2 - A2 * B1;
        if(check == 0)
        {
                if(A1 * C2 == A2 * C1)
```

```
{
// Trung nhau
}
else
{
// Song song
}
}
else
{
double x = (B2 * C1 - B1 * C2) / check;
double y = (A1 * C2 - A2 * C1) / check;
}
```

5. 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.
```

Mảng cộng dồn 2 chiều

$$S_{i,\,j} = S_{i-1,\,j} + S_{i,\,j-1} - S_{i-1,\,j-1} + A_{i-1,\,j-1}$$

Spare table

```
void process2(int M[MAXN][LOGMAXN], int A[MAXN], int N)
{
  int i, j;

  // Khởi tạo M với các khoảng độ dài 1
  for (i = 0; i < N; i++)
    M[i][0] = i;

  // Tính M với các khoảng dài 2^j
  for (j = 1; 1 << j <= N; j++)
    for (i = 0; i + (1 << j) - 1 < N; i++)
      if (A[M[i][j - 1]] < A[M[i + (1 << (j - 1))][j - 1]])
        M[i][j] = M[i][j - 1];
      else
        M[i][j] = M[i + (1 << (j - 1))][j - 1];
}
```

Sort struct

```
struct book
{
  ll t, a, b;
};

bool comp(book x, book y)
{
  if(x.t != y.t) return x.t > y.t;
  if(x.a != y.a) return x.a > y.a;
  return x.b > y.b;
}

sort(A, A+n, comp);
```

Z-function

```
vector<int> z_function(string s) {
    int n = (int) s.length();
    vector<int> z(n);
    for (int i = 1, l = 0, r = 0; i < n; ++i) {
        if (i <= r)
            z[i] = min (r - i + 1, z[i - l]);
        while (i + z[i] < n && s[z[i]] == s[i + z[i]])
            ++z[i];
        if (i + z[i] - 1 > r)
            l = i, r = i + z[i] - 1;
    }
    return z;
}
```

Trie

```
#include<cstdio>
#include<cstring>

const int N = 4000 + 2, L = 300000 + 2, WL = 100 + 2, MOD = 1337377;
struct TrieNode {
    int next[26], nfinish;
} trie[N * WL];
int nnode = 1, n, f[L];
char tmp[WL], s[L];

void insert(const char * s) {
    int u = 0;
    for(int i = strlen(s) - 1; i >= 0; --i) {
        int t = s[i] - 0x61;
        if(trie[u].next[t] == 0) trie[u].next[t] = nnode++;
        u = trie[u].next[t];
    }
    trie[u].nfinish = 1;
}

void enter() {
    scanf("%s%d", s+1, &n);
    for(int i = 0; i < n; ++i) {
        scanf("%s", tmp);
        insert(tmp);
    }
}

void solve() {
    f[0] = 1; int l = strlen(s + 1);
    for(int i = 1; i <= l; ++i) {
        int u = 0;
        for(int j = i; j && trie[u].next[s[j] - 0x61]; --j)
            if(trie[u = trie[u].next[s[j] - 0x61]].nfinish)
                f[i] = (f[j-1] + f[i]) % MOD;
    }
    printf("%d\n", f[l]);
```

Fenwick tree 2D

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

#define N 4 // N-->max_x and max_y

// A structure to hold the queries
struct Query
{
        int x1, y1; // x and y co-ordinates of bottom left
        int x2, y2; // x and y co-ordinates of top right
};

// A function to update the 2D BIT
void updateBIT(int BIT[][N+1], int x, int y, int val)
{
        for (; x <= N; x += (x & -x))
        {
                // This loop update all the 1D BIT inside the
                // array of 1D BIT = BIT[x]
                for (int yy=y; yy <= N; yy += (yy & -yy))
                        BIT[x][yy] += val;
        }
        return;
}

// A function to get sum from (0, 0) to (x, y)
int getSum(int BIT[][N+1], int x, int y)
{
        int sum = 0;

        for(; x > 0; x -= x&-x)
        {
                // This loop sum through all the 1D BIT
                // inside the array of 1D BIT = BIT[x]
                for(int yy=y; yy > 0; yy -= yy&-yy)
                {
                        sum += BIT[x][yy];
                }
        }
        return sum;
}

// A function to create an auxiliary matrix
// from the given input matrix
void constructAux(int mat[][N], int aux[][N+1])
{
        // Initialise Auxiliary array to 0
        for (int i=0; i<=N; i++)
                for (int j=0; j<=N; j++)
                        aux[i][j] = 0;

        // Construct the Auxiliary Matrix
        for (int j=1; j<=N; j++)
                for (int i=1; i<=N; i++)
                        aux[i][j] = mat[N-j][i-1];

        return;
}

// A function to construct a 2D BIT
void construct2DBIT(int mat[][N], int BIT[][N+1])
{
        // Create an auxiliary matrix
        int aux[N+1][N+1];
        constructAux(mat, aux);

        // Initialise the BIT to 0
        for (int i=1; i<=N; i++)
                for (int j=1; j<=N; j++)
                        BIT[i][j] = 0;

        for (int j=1; j<=N; j++)
        {
                for (int i=1; i<=N; i++)
                {
                        // Creating a 2D-BIT using
update function
                        // everytime we/ encounter a
value in the
                        // input 2D-array
                        int v1 = getSum(BIT, i, j);
                        int v2 = getSum(BIT, i, j-1);
                        int v3 = getSum(BIT, i-1, j-1);
                        int v4 = getSum(BIT, i-1, j);

                        // Assigning a value to a
particular element
                        // of 2D BIT
                        updateBIT(BIT, i, j, aux[i][j]-(v1-
v2-v4+v3));
                }
        }

        return;
}

// A function to answer the queries
void answerQueries(Query q[], int m, int BIT[][N+1])
{
        for (int i=0; i<m; i++)
        {
                int x1 = q[i].x1 + 1;
                int y1 = q[i].y1 + 1;
                int x2 = q[i].x2 + 1;
                int y2 = q[i].y2 + 1;

                int ans = getSum(BIT, x2, y2)-getSum(BIT, x2,
y1-1)-
                                        getSum(BIT, x1-1,
y2)+getSum(BIT, x1-1, y1-1);

                printf ("Query(%d, %d, %d, %d) = %d\n",
                                        q[i].x1, q[i].y1,
q[i].x2, q[i].y2, ans);
        }
        return;
}
```

```
// Driver program
int main()
{
        int mat[N][N] = {{1, 2, 3, 4},
                                                {5, 3, 8, 1},
                                                {4, 6, 7, 5},
                                                {2, 4, 8, 9}};

        // Create a 2D Binary Indexed Tree
        int BIT[N+1][N+1];
        construct2DBIT(mat, BIT);

        /* Queries of the form - x1, y1, x2, y2
        For example the query- {1, 1, 3, 2} means the sub-matrix-
        y
        /\
3 |      1 2 3 4       Sub-matrix
2 |      5 3 8 1       {1,1,3,2}      --->        3 8 1
1 |      4 6 7 5
                                                   6 7 5
0 |      2 4 8 9
         |
--|----- 0 1 2 3 ----> x
         |

