**Euclid’s algorithm for solving modular linear equation:**Algorithm:

Euclid(a,b){  
 if(b==a){  
 return a;  
 }else{  
 return Euclid(b,a%b);  
 }  
}

Analysis:

* Since the algorithm is recursive we can find the recurrence relation
* Problem is divided into parts b & a%b, thus the size of problem is
* Dividing and merging cost, constant, O(1).
* Thus, the recurrence relation is T(n)=T( )+O(n)

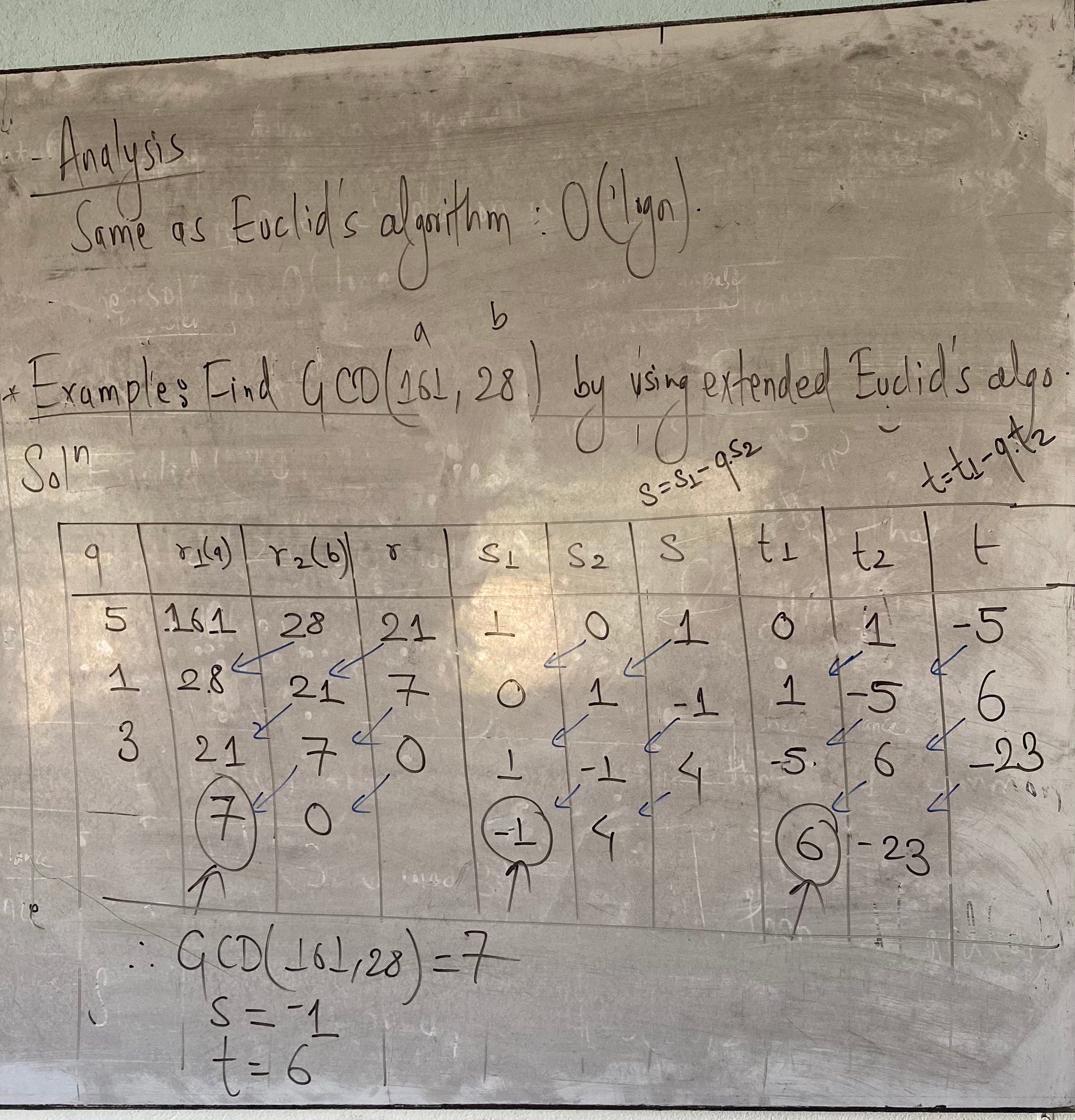
**Extended Euclid’s algorithm:**

Extension to Euclid’s algorithm which computes the coefficients of Bezout’s identity(x,y) in addition to gcd of a & b such that,  
 d = gcd(a,b) = ax + by

Algorithm:  
extended\_euclid(a,b){  
 if b == 0  
 return(a,1,0);  
 (d’,x’,y’)<-extended\_euclid(b,a%b);  
 (d,x,y)<-(d’,x’,y’-floor(a/b)\*y’)  
 return(d,x,y);  
}

Analysis:  
Same as Euclid’s algorithm

Example:



**Miller Robin randomized primality test:**

Determines whether a given number is prime or composite.

Algorithm:  
bool isPrime(int n, int k):  
1. Handle base case fro n<3  
2. If n is even return false  
3. Find the add number “d” such that n-1 can be written as d\*2r (n is odd, (n-1) is even, r>0)  
4. Do the following k times:  
 if MillerTest(n,d)==false  
 return false  
5. Return true

bool MillerTest(int n, int d):  
1. Pick random number a in range [2,n-2]  
2. Compute x = pow(a,d)%n  
3. If x == 1 or x==n-1, then return true  
4. Do the following until d becomes n-1  
 a. x=(x\*x)%n  
 b. if(x==1) then return true  
 c. if(x==n-1) then return true