# Lý thuyết số

## Số nguyên tố

### Sàng số nguyên tố

|  |
| --- |
| //SangNT  #include<bits/stdc++.h>  using namespace std;  vector<int> sang\_nt(){  vector<int>v(1000001, 1);  v[0] = 0;  v[1] = 0;  for(int i = 2; i <= sqrt(1000000); i++){  if(v[i] == 1){  for(int j = i\*i; j <= 1000000; j+=i){  v[j] = 0;  }  }  }    return v;  }  int main(){  vector <int>v = sang\_nt();  for(int i = 1; i <= 100; i++){  if(v[i] == 1){  cout << i << " ";  }  }  } |

### Sàng số nguyên tố trên đoạn

|  |
| --- |
| //  #include <bits/stdc++.h>  using namespace std;  vector <int> sang\_doan(int l, int r){  vector <int> v(r-l+1, 1);  for(int i = 2; i <= sqrt(r); i++){  {  for(int j = max(i\*i, (l+i-1)/i\*i); j<= r;j+=i){  v[j-l] = 0;  }  }  }  return v;  }  int main(){  int l, r;  cin >> l >> r;  vector <int> v = sang\_doan(l, r);  for(int i = max(2, l); i <= r; i++){  if(v[i-l] == 1){  cout << i << " ";  }  }  } |

* Ý tưởng: lưu trữ các giá trị kiểm tra nguyên tố vào 1 mảng có (l-r+1) phần tử
* Duyệt qua các bội của I (từ l đến sqrt(r)) và đánh dấu nó bằng 0

### Ước số nguyên tố nhỏ nhất

|  |
| --- |
| //sàng biến đổi  vector<int> sang\_bd(){  vector<int>v(100001);  for(int i = 0; i <= 100000; i++)  v[i] = i;    for(int i = 2; i <= sqrt(100000); i++){  if(v[i] == i){  for(int j = i\*i; j <= 100000; j+=i){  if(v[j] == j){  v[j] = i;  }  }  }  }    return v;  } |

### N! chia hết cho px

***Ý tưởng***

* Đếm xem từ 1 đến n có bao nhiêu số chia hết cho p. Chú ý trường hợp 1 số nào tích của nhiều số p

|  |
| --- |
| //Cách 1  int dem(int n, int p){  int cnt = 0;  for(int i = 1; i <= n; i++){  if(i % p == 0){  int tmp = i;  while(tmp % p == 0){  cnt++;  tmp /= p;  }  }  }  return cnt;  } |

***Cách 2(Legendre)***

Sử dụng công thức

|  |
| --- |
| (vòng lặp khi mẫu số nhỏ hơn tử số) |

|  |
| --- |
| //  int dem(int n, int p){  int cnt = 0;  for(int i = p; i <= n; i\*=p)  cnt += n/i;  return cnt;  } |

### Số chữ số 0

Đếm số lượng chữ số 0 của n!

***Ý tưởng***

* Số 0 được tạo thánh từ tích với số 10. 10 được phân tích = 2x5. Mà số lần xuất hiện của 5 ít hơn 2 => Cần tìm số lần mà 5 được phân tích trong n!
* Giống bài trên

### Số hoàn hảo

Nếu p là số nguyên tố và cũng là số nguyên tố thì (-1) là số hoàn hảo

|  |
| --- |
| //  int ktra\_nt(int n){  if(n < 2)  return 0;  for(int i = 2; i <= sqrt(n); i++)  if(n % i == 0)  return 0;  return 1;  }  vector <long long> init\_hh(){  vector <long long>v;  for(int i = 2; i <= 32; i++){  if(ktra\_nt(i) && ktra\_nt(pow(2, i)-1))  v.push\_back( pow(2, i-1)\*(pow(2, i)-1) );  }  return v;  } |

### Phi hàm Ơ-le

* Đếm số lượng số nguyên tố cùng nhau với n

|  |
| --- |
| //phi bằng tích n x (1-1/p)  // với p là các số nguyên tố và n chia hết cho p  //n(1-1/p) có thể biến đổi = n – n/p để tránh sai số |

|  |
| --- |
| //Phi ham Ole  long long phi(long long n){  long long res = n;  for(int i = 2; i <= sqrt(n); i++){  if(n % i == 0){  res = res - res/i;  while(n % i == 0)  n /= i;  }  }  if(n != 1){  res = res - res/n;  }  return res;  } |

|  |
| --- |
| //Sang phi ham ole  vector<int> sang\_ole(){  vector<int>v(1000001);  for(int i = 1; i <= 1000000; i++)  v[i] = i;    //do so nguyen to co phi ham ole = chinh no - 1  for(int i = 2; i <= 1000000; i++){  if(v[i] == i){  v[i] -=1;  //duyet cac boi cua i  for(int j = i \* 2; j <= 1000000; j+=i)  v[j] = v[j] - v[j]/i;  }  }    return v;  } |