        Hence sum of the sub-matrix = 3+8+1+6+7+5 = 30

        */

        Query q[] = {{1, 1, 3, 2}, {2, 3, 3, 3}, {1, 1, 1, 1}};
        int m = sizeof(q)/sizeof(q[0]);

        answerQueries(q, m, BIT);

        return(0);
}
```

Segment tree 2D

```
void build(int a[], int v, int tl, int tr) {
  if (tl == tr) {
    t[v] = a[tl];
  } else {
    int tm = (tl + tr) / 2;
    build(a, v*2, tl, tm);
    build(a, v*2+1, tm+1, tr);
    t[v] = t[v*2] + t[v*2+1];
  }
}
int sum(int v, int tl, int tr, int l, int r) {
  if (l > r)
    return 0;
  if (l == tl && r == tr) {
    return t[v];
  }
  int tm = (tl + tr) / 2;
  return sum(v*2, tl, tm, l, min(r, tm))
       + sum(v*2+1, tm+1, tr, max(l, tm+1), r);
}
void update(int v, int tl, int tr, int pos, int new_val) {
  if (tl == tr) {
    t[v] = new_val;
  } else {
    int tm = (tl + tr) / 2;
    if (pos <= tm)
      update(v*2, tl, tm, pos, new_val);
    else
      update(v*2+1, tm+1, tr, pos, new_val);
    t[v] = t[v*2] + t[v*2+1];
  }
}

// 2

pair<int, int> t[4*MAXN];

pair<int, int> combine(pair<int, int> a, pair<int, int> b) {
  if (a.first > b.first)
    return a;
  if (b.first > a.first)
    return b;
  return make_pair(a.first, a.second + b.second);
}

void build(int a[], int v, int tl, int tr) {
  if (tl == tr) {
    t[v] = make_pair(a[tl], 1);
  } else {
    int tm = (tl + tr) / 2;
    build(a, v*2, tl, tm);
    build(a, v*2+1, tm+1, tr);
    t[v] = combine(t[v*2], t[v*2+1]);
  }
}

pair<int, int> get_max(int v, int tl, int tr, int l, int r) {
  if (l > r)
    return make_pair(-INF, 0);
  if (l == tl && r == tr)
    return t[v];
  int tm = (tl + tr) / 2;
  return combine(get_max(v*2, tl, tm, l, min(r, tm)),
                 get_max(v*2+1, tm+1, tr, max(l, tm+1), r));
}

void update(int v, int tl, int tr, int pos, int new_val) {
  if (tl == tr) {
    t[v] = make_pair(new_val, 1);
  } else {
    int tm = (tl + tr) / 2;
    if (pos <= tm)
      update(v*2, tl, tm, pos, new_val);
    else
      update(v*2+1, tm+1, tr, pos, new_val);
    t[v] = combine(t[v*2], t[v*2+1]);
  }
}
```

Deque tịnh tiến max min

Cho một dãy A gồm N phần tử được đánh số từ 1 đến N. Phần tử thứ i có giá trị là A[i]. Cho k là một số nguyên dương (k≤N). Với mỗi phần tử i (k≤i≤N), tìm giá trị nhỏ nhất của các phần tử trong đoạn từ i−k+1 đến i trên dãy A. minRange[i]= giá trị nhỏ nhất trong đoạn [i−k+1…i]

```
deque <int> dq;

/* Làm rỗng deque */
while (dq.size()) dq.pop_front();

/* Duyệt lần lượt các phần tử từ 1 đến N */
for (int i = 1; i <= N; ++i) {
  /* Loại bỏ các phần tử có giá trị lớn hơn hoặc bằng A[i] */
  while (dq.size() && A[dq.back()] >= A[i]) dq.pop_back();

  /* Đẩy phần tử i vào queue */
  dq.push_back(i);

  /* Nếu phần tử đầu tiên trong deque nằm ngoài khoảng tính
     thì ta sẽ loại bỏ ra khỏi deque */
  if (dq.front() + k <= i) dq.pop_front();

  /* minRange[i] là giá trị nhỏ nhất trong đoạn [i – k + 1 … i] */
  if (i >= k) minRange[i] = A[dq.front()];
}
```

6. Graph

7. BFS loang

8.

```
9.  ll moveX[] = {0, 1, -1, 0, 1, -1, -1, 1};
10. ll moveY[] = {1, 0, 0, -1, 1, -1, 1, -1};
11. bool vis[Nn][Nn];
12. char a[Nn][Nn];
13. ll n, m;
14. bool check_id(ll x, ll y)
15. {
16.     return (x >= 0 && x < n && y >= 0 && y < m);
17. }
18.
19. vector<pair<ll, ll>> lo;
20.
21. void bfs(ll sx, ll sy)
22. {
23.     queue<pair<ll, ll>> Q;
24.     vis[sx][sy] = true;
25.     Q.push({sx, sy});
26.     while(!Q.empty())
27.     {
28.                 ll x = Q.front().fi;
29.                 ll y = Q.front().se;
30.                 lo.pb({x, y});
31.                 // fix(sz(lo));
32.                 Q.pop();
33.                 for(int i=0; i<8; i++)
34.                 {
35.                             ll u = x + moveX[i];
36.                             ll v = y + moveY[i];
37.                             if(!check_id(u, v))
38.                             {
39.                                         // show2
40.                                         continue;
41.                             }
42.                             if(a[u][v] == '*' && !vis[u][v])
43.                             {
44.                                         // show1
45.
46.                                         vis[u][v] = true;
47.                                         Q.push({u, v});
48.                                         // a[u][v] = '.';
49.                             }
50.                 }
51.     }
52.
53. }
54.
55. int main()
56. {
57.     Hello_i_am_Salmon
58.     int t; cin >> t;
59.     while(t--)
60.     {
61.                 cin >> n >> m;
62.
```

```
  for(int i=0; i<n; i++)
  {
  for(int j=0; j<m; j++)
  {
  cin >> a[i][j];
  }
  }
  ll endgame = 1;
  for(int i=0; i<n; i++)
  {
  for(int j=0; j<m; j++)
  {
  if(a[i][j] == '*' && !vis[i][j])
  {
  bfs(i, j);
  if(sz(lo) != 3) endgame = 0;
  else
  {
  sort(all(lo));
  ll x1 = lo[0].fi;
  ll y1 = lo[0].se;
  ll x2 = lo[1].fi;
  ll y2 = lo[1].se;
  ll x3 = lo[2].fi;
  ll y3 = lo[2].se;
  ll f = 0;
  if(x2 == x1+1 && y2 == y1 && x3 == x1+1 && y3 == y1+1) f = 1;
  if(x2 == x1+1 && y2 == y1-1 && x3 == x1+1 && y3 == y1) f = 1;
  if(x2 == x1 && y2 == y1+1 && x3 == x1+1 && y3 == y1+1) f = 1;
  if(x2 == x1 && y2 == y1+1 && x3 == x1+1 && y3 == y1) f = 1;
  if(!f) endgame = 0;
  }
  lo.clear();
  }
  }
  }
  cout << (endgame ? "YES" : "NO") << ed;
  for(int i=0; i<n; i++)
  {
  for(int j=0; j<m; j++)
  {
  vis[i][j] = 0;
  }
  }

