Extended Euclidean Algorithm এবং একটুখানি Modular Multiplicative Inverse

GCD (Greatest Common Divisor) বের করার জন্য হয়তো সবাই ইউক্লিডের পদ্ধতি ব্যবহার করেই দেখেছ। এটা আমার জানা মনে দুটি সংখ্যার GCD বের করার সব থেকে সহজ এবং efficient উপায়। যাহোক Extended Euclidean Algorithm জানার জন্য এই পদ্ধতিটাই কাজে লাগে। এখন দেখাই আগে কিভাবে GCD বের করতে হয় যদিও প্রায় সবার জানা আছে।

Python Code:

Python

```python
def GCD(x, y):
    if(y==0):
        return x
    return GCD(y, x%y)
```

C++ Code:

Recursive Method:

C++

```cpp
int GCD(int x, int y)
{
  if (y==0) return x;
  return GCD(y,x%y);
}
```

Iterative Method:

C++

```cpp
int GCD(int x, int y)
{
  while (y != 0)
  {
    int temp = y;
    y = x % y;
    x = temp;
  }
  return x;
}
```

এবার কাজে আসি। প্রথমে দেখা দরকার কিভাবে Extended Euclid Algorithm কাজ করে। তার আগে তোমাদেরকে বলবো Extended Euclid Method দিয়ে হাতে-কলমে কিভাবে সমীকরন সমাধান করা যায় সেটা দেখে আসতে। এজন্য এই লিংকে একটু ঢু মেরে আসো।

কাজে ফিরে আসি। Extended Euclid এর মাধ্যমে সমাধান করার জন্য সমীকরনটা অবশ্যই এমন কাঠামোর হতে হবে: **$ax + by = GCD(a, b)$**

অর্থাৎ যদি ডান পাশে যে সংখ্যা থাকবে সেটা অবশ্যই a, b এর GCD হইতে হবে। কেবল সেক্ষেত্রেই আমরা x এবং y এর মান বের করতে পারবো Extended Euclidean Algorithm এর মাধ্যমে।

এখন একটা উদাহরন দিলে বুঝবে সুত্রটা কেমন সময় ব্যবহার করা যায়। ধর এক একটা আম এবং কলার মুল্য যথাক্রমে ১৫ এবং ৭ টাকা। একজন মহিলা ৮৫০ টাকা দিয়ে তাইলে কতটা আম এবং কলা কিনতে পারবে? এক্ষেত্রে এই সুত্রটা ব্যবহার করা যেতে পারে। অর্থাৎ $15x+7y=850$. তবে

এটা কেবল উদাহরন। Extended Euclid Algorithm এর জন্য ডান পাশের ধ্রুবক মানটি অবশ্যই (15, 7) এর গসাগু হইতে হবে।

প্রথমেই জানা দরকার দুটি সংখ্যার ভাগশেষ বের করার সুত্র :

$$X \% Y = X - \mathrm{floor}(X/Y)*Y$$

Extended Euclid এর সুত্রটা এই form এ কাজ করে: $r_i = ax_i + by_i$ ……(1)

এখানে r হল রিমাইন্ডার। এখন ভাগশেষ বের করার সুত্রটা যদি এখানে r_i বের করার জন্য ব্যবহার কর তাইলে সুত্রটা কিছুটা এমন হয়।

যদি $q_i = \mathrm{floor} (r_{i-2} / r_{i-1})$ লিখি তাইলে সুত্রটা দাড়ায়:

এখন (1) নং সমীকরন থেকে যদি r_{i-1} এবং r_{i-2} মান উপরের সমীকরনে বসাও তাইলে পাবে:

$$r_i = (ax_{i-2} + by_{i-2}) - q_i (ax_{i-1} + by_{i-2})$$

$$=> r_i = a (x_{i-2} - q_i x_{i-1}) + b (y_{i-2} - q_i y_{i-2})$$

এখানে $x_i=(x_{i-2} - q_i x_{i-1})$ …… (2) এবং $y_i= (y_{i-2} - q_i y_{i-2})$ ….. (3)

Initially r_i এবং r_2 এভাবে পাই:

$r_1 = a(1) + b(0) = a$ // x=1, y=0

$r_2 = a(0) + b(1) = b$ // x=0, y=1

r_1 এবং r_2 এর মান তো পেয়ে গেলাম এখন শুধু q_i , r_i , x_i , y_i এর মানগুলা লুপ ঘুরিয়ে কেবল আপডেট করব এবং a%b==0 হইলে অর্থাৎ r এর মান ০ হইলে লুপের কাজ শেষ করে দিব। এখন একটা উদাহরন দিলে ভাল ভাবে বুঝতে পারবে আরও। ধর, a=23 এবং b=120.