### Số ước của n!

|  |
| --- |
| Giả sử n! có thể chia hết cho tối đa px. Giả sử có nhiều hơn 1 p. Ta có thể viết như sau  Khi đó số lượng ước của n! = ) ) )… )  với p là các số nguyên tố |

|  |
| --- |
| //  //tim bac cua p trong n!  int tim\_bac(int p, int n){  int res = 0;  for(int i = p; i <= n; i\*=p)  res += n/i;  return res;  }  int ktra\_nt(int n){  if(n < 2)  return 0;  for(int i = 2; i <= sqrt(n); i++)  if(n % i == 0)  return 0;  return 1;  }  long long dem\_uoc(int n){  long long res = 1;  for(int i = 2; i <= n; i++){  if(ktra\_nt(i)){  res \*= (tim\_bac(i, n) + 1);  }  }  return res;  } |

### lcm(sum)

Cho số n. Yêu cầu tính tổng lcm(1, n) + lcm(2, n) + lcm(3, n) + … + lcm(n, n)

Với lcm(a, b) là bcnn(a, b)

|  |
| --- |
| Sử dụng công thức  Với k là ước của n  Ví dụ với lcm(i, 6). Với i = 1, 2…6 sẽ bằng )\*6/2 |

|  |
| --- |
| //  vector <int> phi\_ham\_ole(){  vector<int> v(1000001);  for(int i = 1; i <= 1000000; i++)  v[i] = i;    for(int i = 2; i <= 1000000; i++){  if(v[i] == i){  v[i]--;  for(int j = i \* 2; j <= 1000000; j+=i){  v[j] = v[j] - v[j]/i;  }  }  }  return v;  }  vector<long long>sang\_tong(){  vector <int> ol = phi\_ham\_ole();  vector <long long> v(1000001, 0);  for(int i = 1; i<= 1000000; i++){  for(int j = i ; j <= 1000000; j+=i){  v[j] += i \* ol[i];  }  }  return v;  } |

## Đồng dư

1 số công thức cần ghi nhớ

(a+b)%c = [(a%c)+(b%c)]%c

(a-b)%c = [(a%c)-(b%c)]%c

(a\*b)%c = [(a%c)\*(b%c)]%c

### Tính tổng modulo

Cho 2 số nguyên n và k. Hãy tính tổng S = n%1 + n%2 + n%3 +… +n%k

|  |
| --- |
| Các số hạng của S sẽ tuần hoàn như sau  1 2 3 … 0 1 2 3 …0 1,2 3 n%k  Các số hạng 0 tại các vị trí là bội của k. Cụm cuối cùng có thể không phủ hết 1 lần k, nên tổng quát là n%k  Các cụm đầy đủ sẽ là dãy tự nhiên từ 1 đến k – 1   * Công thức tổng quát |

### Số nguyên dương thứ k không chia hết cho n

Tìm số nguyên dương thứ k không chia hết cho n

|  |
| --- |
| Giả sử có n = 5, k = 8  1 2 3 4 [5] 6 7 8 9 [10]  Nhận thấy cứ n số thì có n-1 số không chia hết cho n  lấy có thể biết được số cụm. nếu có phần dư, thì số cần tìm sẽ ở cụm tiếp theo. Nếu không, số cần tìm sẽ ở cụm hiện tại. r = k%(n-1)  Với phần dư (r == 0, số cụm = x) => số cần tìm = n \* x – 1;  Với phần dư (r != 0, số cụm = x) => số cần tìm = n \* x + r; |

### Chữ số tận cùng 1378^n

|  |
| --- |
| Nhận thấy số tận cùng của 2^n sẽ tuần hoàn theo chu kì  2^1 = 2  2^2 = 4  2^3 = 8  2^4 = 16  Số tận cùng lặp theo chu kỳ 2/4/8/6  -  1378^n, số tận cùng phụ thuộc vào 8^n  8^1 = 8  8^2 = 4  8^3 = 2  8^4 = 6 |

### (1^n+2^n+3^n+4^n+5^n)mod 5

Nhận thấy 1^n, 2^n, 3^n, 4^n đều là những số có chữ số cuối tuần hoàn

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 |
| 1^n | 1 | 1 | 1 | 1 |
| 2^n | 2 | 4 | 8 | 6 |
| 3^n | 3 | 9 | 7 | 1 |
| 4^n | 4 | 6 | 4 | 6 |
| Sum | 0 | 0 | 0 | 4 |