  }
}
```

DFS

```
int timeDfs = 0; // Thứ tự duyệt DFS

void dfs(int u, int pre) {
   num[u] = low[u] = ++timeDfs;
   for (int v : g[u]){
      if (v == pre) continue;
      if (!num[v]) {
         dfs(v, u);
         low[u] = min(low[u], low[v]);
      }
      else low[u] = min(low[u], num[v]);
   }
   tail[u] = timeDfs;
}
```

Dijkstra

```
vector<pair<ll, ll>> adj[N];
ll pre[N];
ll n, m;
void dijkstra(ll s)
{
  vector<ll> d(n+1, INF); // mang luu khoang cach duong di
  d[s] = 0;
  priority_queue<pair<ll, ll>, vector<pair<ll, ll>>, greater<pair<ll, ll>>> Q;
  // {khoang cach, dinh}
  Q.push({0, s});
  while(!Q.empty())
  {
    // show1
    pair<ll, ll> temp = Q.top(); Q.pop();
    ll u = temp.se, kc = temp.fi;
    if(kc > d[u]) continue;
    for(auto it : adj[u])
    {
                ll v = it.fi, w = it.se;
                if(d[v] > d[u] + w)
                {
                            d[v] = d[u] + w;
                                                Q.push({d[v], v});
                                                pre[v] = u;
                                    }
                            }
                }
                vector<ll> path;
                ll endd = n;
                if(d[endd] == INF) cout << -1 << ed;
                else
                {
                            while(1)
                            {
                                        path.pb(endd);
                                        if(endd == 1) break;
                                        endd = pre[endd];
                            }
                            reverse(all(path));
                            for(auto it : path) cout << it << _;
                            cout << ed;
                }
    }
    int main()
    {
                Hello_i_am_Salmon
                cin >> n >> m;
                for(int i=0; i<m; i++)
                {
                            ll x, y, z; cin >> x >> y >> z;
                            adj[x].pb({y, z});
                            adj[y].pb({x, z});
                }
                dijkstra(1);
    }
```

DSU

```
ll parent[maxN];
ll sz[maxN];

void make_set(ll v)
{
            parent[v] = v;
            sz[v] = 1;
}

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

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

int main()
{
            Hello_i_am_Salmon
            int t; cin >> t;
            while(t--)
            {
                        ll n; cin >> n;
                        ll a[n+1];
                        for(int i=1; i<=n; i++) cin >> a[i];
                        for(int i=1; i<=n; i++)
                        {
                                    make_set(i);
                        }
                        for(int i=1; i<=n; i++)
                        {
                                    union_sets(i, a[i]);
                        }
                        for(int i=1; i<=n; i++)
                        {
                                    find_set(i);
                        }
                        cout << endl;
                        for(int i=1; i<=n; i++)
                        {
                                    cout << sz[parent[i]] << " ";
                        }
                        cout << endl;
            }

}
```

DSU lưu sum min max

```
void make_set(int v) {
   parent[v] = v;
   sz[v] = 1;
   mn[v] = value[v];
   sum[v] = value[v];
   // value[v] là giá trị của phần tử thứ v
}

int find_set(int v) {
   return v == parent[v] ? v : 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 (sz[a] < sz[b]) swap(a, b);
      parent[b] = a;
```

```
        sz[a] += sz[b];                                                    }
        sum[a] += sum[b];
        mn[a] = min(mn[a], mn[b]);
    }
}

int find_sum(int v) { // Trả về tổng của các phần tử trong tập hợp chứa v
    v = find_set(v);
    return sum[v];
}

int find_min(int v) { // Trả về giá trị bé nhất của các phần tử trong tập hợp chứa v
    v = find_set(v);
    return mn[v];
}
```

Bell man ford

```
const long long INF = 2000000000000000000LL;
struct Edge {
    int u, v;
    long long w; // cạnh từ u đến v, trọng số w
};
void bellmanFord(int n, int S, vector<Edge> &e, vector<long long> &D, vector<int> &trace)
{
    // e: danh sách cạnh
    // n: số đỉnh
    // S: đỉnh bắt đầu
    // D: độ dài đường đi ngắn nhất
    // trace: mảng truy vết đường đi
    // INF nếu không có đường đi
    // -INF nếu có đường đi âm vô tận
    D.resize(n, INF);
    trace.resize(n, -1);

    D[S] = 0;
    for(int T = 1; T < n; T++) {
        for (auto E : e) {
            int u = E.u;
            int v = E.v;
            long long w = E.w;
            if (D[u] != INF && D[v] > D[u] + w) {
                D[v] = D[u] + w;
                trace[v] = u;
            }
        }
    }
}

vector<int> trace_path(vector<int> &trace, int S, int u) {
    if (u != S && trace[u] == -1) return vector<int>(0); // không có đường đi

    vector<int> path;
    while (u != -1) { // truy vết ngược từ u về S
        path.push_back(u);
        u = trace[u];
    }
    reverse(path.begin(), path.end()); // cần reverse vì đường đi lúc này là từ u về S

    return path;
}
```

Thuật toán Floyd-Warshall

```
void init_trace(vector<vector<int>> &trace) {
    int n = trace.size();
    for (int u = 0; u < n; u++) {
        for (int v = 0; v < n; v++) {
            trace[u][v] = u;
        }
    }
}

void floydWarshall(int n, vector<vector<long long>> &w,
vector<vector<long long>> &D, vector<vector<int>> &trace) {
    D = w;
    init_trace(trace); // nếu cần dò đường đi

    for (int k = 0; k < n; k++) {
        for (int u = 0; u < n; u++) {
            for (int v = 0; v < n; v++) {
                if (D[u][v] > D[u][k] + D[k][v]) {
                    D[u][v] = D[u][k] + D[k][v];
                    trace[u][v] = trace[k][v];
                }
            }
        }
    }
}

vector<int> trace_path(vector<vector<int>> &trace, int u, int v) {
    vector<int> path;
    while (v != u) { // truy vết ngược từ v về u
        path.push_back(v);
        v = trace[u][v];
    }
    path.push_back(u);

    reverse(path.begin(), path.end()); // cần reverse vì đường đi từ v ngược
về u

    return path;
}
```

7. DP

Dãy con có tổng lớn nhất

```
max_so_far = a[0];
curr_max = a[0];
for (int i = 1; i < n-1; i++)
{
        curr_max = max(a[i], curr_max+a[i]);
        max_so_far = max(max_so_far, curr_max);
}
```

LIS

