Homework 1

Bài tập I

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

M2 = ;

Với K = {s, q, h} – h là trạng thái dừng

∑ = {0 , 1, □, >} – với > là kí tự khởi động và là □ kí tự kết thúc.

Bảng hàm chuyển

|  |  |  |  |
| --- | --- | --- | --- |
| STT | K | ∑ |  |
| 1 | S | 0 | (s, 0, →) |
| 2 | S | 1 | (s, 1, →) |
| 3 | S | □ | (q, □, ←) |
| 4 | s | > | (s, >, →) |
| 5 | q | 1 | (h, 0, − ) |
| 6 | q | 0 | (q, 1, ←) |
| 7 | q | > | (h, >, − ) |

1. 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.

M3 = ;

Với K = {s, h} – h là trạng thái dừng

∑ = {0 , 1, □, >} – với > là kí tự khởi động và là □ kí tự kết thúc.

Bảng hàm chuyển

|  |  |  |  |
| --- | --- | --- | --- |
| STT | K | ∑ |  |
| 1 | S | 0 | (s, 0, →) |
| 2 | S | 1 | (s, 1, →) |
| 3 | S | □ | (h, □, − ) |
| 4 | s | > | (s, >, →) |

Bài tập II

1. Chứng minh hàm factorial( a ) = a! là hàm đệ quy nguyên thủy:

Ta có: f ( 0 ) = 1;

f( S(a) ) = mul ( a, f (sub(a – 1)));

mà mul(a,b) = a \* b và sub(a,b) = a - b là hàm đệ quy nguyên thủy nên f là hàm đệ quy nguyên thủy

1. Chứng minh hàm Proper subtraction a ┴ b : If a ≥ b then a-b else 0.

Xét a, b ∈ N

Ta có Ps(a, 0) = a;

Ps(S(a), b) = P11( Ps (a, b));

S(a) và P là các hàm đệ quy nguyên thủy nên hàm Ps là hàm đệ quy nguyên thủy.

1. Chứng minh hàm minimum(a1 , a2 , … , an)

Xét hàm min(a, b) = sub(a, Ps(a,b));

Với sub(a, b) = a – b và Ps(a, b) là hàm proper subtraction là các hàm đệ quy nguyên thủy nên min(a, b) là hàm đệ quy nguyên thủy.

Giả sử min(a1, a2, …., am) là hàm ĐQNT m đối ta sẽ CM min(a1, a2, … am+1) là hàm ĐQNT m + 1 đối.

Thật vậy: min(a1, a2, …., am+1) = min(am + 1, min(a1, a2, …., am));

Ta có min(a1, a2, … am) và min(a, b) là các hàm ĐQNT nên min(a1, a2, … am+1) là hàm ĐQNT m + 1 đối. Vậy minimum(a1 , a2 , … , an) là hàm ĐQNT n đối.

Bài tập III

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

Thuật toán Euclid tìm UCLN của 2 số nguyên dương a và b

Input: 2 số nguyên dương a và b

Output: UCLN của a và b;

* 1. Thuật toán
     + Nêu a > b và a % b == 0 thì trả về b
     + Nếu a > b và a % b != 0 thì ta sẽ tìm lặp lại với a = b và b = a – b
     + Lặp lại cho tới khi a % b == 0 hoặc b % a == 0
     + Tương tự với b > a
     + Độ phức tạp O(log(a + b))

gcd(a, b):

if (a % b == 0) return b

else return gcd(b, a % b)

1. Liệt kê các số nguyên tố nhỏ hơn n bằng phương pháp sàng Eratosthenes.
   1. Phân tích

Input: số nguyên n;

Output: mảng các số nguyên tố nhỏ hơn n;

* 1. Thuật toán
* Bước 1: Tạo mảng các số nguyên từ 2 đến n
* Bước 2: Đặt p = 2;
* Bước 3: Xét các số x >= p2 nếu x % p == 0 thì đánh dấu x không là số nguyên tố;
* Bước 4: Tăng p lên 1 và kiểm tra p có là số nguyên tố hay không. Nếu plaf số nguyên tố quay về bước 3. Nếu p không là số nguyên tố lặp lại bước 4;
* Khi p = [] + 1 thì dừng;
* Độ phức tạp O(n log(log(n))

1. Hãy tìm sân bay có thể bay trực tiếp tới nhiều thành phố nhất.
   1. Phân tích

Input: ma trận vuông đối xứng cỡ n\*n với n là số lượng các sân bay. Các giá trị trong ma trận là 0 hoặc 1.

Output: Các sân bay có nhiều đường bay nhất.

* 1. Thuật toán
     + Bước 1: Tạo ma trận vuông tương ứng với đồ thi
     + Bước 2: Tính tổng các hàng
     + Bước 3: Tìm hàng có tổng lớn nhất trong ma trận
     + Độ phức tạp O(n2)