Initial Step:

```
r1 = a(1) + b(0) = 120 x 1 + 23 x 0 = 120 // x=1, y=0
r2 = a(0) + b(1) = 120 x 0 + 23 x 1 = 23   // x=0, y=1
```

Iterative Steps:

```
Step 1:
         a=23, b=120 ,  r3 = 120 % 23 = 5 ,  q3 = 120 / 23 = 5
         x3 = 0 – 5*1 ,  y3 = 1 – 5*0     /// From (2) and (3)

Step 2:
        a=5, b=23 ,  r4 = 23 % 5 = 3 ,  q4 = 23 / 5 = 4
        x4 = 1 – 4*(-5) = 21 ,  y4 = 0 – 4*1 = -4

Step 3:
        a=3, b=5 ,  r5 = 5 % 3 = 2 ,  q5 = 5 / 3 = 1
        x5 = -5 – 1*21 = -26 ,  y5 = 1 – 1 * (-4) = 5

Step 4:
        a=2, b=3 ,  r6 = 3 % 2 = 1 ,  q6 = 3 / 2 = 1
       x6 = 21 – 1*(-26) = 47 ,  y6 = -4 – 1 * 5 = -9
```

লুপের সমাপ্তি, কারন পরের ধাপে r এর মান 0 হবে।

উপরের উদাহরন থেকে আমরা কিছু তথ্য খুজে পেতে পারি। যেমন যদি x এর মান পজিটিভ হয় তাইলে y নেগেটিভ এবং y এর মান পজিটিভ হলে x নেগেটিভ।

আবার যদি GCD(a, b) = 1 হয় অর্থাৎ যদি a, b co-prime হয়ে থাকে তাইলে x হবে (a modulo b) এর Modular Multiplicative Inverse, এবং y হবে (b modulo a) এর Modular Multiplicative Inverse.

অর্থাৎ উপরের উদাহরনে -9 হল 120 modulo 23 এর multiplicative inverse এবং 47 হবে 23 modulo 120 এর multiplicative inverse.

Extended Euclid ফাংশনের কোড:

Iterative Method:

C++

```
int Extended_Euclid(int A, int B, int *X, int *Y)
{
    int x, y, u, v, m, n, a, b, q, r;
    ///* B = A(0) + B(1) */
    x = 0;  /// x [i-2]
    y = 1;  /// y [i-2]
    ///* A = A(1) + B(0) */
    u = 1;  /// x[i-1]
    v = 0;  /// y[i-1]
    a = A;
    b = B;

    while ( a != 0 )
    {
        ///* b = aq + r and 0 <= r < a */
        q = b / a;

        /// GCD function
        r = b % a;
        b = a;
        a = r;

       /// /* r = A(x - uq) + B(y - vq) */ ///
        m = x - (u * q); /// m = x[i] = (x - uq)
        n = y - (v * q);  /// n = y[i] = (y - vq)
        x = u;  /// updating x[i-1] = x[i-2]
        y = v;  /// updating y[i-1] = y[i-2]
        u = m;  /// updating x[i] = x[i-1]
        v = n;  /// updating y[i] = y[i-1]
    }

    ///* Ax + By = gcd(A, B) */
    *X = x;
    *Y = y;

    return b;
}
```

Recursive Method:

C++

```
int extendedEulid(int a, int b)
{
    if(b==0)
    {
        x=1; y=0; d=a;  /// some extensions
        return a;
    }

    int ret = extendedEulid(b, a%b);   /// GCD function
    int x1 = y;         /// some extensions
    int y1 = x - (a/b) *y;
    x = x1;
    y = y1;

    return ret;
}
```

Python Code:

Python

```
def ExtendedEuclid(a, b):
    x, xi = 0, 1
    y, yi = 1, 0

    while b>0:
        q = int(a / b)
        a, b = b, a % b
        x, xi = xi - q*x, x
        y, yi = yi - q*y, y

    return (xi, yi, a)
```

যথাসম্ভব ব্যাখ্যা করে লেখার চেষ্টা করেছি। ভুলত্রুটি থাকাটা স্বাভাবিক। সংশোধন অথবা কোন সমস্যা থাকলে কমেন্টে জানাও এবং keep coding…

342 total views, 3 views today