```
long long n; cin >> n;
        long long a[n];
        for(int i=0; i<n; i++) cin >> a[i];
        long long check[n+1];
        // for(int i=0; i<n; i++)
        // {
        //      check[i] = INF;
        // }
        memset(check, 0x3f, sizeof(check));

        for(int i=0; i<n; i++)
        {
                int id = lower_bound(check, check+n, a[i]) -
check;
                check[id] = min(check[id], a[i]);
        }
        int sz = 0;
        for(int i=0; i<n; i++)
        {
                if(check[i] < INF) sz++;
                else break;
        }
        cout << sz << ed;
```

Tính tổng các mảng con

```
const int MOD = 1e9 + 7;

int SubArraySum(int arr[] , int n)
{
        int result = 0;

        for (int i=0; i<n; i++)
                result += (arr[i]%MOD * (i+1)%MOD * (n-
i)%MOD) %MOD;

        return result % MOD ;
}
```

Cái túi dởm

```
ll dp[1010][100010];

int main()
{
        Hello_i_am_Salmon
        ll n, x; cin >> n >> x;
        ll coin[n+1];
        ll book[n+1];
        for(int i=1; i<=n; i++) cin >> coin[i];
        for(int i=1; i<=n; i++) cin >> book[i];
        for(int i=1; i<=n; i++)
        {
                for(int j=1; j<=x; j++)
                {
                        if(j >= coin[i])
                                dp[i][j] = max(dp[i-
1][j], dp[i-1][j-coin[i]] + book[i]);
                        else dp[i][j] = dp[i-1][j];
                }
        }
        cout << dp[n][x] << ed;
```

Cái túi xịn knapsack

```
ll dp[1010][100010];

int main()
{
        Hello_i_am_Salmon
        int n, x;
        cin >> n >> x;
        int arr[n];
        int brr[n];
        for(int i=0; i<n; i++) cin >> arr[i];
        for(int i=0; i<n; i++) cin >> brr[i];
        vector<int> dp(x+2);

        for(int i=0; i<n; i++) {
                for(int j=x; j-arr[i]>=0; j--) {
                        dp[j] = max(dp[j], dp[j-
arr[i]]+brr[i]);
                }
        }
        cout<<dp[x]<<'\n';
}
```

Dp chữ số

/// How many numbers x are there in the range a to b, where the digit d occurs exactly k times in x?

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

vector<int> num;
int a, b, d, k;
int DP[12][12][2];
/// DP[p][c][f] = Number of valid numbers <= b from this state
/// p = current position from left side (zero based)
/// c = number of times we have placed the digit d so far
/// f = the number we are building has already become smaller than b? [0 = no, 1 = yes]

int call(int pos, int cnt, int f){
   if(cnt > k) return 0;

   if(pos == num.size()){
      if(cnt == k) return 1;
      return 0;
   }

   if(DP[pos][cnt][f] != -1) return DP[pos][cnt][f];
   int res = 0;

   int LMT;

   if(f == 0){
      /// Digits we placed so far matches with the prefix of b
      /// So if we place any digit > num[pos] in the current position, then the number will
become greater than b
      LMT = num[pos];
   } else {
      /// The number has already become smaller than b. We can place any digit now.
      LMT = 9;
   }

   /// Try to place all the valid digits such that the number doesn't exceed b
   for(int dgt = 0; dgt<=LMT; dgt++){
      int nf = f;
      int ncnt = cnt;
      if(f == 0 && dgt < LMT) nf = 1; /// The number is getting smaller at this position
      if(dgt == d) ncnt++;
      if(ncnt <= k) res += call(pos+1, ncnt, nf);
   }

   return DP[pos][cnt][f] = res;
}

int solve(int b){
   num.clear();
   while(b>0){
      num.push_back(b%10);
      b/=10;
   }
   reverse(num.begin(), num.end());
   /// Stored all the digits of b in num for simplicity

   memset(DP, -1, sizeof(DP));
   int res = call(0, 0, 0);
   return res;
}

int main () {

   cin >> a >> b >> d >> k;
   int res = solve(b) - solve(a-1);
   cout << res << endl;

                        return 0;
                    }
```

KMP

8. STRINGS

```
void KMP(string text,string pattern){
    int n = text.length(), m = pattern.length(), F[m+2],i,j;
    F[0] = F[1] = 0;
    for(int i = 2;i<=m;i++){
        j = F[i-1];
        while(true){
            if(pattern[j] == pattern[i-1]) { F[i] = j+1; break;}
            else if(j==0) {F[i] = 0; break;}
            else j = F[j];
        }
    }

    i = j = 0;
    while(j<n){
        if(text[j] == pattern[i]){
            i++; j++;
            if(i==m) printf("%d ",j-i+1);
        }
        else if(i>0) i = F[i];
        else j++;
    }
}

int main(){
    string a,b;
    cin>>a>>b;
    KMP(a,b);
   // getch();
}
```

Suffix automaton

```
struct state {
   int len, link;
   map<char, int> next;
};

const int MAXLEN = 100000;
state st[MAXLEN * 2];
int sz, last;

void sa_init() {
   st[0].len = 0;
   st[0].link = -1;
   sz++;
   last = 0;
}

void sa_extend(char c) {
   int cur = sz++;
   st[cur].len = st[last].len + 1;
   int p = last;
   while (p != -1 && !st[p].next.count(c)) {
      st[p].next[c] = cur;
      p = st[p].link;
   }
   if (p == -1) {
      st[cur].link = 0;
   } else {
      int q = st[p].next[c];
      if (st[p].len + 1 == st[q].len) {
         st[cur].link = q;
      } else {
      int clone = sz++;
      st[clone].len = st[p].len + 1;
      st[clone].next = st[q].next;
      st[clone].link = st[q].link;
                  while (p != -1 && st[p].next[c] == q) {
                     st[p].next[c] = clone;
                     p = st[p].link;
                  }
                  st[q].link = st[cur].link = clone;
               }
            }
            last = cur;
         }

struct state {
   ...
   bool is_clone;
   int first_pos;
   vector<int> inv_link;
};

// after constructing the automaton
for (int v = 1; v < sz; v++) {
   st[st[v].link].inv_link.push_back(v);
}

// output all positions of occurrences
void output_all_occurrences(int v, int P_length) {
   if (!st[v].is_clone)
      cout << st[v].first_pos - P_length + 1 << endl;
   for (int u : st[v].inv_link)
      output_all_occurrences(u, P_length);
}

string lcs (string S, string T) {
   sa_init();
   for (int i = 0; i < S.size(); i++)
      sa_extend(S[i]);

   int v = 0, l = 0, best = 0, bestpos = 0;
   for (int i = 0; i < T.size(); i++) {
      while (v && !st[v].next.count(T[i])) {
         v = st[v].link ;
         l = st[v].length ;
      }
      if (st[v].next.count(T[i])) {
         v = st [v].next[T[i]];
         l++;
      }
      if (l > best) {
         best = l;
         bestpos = i;
      }
   }
   return T.substr(bestpos - best + 1, best);
}
```

