ÔN TẬP

1. Số học

* Kiến thức số học: ước số, bội số, số chính phương, số nguyên tố, UCLN, BCNN, …
* Các thuật toán cơ bản

Kiểm tra số chính phương

bool checkCP(n)

{

if (n < 0) return false;

m = sqrt(n);

return (m\*m == n);

}

Kiểm tra số nguyên tố

bool checkNT(n)

{

if (n < 2) return false;

m = sqrt(n);

for (i = 2; i <= m; ++i)

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

return true;

}

* Sử dụng khi cần kiểm tra 1 hoặc và số có phải là số nguyên tố.
* Không sử dụng để kiểm tra các phần tử của một dãy số có là số nguyên tố

Tìm UCLN

//điều kiện: a, b >= 0

GCD(a, b)

{

while (b > 0)

{

c = a % b;

a = b;

b = c;

}

return a;

}

Tìm BCNN

LCM(a, b)

{

d = GCD(a, b);

return a\*b/d;

}

Thuật toán tạo sàng nguyên tố

//d[x] = false thì x là số nguyên tố, ngược lại x là hợp số

void Sieve()

{

d[0] = d[1] = true;

m = sqrt(maxN);

for (x = 2; x <= m; ++x)

if (d[x] == false) //x là số nguyên tố

for (y = x; x\*y <= maxN; ++y)

d[y\*x] = true; //y\*x là hợp số

}

Lưu ý:

* Lọc các số nguyên tố từ sàng ra một mảng p[]

np = 0;

for (x = 2; x <= maxN; ++x)

if (d[x] == false)

p[np++] = x;

Phân tích số nguyên dương x thành tích các thừa số nguyên tố

void Parse(x)

{

a = 2; m = sqrt(x);

while (x > 1 && a <= m)

{

if (x % a == 0)

{

cout<<a<<" ";

x = x/a;

}

else ++a;

}

if (x > 1) cout<<x;

}

Phân tích thành tích các thừa số nguyên tố của số

* Sử dụng mảng mu[] để lưu trữ lại số mũ của các thừa số nguyên tố, mu[p] là số mũ của thừa số nguyên tố p trong phân tích của số n nào đó. Ví dụ: P = 12 = 2\*2\*3 => mu[2] = 2, mu[3] = 1, mu[i] = 0 với mọi i khác 2 và 3.
* Phân tích từng phần tử a[i] thành tích các thừa số nguyên tố và cập nhật số mũ. P = 12x8 => mu[2] = 5, mu[3] = 1, còn lại là 0.
* Gọi d[x] là một thừa số nguyên tố của x, nếu x là số nguyên tố thì quy ước d[x] = 0. Ví dụ d[12] = 2 hoặc d[12] = 3, d[87] = 3 hoặc d[87] = 29

Phân tích thừa số nguyên tố nhanh: phân tích một dãy các phần tử thành tích các thừa số nguyên tố

void ModifiedSieve()

{

m = sqrt(maxN);

for (x = 2; x <= m; ++x)

if (d[x] == 0) //x là số nguyên tố

for (y = x; x\*y <= maxN; ++y)

d[y\*x] = x;

}

void Parse(x)

{

while (x > 1)

{

p = d[x];

if (p == 0) p = x;

while (x % p == 0)

{

++mu[p];

x = x/p;

}

}

}

Cho dãy số . Cho biết số ước của từng phần tử của dãy số.

* Gọi d[x] là số ước của x

void ModifiedSieve()

{

m = sqrt(maxN);

for (x = 1; x <= m; ++x)

{

++d[x\*x];

for (y = x+1; x\*y <= maxN; ++y)

d[x\*y] = d[x\*y] + 2;

}

}

1. Sắp xếp và tìm kiếm

* Xem lại cách sử dụng hàm sort trong thư viện <algorithm>. Xem lại cách sắp xếp tăng dần, giảm dần và sắp xếp trên nhiều khóa.
* Xem lại thuật toán tìm kiếm nhị phân

int BinSearch(x, lo, hi)

{

while (lo <= hi)

{

m = (lo + hi)/2;

if (a[m] == x) return m;

if (a[m] < x) lo = m+1;

else hi = m-1;

}

return -1;

}

Chặt nhị phân

* Cho hàm số f(x) là đồng biến hoặc nghịch trên miền giá trị của x. Tìm x lớn nhất hoặc nhỏ nhất để f(x) thỏa một điều kiện nào đó.
* Thuật toán tìm x lớn nhất/nhỏ nhất để f(x) thỏa điều kiện nào đó

void BSMax(lo, hi)

{

ans = -1; //dấu hiệu cho biết không có nghiệm

while (lo <= hi)

{

x = (lo + hi)/2;

if (f(x) == true)

TH1 (tìm x lớn nhất): ans = x, lo = x+1;

TH2 (tìm x nhỏ nhất): ans = x, hi = x-1

else

TH1 (tìm x lớn nhất): hi = x-1;

TH2 (tìm x nhỏ nhất): lo = x+1;

}

cout<<ans;

}