Họ tên: Nguyễn Anh Tuấn - 17000909

Lớp: K62A3 - Máy tính & KHTT

Môn: Thiết kế & đánh giá thuật toán

**Bài Tập 1**

1. **Máy Turning**

**2. Xây dựng máy Turing M3 thực hiện việc thay tất cả các số 0 trong một dãy nhị**

**phân thành các số 1 và ngược lại.**

Ví dụ: 01001 -> 10110.

|  |  |
| --- | --- |
| Step | Fune |
| 1 | 3 |
| 2 | 1 |
| 3 | 2 |
| 4 | 1 |
| 5 | 1 |
| 6 | 2 |
| 7 | 4 |
| 8 | 6 |
| 9 | 5 |
| 10 | 5 |
| 11 | 6 |
| 12 | 5 |

**1. Xây dựng máy Turing M2 thực hiện phép trừ 1 của số nhị phân.**

3. / , ->

1. 0/0,-> 6. 1/0, <-

2.1/1,-> 5.0/1,->

7. / ,->

1. **Hàm đệ quy nguyên thủy**
2. **Multiplication: a×b**

mul (a, b) = a.b

mul (a, 0) = a.0 = 0

mul (a, b+1) = g (b, mul(a, b), a)

Có g (a1, a2, a3) = a1, a2, a3) + a1, a2, a3) = a2 + a3

* g (b, mul(a, b), a) = mul (a, b) + a = a.(b + 1) = mul (a, b +1)

Vậy mul(a, b) là hàm ĐQNT

1. **Exponentiation: ab**

pow (a, 0) = 1

pow (a, b+1) = g (b, pow(a, b), a)

Có g(a1, a2, a3) = a1, a2, a3) . a1, a2, a3) = a2 . a3

* g (b, pow(a, b), a) = pow(a, b) . a = pow (a, b+1)

Vậy **ab** là hàm ĐQNT

1. **Factorial a!** : 0! = 1, a'! = a!×a'

fact (0) = 1

fact (a+1) = g (a, fact(a))

Có g(a1, a2) = S ((a1, a2) . (a1, a2))

* g (a, fact(a)) = S(a). fact (a)

Vậy a! là hàm ĐQNT.

**III. Phân tích bài toán, viết thuật toán, xây dựng chương trình**

1. **Tìm ước chung lớn nhất của 2 số a và b bằng phương pháp Euclit.**

***Phân tích bài toán:***

Cho a, b là hai số nguyên (giả sử a ≥ b), để tìm ước chung lớn nhất của hai số a và b ta cần thực hiện chia a cho b được thương q và số dư r (r ≥ 0) tức là a = b\*q + r, khi đó ta có:

UCLN(a,b) = b nếu r =0

UCLN(b,r) nếu r khác 0

***Thuật toán:***

Input: a, b

Output: UCLN(a,b)

//Nhap a, b//

While (a!=b) {

if ( a>b)

a = a-b

else

b = b-a

}

//Xuat a//

***Xây dựng chương trình***

import java.util.Scanner;

public class ucln {

public static void main(String []args){

Scanner sc = new Scanner(System.*in*);

System.*out*.print("Nhap a = ");

int a = sc.nextInt();

System.*out*.print("Nhap b = ");

int b = sc.nextInt();

System.*out*.println("UCLN cua " + a + " va " + b + " la: " + *UCLN*(a, b));

}

public static int UCLN(int a, int b) {

while (a != b) {

if (a > b)

a = a - b;

else

b = b - a;

}

return a;

}

}

**2. Liệt kê các số nguyên tố nhỏ hơn n bằng phương pháp sàng Eratosthenes.**

***Thuật toán sàng nguyên tố Eratosthenes***

- B1. Tạo mảng đánh dấu cho tất cả các phần tử từ 2 đến N và mặc định tất cả đều là số nguyên tố

- B2. Xét số đầu tiên tìm được là số nguyên tố – giả sử x, đánh dấu tất cả các ước của x: 2x, 3x, 4x,… trong đoạn [x, N] không phải số nguyên tố.

- B3. Tìm số tiếp theo được đánh dấu là số nguyên tố trong [x, N]. Nếu không còn số nào, thoát chương trình. Nếu còn, gán nó bằng x và lặp lại bước 2.

- B4. Khi kết thúc giải thuật, các số không bị đánh dấu là các số nguyên tố

***// Thuật toán sàng nguyên tố***

***// Nếu một số là số nguyên tố, thì tất cả các bội của nó không phải số nguyên tố***

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

if (check[i] == true) {

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

check[j] = false;

}

}

}

***// In ra các số là số nguyên tố***

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

if (check[i] == true) {

System.out.print(i + " ");

}

}

***Xây dựng chương trình:***

import java.util.\*;

import java.lang.\*;

import java.io.\*;

class Eratosthenes {

public static void main (String[] args) throws java.lang.Exception {

int N = 1000;

boolean[] check = new boolean[N + 1];

***// Khởi tạo tất cả các số [2...N] đều là số nguyên tố***

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

check[i] = true;

}

***// Thuật toán sàng nguyên tố***

***// Nếu một số là số nguyên tố, thì tất cả các bội của nó không phải số nguyên tố***

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

if (check[i] == true) {

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

check[j] = false;

}

}

}

***// In ra các số là số nguyên tố***

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

if (check[i] == true) {

System.out.print(i + " ");

}

}

}

}

**3. Cho một mạng lưới các sân bay có các chuyến bay trực tiếp giữa hai sân bay.**

**Hãy tìm sân bay có thể bay trực tiếp tới nhiều thành phố nhất. (bài toán tìm đỉnh**

**có bậc lớn nhất trong đồ thị)**

***Thuật toán***

FineAirport()

***//Nhập ma trận kề đồ thị mạng lưới sân bay mj***

***//Tìm bậc lớn nhất các đỉnh***

max = tonghang(0);

for(i=1….N-1)

if(max < tonghang(i))

max = tonghang(i)

***//Tìm đỉnh có bậc lớn nhất bằng sân bay cần tìm***

for i = 0….N-1

if(tonghang(i) = max)

print(i); //Sân bay từ đỉnh i