9. VARIOUS

Random

```
mt19937 gen(chrono::steady_clock::now().time_since_epoch().count());
```

Sinh tổ hợp

```
string sinhbit(string x)
{
  for(int i=sz(x); i>=0; i--)
  {
  if(x[i] == '0')
  {
  x[i] = '1';
  return x;
  }
  else x[i] = '0';
  }
  return "";
}

int main()
{
  Hello_i_am_Salmon
  int n; cin >> n;
  string a = "";
  for(int i=0; i<n; i++)
  {
  a += '0';
  }
  for(int i=0; i<pow(2, n); i++)
  {
  cout << a << ed;
  a = sinhbit(a);
  }
}
```

Bitmask

```
long long n; cin >> n;
  for(int mask = 0; mask < (1 << n); mask++)
  {
  for(int j=0; j<n; j++)
  {
  if(mask & (1 << j))
  {
  cout << "1";
  }
  else cout << "0";
  }
  cout << ed;
  }
```

Phương trình tuyến tính Linear Diophantine Equations

```
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;
}

bool find_any_solution(int a, int b, int c, int &x0, int &y0, int &g) {
    g = gcd(abs(a), abs(b), x0, y0);
    if (c % g) {
        return false;
    }

    x0 *= c / g;
    y0 *= c / g;
    if (a < 0) x0 = -x0;
    if (b < 0) y0 = -y0;
    return true;
}
```

Ngịch đảo modulo từ 1 -> m

```
inv[1] = 1;
for(int i = 2; i < m; ++i)
    inv[i] = m - (m/i) * inv[m%i] % m;
```

Tarjan tìm cầu

```
// A C++ program to find bridges in a given undirected graph
#include<iostream>
#include <list>
#define NIL -1
using namespace std;

// A class that represents an undirected graph
class Graph
{
  int V; // No. of vertices
  list<int> *adj; // A dynamic array of adjacency lists
  void bridgeUtil(int v, bool visited[], int disc[], int low[],
                                                    int parent[]);
public:
  Graph(int V); // Constructor
  void addEdge(int v, int w); // to add an edge to graph
  void bridge(); // prints all bridges
};

Graph::Graph(int V)
{
  this->V = V;
  adj = new list<int>[V];
}

void Graph::addEdge(int v, int w)
{
  adj[v].push_back(w);
  adj[w].push_back(v); // Note: the graph is undirected
}

// A recursive function that finds and prints bridges using
// DFS traversal
// u --> The vertex to be visited next
// visited[] --> keeps track of visited vertices
// disc[] --> Stores discovery times of visited vertices
// parent[] --> Stores parent vertices in DFS tree
void Graph::bridgeUtil(int u, bool visited[], int disc[],

                        int low[], int parent[])
{
  // A static variable is used for simplicity, we can
  // avoid use of static variable by passing a pointer.
  static int time = 0;

  // Mark the current node as visited
  visited[u] = true;

  // Initialize discovery time and low value
  disc[u] = low[u] = ++time;

  // Go through all vertices adjacent to this
  list<int>::iterator i;
  for (i = adj[u].begin(); i != adj[u].end(); ++i)
  {
                        int v = *i; // v is current adjacent of u

                        // If v is not visited yet, then recur for it
                        if (!visited[v])
                        {
                                parent[v] = u;
                                bridgeUtil(v, visited, disc, low, parent);

                                // Check if the subtree rooted with v has a
                                // connection to one of the ancestors of u
                                low[u] = min(low[u], low[v]);

                                // If the lowest vertex reachable from subtree
                                // under v is below u in DFS tree, then u-v
                                // is a bridge
                                if (low[v] > disc[u])
                                cout << u <<" " << v << endl;
                        }

                        // Update low value of u for parent function calls.
                        else if (v != parent[u])
                                low[u] = min(low[u], disc[v]);
  }
}

// DFS based function to find all bridges. It uses recursive
// function bridgeUtil()
void Graph::bridge()
{
  // Mark all the vertices as not visited
  bool *visited = new bool[V];
  int *disc = new int[V];
  int *low = new int[V];
  int *parent = new int[V];

  // Initialize parent and visited arrays
  for (int i = 0; i < V; i++)
  {
                parent[i] = NIL;
                visited[i] = false;
  }

  // Call the recursive helper function to find Bridges
  // in DFS tree rooted with vertex 'i'
  for (int i = 0; i < V; i++)
                if (visited[i] == false)
                        bridgeUtil(i, visited, disc, low, parent);
}

// Driver program to test above function
int main()
{
  // Create graphs given in above diagrams
  cout << "\nBridges in first graph \n";
  Graph g1(5);
  g1.addEdge(1, 0);
  g1.addEdge(0, 2);
  g1.addEdge(2, 1);
  g1.addEdge(0, 3);
  g1.addEdge(3, 4);
  g1.bridge();

  cout << "\nBridges in second graph \n";
  Graph g2(4);
  g2.addEdge(0, 1);
  g2.addEdge(1, 2);
  g2.addEdge(2, 3);
  g2.bridge();

  cout << "\nBridges in third graph \n";
  Graph g3(7);
  g3.addEdge(0, 1);
  g3.addEdge(1, 2);
```

```
    g3.addEdge(2, 0);
    g3.addEdge(1, 3);
    g3.addEdge(1, 4);
    g3.addEdge(1, 6);
    g3.addEdge(3, 5);
    g3.addEdge(4, 5);
    g3.bridge();

    return 0;
}
```

