For any nonnegative integer a and any positive integer b, gcd(a, b) = gcd(b, a mod b).

Function Euclid (m, n)

//Compute gcd(m, n) by Euclid’s algorithm

Input: two non-negative m and n, not both zero integers

Output: the greatest common divisor of m and n

if (n == 0)

then return m;

else Euclid(n, m mod n);

* the total running time is 2n\*O(n2)= O(n3)
* The input size is the number of bits it takes to encode the numbers m and n, which are └ log2 m ┘ + 1 and └ log n ┘ + 1, respectively.
* Basic operation: One-bit manipulation in the computation of a remainder.
* For the case 1 ≤ m < n, the worst-case number of recursive calls for input size s, t is W(s, t) ∈ 𝜃 𝑡 .
* K = {max((└ log2 x ┘ + 1) , (└ log y ┘))}

function extended-Euclid(x, y)

Input: Two integers x and y with x ≥ y ≥ 0.

Output: Integers i, j, d such that d = gcd(x, y) and i\*x + j\*y = d.

if (y == 0) then return (1, 0, x);

// 1\*x + 0\*0 = x

else {(i', j', d') = extended-Euclid(y, x mod y);

return (j', i'-└ 𝑥 𝑦 ┘ \* j', d');} // r = i' mod j'.

return (i, j, d) }

* The total running time is 2n\*O(n2)= O(n3)
* The input size is the number of bits it takes to encode the numbers x and y, which are └ log2 x ┘ + 1 and └ log y ┘ + 1, respectively.
* Basic operation: One-bit manipulation in the computation of a remainder.
* For the case 1 ≤ m < n, the worst-case number of recursive calls for input size s, t is W(s, t)
* K = {max((└ log2 x ┘ + 1) , (└ log y ┘))}

Which number (x, y) controls the excecution flow number of time (who is stop)