Do đó, (1^n+2^n+3^n+4^n) mod 5 dư 4 nếu n chia hết cho 4. Và (1^n+2^n+3^n+4^n) mod 5 = 0, trong các trường hợp còn lại

### Last digit a^n

Tìm chữ số tận cùng của a^n, với a và n rất lớn (1<a.n<1010\_000\_000)

Quy luật của các chữ số cuối

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 |
| 0^n | 0 | 0 | 0 | 0 |
| 1^n | 1 | 1 | 1 | 1 |
| 2^n | 2 | 4 | 8 | 6 |
| 3^n | 3 | 9 | 7 | 1 |
| 4^n | 4 | 6 | 4 | 6 |
| 5^n | 5 | 5 | 5 | 5 |
| 6^n | 6 | 6 | 6 | 6 |
| 7^n | 7 | 9 | 3 | 1 |
| 8^n | 8 | 4 | 2 | 6 |
| 9^n | 9 | 1 | 9 | 1 |

|  |
| --- |
| //  int tim\_so(string a, string n){  int x = a[a.size()-1] - '0';  if(x == 0||x == 1||x == 5||x == 6)  return x;  if(n.size() == 1)  n = "0" + n;  int y = (n[n.size()-2] - '0') \* 10;  y += n[n.size()-1] - '0';    int r2[4] = {6, 2, 4, 8};  int r3[4] = {1, 3, 9, 7};  int r4[4] = {6, 4, 6, 4};  int r7[4] = {1, 7, 9, 3};  int r8[4] = {6, 8, 4, 2};  int r9[4] = {1, 9, 1, 9};    y %= 4;  if(x == 2)  return r2[y];  if(x == 3)  return r3[y];  if(x == 4)  return r4[y];  if(x == 7)  return r7[y];  if(x == 8)  return r8[y];  if(x == 9)  return r9[y];  } |

## No Title

#### Đếm số chia hết

Một số nguyên không âm được gọi là đẹp nếu như chia hết cho một trong ba số sau: 4, 7 và 11. Hãy đếm số lượng số đẹp thỏa mãn L <= x <= R

***Gợi ý***

1. Đếm số lượng số chia hết cho 1 số nguyên x. Sử dụng công thức

* Công thức tính cận dưới (l)
* Công thức tính cận trên (r)
  + Trong đó, cận dưới là số nhỏ nhất lớn hơn hoặc bằng l chia hết cho x
  + Trong đó, cận trên là số lớn nhất nhỏ hơn hoặc bằng r chia hết cho x

2. Vì 4, 7, 11 là 3 số nguyên tố cùng nhau. Gọi số lượng số chia hết cho 1 số x là N(x). Công thức cuối cùng như sau

N = N(4) + N(7) + N(11) + N(4\*7\*11)-N(4\*7)-N(4\*11)-N(7\*11)

* Do, số lượng bội của 4, 7, 11 khi đếm sẽ có những số là bội chung nên phải trừ đi số lượng trùng này

#### Giải hệ phương trình bậc nhất

Viết chương trình tìm nghiệm tổng quát của hệ

***Gợi ý***

* Sử dụng công thức nghiệm với phương pháp Cramer. Lưu ý, phương pháp Cramer dùng để lấy nghiệm của hệ có số ẩn bằng số phương trình, chỉ áp dụng khi hệ có 1 nghiệm duy nhất.
* Giả sử với hệ tổng quát

Có thể viết hệ dưới dạng tổng quát

Ax=b

* Giả sử A có định thức khác 0. Công thức tính nghiệm của hệ

***Xét bài toán bậc 2***

* Biện luận nghiệm như sau

***Xét bài toán ban đầu***

Det(A) = ae-bd

Det(x) = ce-bf

Det(y) = af-dc

Nghiệm duy nhất

Điều kiện: Det(A)!=0

x = Det(x)/Det(A)

y = Det(y)/Det(A)

Vô nghiệm

Det(A) = 0

Vô số nghiệm

|  |
| --- |
| #include <bits/stdc++.h>  using namespace std;  int main(){  long long a, b, c;  long long d, e, f;  cin >> a >> b >> c >> d >> e >> f;  double delta = a\*e-b\*d;  double delta\_x = c\*e-b\*f;  double delta\_y = a\*f-c\*d;  if(delta == 0){  if( delta\_x == 0 && delta\_y == 0){  cout <<"VOSONGHIEM";  }  else  cout <<"VONGHIEM";  }  else{  double n1 = delta\_x/delta;  double n2 = delta\_y/delta;      if(abs(n1-0) <= 0.00)  n1 = 0;  if(abs(n2-0) <= 0.00)  n2 = 0;  printf("%.2lf %.2lf", n1, n2);  }  } |