Tarjan tìm khung

```
// C++ program to find articulation points in an undirected graph
#include <bits/stdc++.h>
using namespace std;

// A recursive function that find articulation
// points using DFS traversal
// adj[] --> Adjacency List representation of the graph
// u --> The vertex to be visited next
// visited[] --> keeps track of visited vertices
// disc[] --> Stores discovery times of visited vertices
// low[] -- >> earliest visited vertex (the vertex with minimum
// discovery time) that can be reached from subtree
// rooted with current vertex
// parent --> Stores the parent vertex in DFS tree
// isAP[] --> Stores articulation points
void APUtil(vector<int> adj[], int u, bool visited[],
                        int disc[], int low[], int& time, int parent,
                        bool isAP[])
{
    // Count of children in DFS Tree
    int children = 0;

    // Mark the current node as visited
    visited[u] = true;

    // Initialize discovery time and low value
    disc[u] = low[u] = ++time;

    // Go through all vertices adjacent to this
    for (auto v : adj[u]) {
                // If v is not visited yet, then make it a child of u
                // in DFS tree and recur for it
                if (!visited[v]) {
                        children++;
                        APUtil(adj, v, visited, disc, low, time, u, isAP);

                        // Check if the subtree rooted with v has
                        // a connection to one of the ancestors of u
                        low[u] = min(low[u], low[v]);

                        // If u is not root and low value of one of
                        // its child is more than discovery value of u.
                        if (parent != -1 && low[v] >= disc[u])
                                isAP[u] = true;
                }

                // Update low value of u for parent function calls.
                else if (v != parent)
                        low[u] = min(low[u], disc[v]);
    }

    // If u is root of DFS tree and has two or more children.
    if (parent == -1 && children > 1)
                isAP[u] = true;
}

void AP(vector<int> adj[], int V)
{
    int disc[V] = { 0 };
    int low[V];
    bool visited[V] = { false };
    bool isAP[V] = { false };
    int time = 0, par = -1;

    // Adding this loop so that the
    // code works even if we are given
    // disconnected graph
    for (int u = 0; u < V; u++)
                if (!visited[u])
                        APUtil(adj, u, visited, disc, low,
                                time, par, isAP);

    // Printing the APs
    for (int u = 0; u < V; u++)
                if (isAP[u] == true)
                        cout << u << " ";
}

// Utility function to add an edge
void addEdge(vector<int> adj[], int u, int v)
{
    adj[u].push_back(v);
    adj[v].push_back(u);
}

int main()
{
    // Create graphs given in above diagrams
    cout << "Articulation points in first graph \n";
    int V = 5;
    vector<int> adj1[V];
    addEdge(adj1, 1, 0);
    addEdge(adj1, 0, 2);
    addEdge(adj1, 2, 1);
    addEdge(adj1, 0, 3);
    addEdge(adj1, 3, 4);
    AP(adj1, V);

    cout << "\nArticulation points in second graph \n";
    V = 4;
    vector<int> adj2[V];
    addEdge(adj2, 0, 1);
    addEdge(adj2, 1, 2);
    addEdge(adj2, 2, 3);
    AP(adj2, V);

    cout << "\nArticulation points in third graph \n";
    V = 7;
    vector<int> adj3[V];
    addEdge(adj3, 0, 1);
    addEdge(adj3, 1, 2);
    addEdge(adj3, 2, 0);
    addEdge(adj3, 1, 3);
    addEdge(adj3, 1, 4);
    addEdge(adj3, 1, 6);
    addEdge(adj3, 3, 5);
    addEdge(adj3, 4, 5);
    AP(adj3, V);

    return 0;
}
```

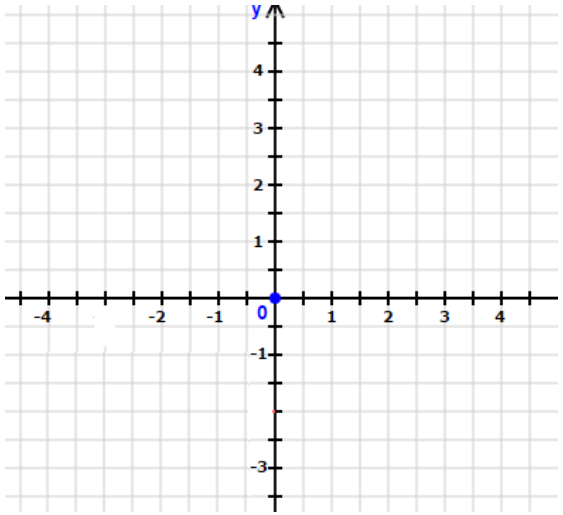
Dấu hiệu chia hết

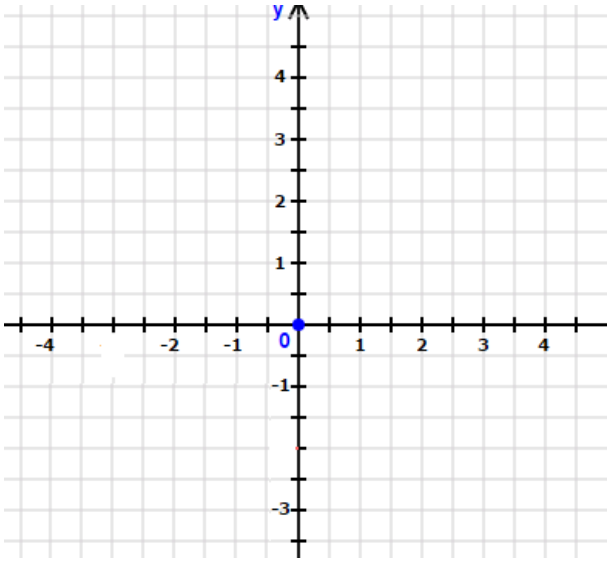
Ước số	Điều kiện chia hết	Ví dụ
1	Không cần điều kiện đặc biệt nào. Mọi số nguyên bất kì đều chia hết cho 1.	2: chia hết cho 1.
2	Chữ số tận cùng (hàng đơn vị) là chẵn (0, 2, 4, 6, hay 8).[2][3]	Số 1294: chữ số 4 chẵn nên chia hết cho 2.
3	Cộng các chữ số của số cần kiểm tra. Tổng phải chia hết cho 3.[2][4][5]	405 → 4 + 0 + 5 = 9 và 636 → 6 + 3 + 6 = 15, cả hai số đều chia hết cho 3. 16,499,205,854,376 → 1+6+4+9+9+2+0+5+8+5+4+3+7+6 tổng là 69 → 6 + 9 = 15 → 1 + 5 = 6, 6 rõ ràng chia hết cho 3.
	Lấy số lượng các chữ số 2, 5, và 8 có trong số cần kiểm tra trừ đi số các chữ số 1, 4, và 7 trong con số đó. Kết quả phải chia hết cho 3.	Sử dụng ví dụ trên: 16,499,205,854,376 có bốn chữ số nhóm 1, 4 và 7 và có bốn chữ số nhóm 2, 5 và 8; ∴ Bởi vì 4 − 4 = 0 là một bội của 3, số 16,499,205,854,376 chia hết cho 3.
4	Hai chữ số cuối cùng tạo thành một số chia hết cho 4.[2][3]	40,832: có 32 chia hết cho 4.
	Nếu chữ số hàng chục là chẵn, thì chữ số hàng đơn vị phải là 0, 4, hoặc 8. Nếu chữ số hàng chục là lẻ, chữ số hàng đơn vị phải là 2 hoặc 6.	40,832: chữ số 3 lẻ, còn chữ số hàng đơn vị là 2.
	Nhân đôi chữ số hàng chục, rồi cộng với chữ số hàng đơn vị được số chia hết cho 4.	40832: 2 × 3 + 2 = 8, chia hết cho 4.
5	Chữ số tận cùng là 0 hoặc 5.[2][3]	495: chữ số tận cùng là 5.
6	Số chia hết cho cả 2 và 3 thì chia hết cho 6	1458: có 1 + 4 + 5 + 8 = 18, nên nó chia hết cho 3 và chữ số tận cùng là chẵn, vì thế nó chia hết cho 6.
7	Lập một tổng xen kẽ đan dấu (tức tổng đại số có dấu cộng trừ xen kẽ nhau giữa các số hạng) của từng nhóm ba chữ số từ phải qua trái được kết quả là một bội số của 7.[5][6]	1,369,851: 851 − 369 + 1 = 483 = 7 × 69
	Lấy 5 nhân với chữ số tận cùng rồi cộng vào phần còn lại của số thu được một số chia hết cho 7. (Có hiệu lực bởi 49 chia hết cho 7, xem chứng minh ở dưới.)	483: có 48 + (3 × 5) = 63 = 7 × 9.
	Lấy 2 nhân với chữ số tận cùng rồi trừ vào lấy phần còn lại được một bội của 7. (Cách làm này có hiệu lực vì 21 chia hết 7.)	483: có 48 − (3 × 2) = 42 = 7 × 6.
	Lấy 9 nhân với chữ số tận cùng rồi trừ vào phần còn lại được kết quả chia hết cho 7.	483: có 48 − (3 × 9) = 21 = 7 × 3.
	Cộng 3 lần chữ số đầu vào chữ số tiếp theo của số đó rồi viết thêm vào kết quả chữ số còn lại thì phải được một bội số của 7. (Cách làm này có hiệu lực vì $10a + b - 7a = 3a + b$; số thu được đồng dư modulo 7 với $10a + b$.)	483: có 4×3 + 8 = 20, 203: có 2×3 + 0 = 6, 63: có 6×3 + 3 = 21.
	Cộng hai chữ số sau cùng vào hai lần phần còn lại thì được bội của 7. (Có hiệu lực vì 98 chia hết cho 7)	483,595: có 95 + (2 × 4835) = 9765: 65 + (2 × 97) = 259: 59 + (2 × 2) = 63.
	Nhân từng chữ số (từ phải qua trái) với từng số tương ứng (từ trái qua phải) trong dãy sau: 1, 3, 2, −1, −3, −2 (thực hiện lặp lại với các chữ số ở vị trí vượt quá hàng trăm nghìn). Tổng các tích trên là bội của 7.	483,595: có (4 × (−2)) + (8 × (−3)) + (3 × (−1)) + (5 × 2) + (9 × 3) + (5 × 1) = 7.
	Cộng chữ số tận cùng với 3 lần phần còn lại của số được một bội của 7.[7]	224: có 4 + (3 x 22) = 70
	Cộng thêm 3 lần chữ số tận cùng vào 2 lần phần còn lại được một bội của 7.[7]	245: có (3 x 5) + (2 x 24) = 7 x 9 = 63

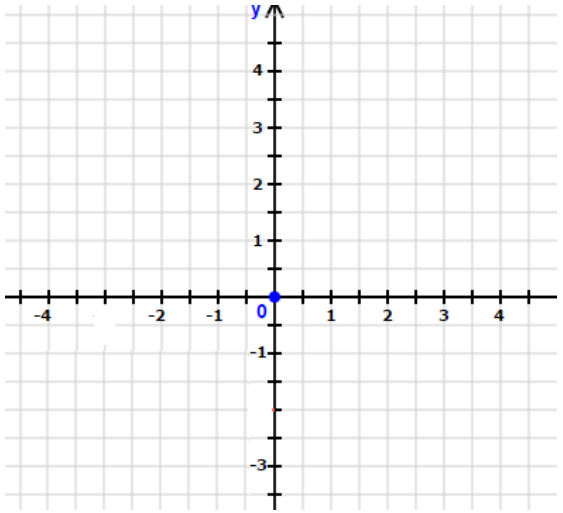
8	Nếu chữ số hàng trăm là chẵn, thì số tạo thành bởi hai chữ số sau cùng phải chia hết cho 8. Nếu chữ số hàng trăm là lẻ, thì số tạo thành bởi hai chữ số sau cùng cộng thêm 4 phải được số chia hết cho 8.	624: 24 chia hết cho 8. 352: 52 + 4 = 56 chia hết cho 8.
	Cộng chữ số sau cùng vào hai lần phần còn lại. Giá trị thu được phải là bội của 8.	56: có (5 × 2) + 6 = 16.
	Ba chữ số sau cùng tạo thành số chia hết cho 8.[2][3]	34,152: chỉ cần xét tính chia hết cho 152: =19 × 8
	Cộng 4 lần chữ số hàng trăm vào 2 lần chữ số hàng chục và 1 lần chữ số hàng đơn vị được kết quả phải là bội của 8.	34,152: 4 × 1 + 5 × 2 + 2 = 16
9	Tính tổng các chữ số, được kết quả chia hết cho 9.[2][4][5]	2880: có 2 + 8 + 8 + 0 = 18: có 1 + 8 = 9.
10	Chữ số hàng đơn vị là 0.[3]	130: chữ số hàng đơn vị là 0.
11	Lập tổng xen kẽ đan dấu giữa các chữ số, được kết quả phải chia hết cho 11.[2][5]	918,082: có 9 − 1 + 8 − 0 + 8 − 2 = 22 = 2 × 11.
	Cộng các nhóm gồm hai chữ số từ phải qua trái, kết quả phải chia hết cho 11.[2]	627: có 06 + 27 = 33 = 3 × 11.
	Trừ đi chữ số tận cùng vào phần còn lại của số, kết quả phải chia hết cho 11.	627: có 62 − 7 = 55 = 5 × 11.
	Cộng thêm chữ số tận cùng tới hàng trăm (hay thêm 10 lần chữ số hàng đơn vị vào phần còn lại). Kết quả phải chia hết cho 11.	627: có 62 + 70 = 132: có 13 + 20 = 33 = 3 × 11.
	Nếu số lượng các chữ số là chẵn thì cộng chữ số đầu và trừ chữ số cuối vào phần còn lại. Kết quả phải chia hết cho 11.	918,082: số chữ số là chẵn (6) → 1808 + 9 − 2 = 1815: có 81 + 1 − 5 = 77 = 7 × 11
	Nếu số lượng chữ số là lẻ thì trừ cả chữ số đầu và chữ số cuối vào phần còn lại. Kết quả phải chia hết cho 11.	14,179: số chữ số là lẻ (5) → 417 − 1 − 9 = 407 = 37 × 11
12	Số đó chia hết cho cả 3 và 4.	324: chia hết cho cả 3 và 4
	Trừ chữ số sau cùng vào hai lần phần còn lại. Kết quả phải chia hết cho 12.	324: 32 × 2 − 4 = 60 = 5 × 12.
13	Lập tổng xen kẽ từng nhóm ba chữ số từ phải qua trái. Kết quả phải chia hết cho 13.[6]	2,911,272: 272 − 911 + 2 = −637
	Cộng thêm 4 lần chữ số hàng đơn vị vào phần còn lại, kết quả phải chia hết cho 13.	637: 63 + 7 × 4 = 91, 9 + 1 × 4 = 13.
	Trừ đi số gồm hai chữ số cuối vào bốn lần phần còn lại, được kết quả chia hết cho 13.	923: 9 × 4 − 23 = 13.
	Trừ đi 9 lần chữ số tận cùng vào phần còn lại, được kết quả chia hết cho 13.	637: 63 − 7 × 9 = 0.
14	Số đó chia hết cho cả 2 và 7.	224: sử dụng tính chất chia hết cho 2 và 7 ta thấy nó đều chia hết.
	Cộng hai chữ số cuối vào hai lần phần còn lại, được kết quả chia hết cho 14.	364: 3 × 2 + 64 = 70. 1764: 17 × 2 + 64 = 98.
15	Số đó chia hết cho cả ba và 5.[8]	390: nó chia hết cho cả ba và 5.
16	Nếu chữ số hàng nghìn là chẵn thì số tạo thành bởi ba chữ số cuối phải chia hết cho 16. Nếu chữ số hàng nghìn là lẻ, thì số tạo thành bởi ba chữ số cuối phải chia hết cho 16.	254,176: 176 = 16 × 11. 3408: 408 + 8 = 416 = 26 × 16.
	Cộng hai chữ số cuối vào 4 lần phần còn lại, kết quả phải chia hết cho 16.	176: 1 × 4 + 76 = 80. 1168: 11 × 4 + 68 = 112.
	Bốn chữ số tận cùng phải chia hết cho 16.[2][3]	157,648: 7,648 = 478 × 16.
17	Trừ 5 lần chữ số tận cùng vào phần còn lại, được số chia hết cho 17.	221: 22 − 1 × 5 = 17.
	Trừ hai chữ số tận cùng vào hai lần phần còn lại.	4,675: 46 × 2 − 75 = 17.
	Cộng 9 lần chữ số tận cùng vào 5 lần phần còn lại, bỏ đi chữ số 0 tận cùng của kết quả nếu có rồi lặp lại.	4,675: 467 × 5 + 5 × 9 = 2380; 238: 23 × 5 + 8 × 9 = 187.