#### Căn bậc 2 lồng nhau

Cho số nguyên dương n, tính

***Gợi ý***

* Có thể tính từ cuối về

|  |
| --- |
| //  #include <bits/stdc++.h>  using namespace std;  int main() {  int n;  cin >> n;    double res = 0;  for(int i = n; i >= 1; i--){  res = sqrt(i + res);  }    cout << res;  } |

## Nhị phân

### Lũy thừa

* Ý tưởng

Lũy thừa an có thể được tính nhanh với công thức như sau

* Cài dặt

|  |
| --- |
| //Vòng lặp  long long tinh\_luy\_thua(int a, int b){  int res = 1;  while(b != 0){  if(b%2 == 1){  res \*= a;  }  b/=2;  a\*=a;  }  return res;  } |

|  |
| --- |
| //Đệ quy  long long tinh\_luy\_thua(int a, int b){  if(b == 0)  return 1;  else{  long long x = tinh\_luy\_thua(a, b/2);  if(b % 2 == 1)  return x \* x \*a;  else  return x \* x;  }  } |

# Cấu trúc dữ liệu

## Vector

### Lý thuyết

* Vector là một cấu trúc dữ liệu không cần khai báo trước số phần tử (mảng động)
* Vector là một lớp với các phương thức
* Một số phương thức
* Cú pháp khai báo

|  |
| --- |
| vector<int>*v*(*size*, *value*);  //v là tên vector  //size là kích thước  // value là giá trị cho tất cả |

### push\_back()

* thêm một phần tử vào cuối vector

### size()

* Lấy ra kích thước

## String

### Nhập/xuất

* Lệnh nhập
  + Nếu trước khi nhập string, có phím enter, cần xử lý trôi lệnh

|  |
| --- |
| cin.ignore()  //mặc định xóa 1 ký tự trước đó |

* + getline(cin, s); //lệnh nhập

### stringstream

* Dùng để tách chuỗi. Mặc định tách tại các khoảng trắng

|  |
| --- |
| //  #include <iostream>  #include <string>  #include<sstream>  using namespace std;  int main(){  string s;  getline(cin , s);  stringstream ss(s);  string tmp;  while(ss >> tmp){  cout << tmp << " ";  }  } |

## Pair

### Khai báo

|  |
| --- |
| pair<kdl\_1, kdl\_2>  //pair lưu được 1 cặp dữ liệu ứng với 2 thuộc tính: first, second |

# Mảng

## Dãy con tăng

### O(n)^2

Đề bài: Cho một dãy số, đếm số lượng phần tử của dãy con tăng dài nhất

Giả sử có dãy số

[1, 2, 3, 1, 4, 8, 5]

* Dãy con tăng có thể là [1, 2, 3, 4, 5] hoặc [1, 2, 3, 4, 8]

***Ý tưởng***

* Giả sử ban đầu coi mỗi phần tử của dãy là một dãy con
* Ta tạo một mảng L[n] với L[i] là số lượng phần tử trong dãy con tăng của vị trí i
* Sau đó, tìm cách thêm phần tử này vào các dãy con phía trước nó. nếu phần tử a[i] (ở vị trí đang xét) lớn hơn phần tử a[j] nào (đứng trước a[i]) thì khi này L[i] sẽ được cập nhật bằng công thức max(l[j] + 1, L[i])
* Xét dãy [1, 2, 3, 1, 4, 8, 5]

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Bước |  | 1 | 2 | 3 | 1 | 4 | 8 | 5 |
| 1 | L[i] | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 2 |  | 1 | 2 | 3 | 1 | 4 | 5 | 5 |

* Tại bước 1, khởi tạo L[n]= 1
* Tại bước 2
  + Xét phấn tử 1, không có phần tử nào trước nó
  + Xét phần tử 2, 2>1 (1 phía trước), và 1 có L[0] = 1. Cập nhật L[1] (của phần tử 2) = max(L[0]+1, L[1]) = 2
  + Xét phần tử 3, (3>1) khi này L[2] = max(L[0]+1, L[2]) = max(2, 1) = 2.

Tiếp túc thấy 3>1, khi này L[2] = max(L[1]+1, L[2]) = max(3, 2) = 3

…

* Số phần tử của dãy con tăng dài nhất, là max của mảng L[n]

***Code demo***

|  |
| --- |
| #include <bits/stdc++.h>  #include <vector>  using namespace std;  int main(){  int n;  cin >> n;  vector <int>a(n);  for(int i = 0; i < n; i++){  cin >> a[i];  }    vector <int>L(n, 1);  for(int i = 0; i < n; i++){  for(int j = 0; j < i; j++){  if(a[i] > a[j]){  L[i] = max(L[i], L[j]+1);  }  }  }  cout <<\*max\_element(L.begin(), L.end());  } |