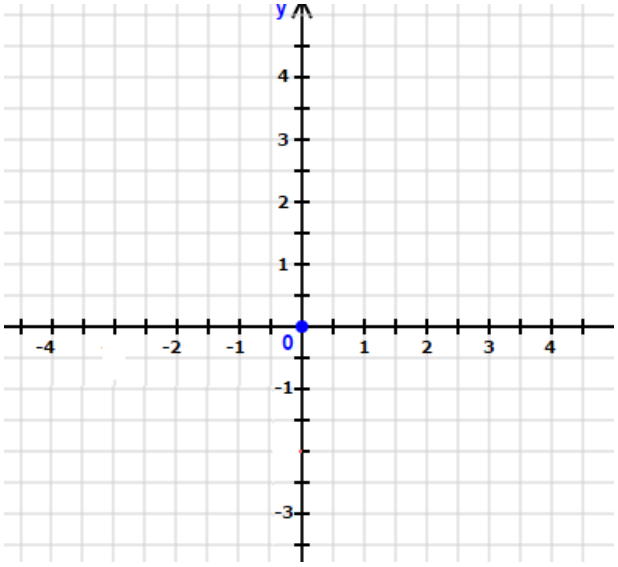
--- End template IUH.Ocen---

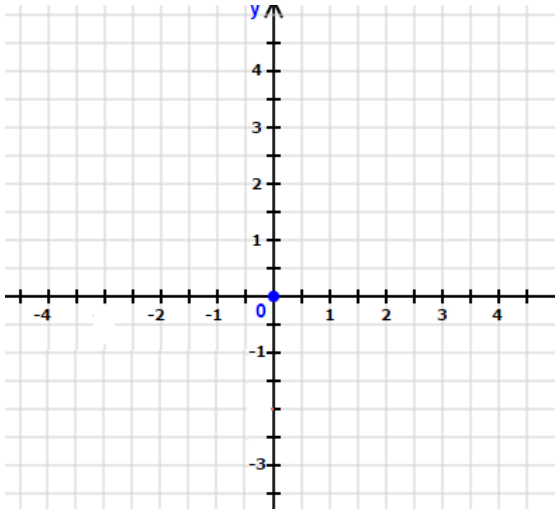
Tọa độ để nháp

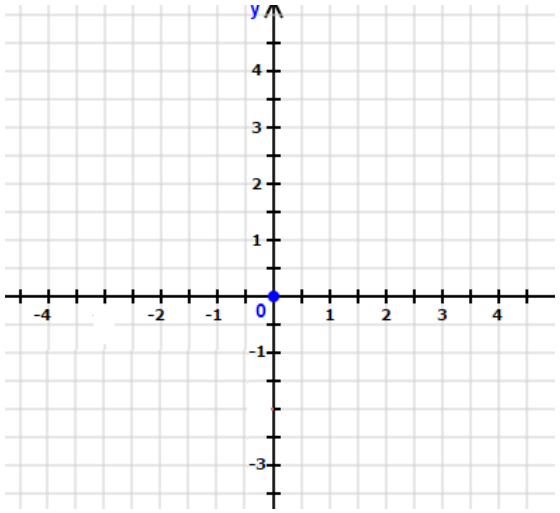


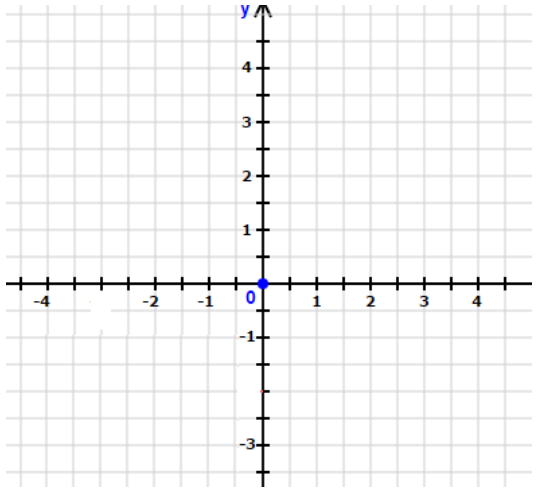


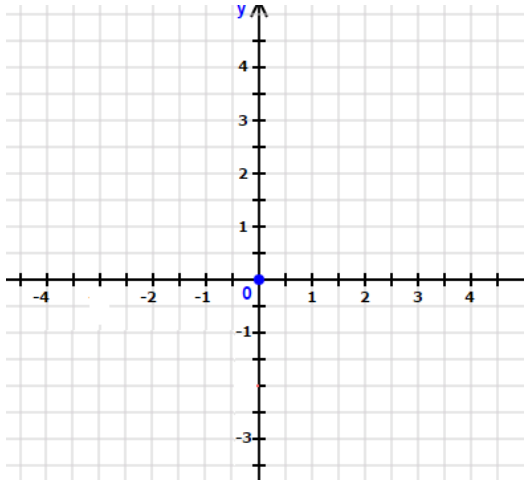












Hình may mắn

IUH.Ocean

Không bao giờ cúi đầu, không bao giờ bỏ cuộc hay chỉ ngồi than thở. Hãy tìm một cách khác.

Sẽ không bao giờ có bế tắc thật sự khi trong bạn còn niềm tin. Chỉ cần có niềm tin, bạn sẽ có hi vọng, sẽ tìm thấy con đường để bước tiếp.

Try hard đến giây cuối cùng của kì thi…
Vì mọi người, vì teamate…