CS F364
Design & Analysis of Algorithms

PROBLEM DOMAIN – NUMBER THEORY

Basic Problems and Algorithms:

- **Euclid's algorithm for *gcd*:**
  - **Correctness and Time Complexity**
- **Extended Euclidean / Aryabhatia's algorithm:**
  - **Correctness**

Greatest Common Divisor

- Notation:
  - a divides b (i.e. b is divisible by a): $a|b$
  - a does not divide b: $a \nmid b$
- Euclid's Algorithm: (given a and b s.t. $a > b > 0$)
  - $r_0 = a$; $r_1 = b$
  - $i = 2$
  - repeat

    $r_i = r_{i-2} \bmod r_{i-1}$;

    $q_i = r_{i-2} \operatorname{div} r_{i-1}$

    until $(r_{i-1} \bmod r_i == 0)$
  - return r_i

EUCLID'S ALGORITHM - CORRECTNESS

- Theorem:
  - If Euclid's algorithm returns r_k, then r_k is gcd(a,b)
- Proof:
  - Let g = gcd(a,b).
  - We claim: $\mathbf{r_k \mid g}$ and $\mathbf{g \mid r_k}$
    - and hence the conclusion.

EUCLID'S ALGORITHM - CORRECTNESS

- Algorithm gcd(a,b):
- //Precondition:(a > b > 0)
  - $r_0 = a$; $r_1 = b$
  - $i = 2$
  - repeat

    $r_i = r_{i-2} \bmod r_{i-1}$;

    $q_i = r_{i-2} \operatorname{div} r_{i-1}$

    until $(r_{i-1} \bmod r_i == 0)$
  - return r_i

// returns r_k

- Proof of $\mathbf{r_k | g}$:
  - Observe that $\mathbf{r_k | r_{k-1}}$ and

    $\mathbf{r_{k-2} = r_{k-1} * q_k + r_k}$
    - Implication: $\mathbf{r_k | r_{k-2}}$
  - Since $\mathbf{r_k | r_{k-1}}$ and $\mathbf{r_k | r_{k-2}}$ and $\mathbf{r_{k-3} = r_{k-2} * q_{k-1} + r_{k-1}}$
    - $\mathbf{r_k | r_{k-3}}$
  - Inductively, we can show that
    - $\mathbf{r_k | r_i}$ and $\mathbf{r_k | r_{i-1}}$ implies

      $\mathbf{r_k | r_{i-2}}$ for all $\mathbf{i}$
  - Then $\mathbf{r_k | r_1 = b}$ and $\mathbf{r_k | r_0 = a}$.
    - i.e. $\mathbf{r_k | g}$

EUCLID'S ALGORITHM – CORRECTNESS [CONTD.]

- Algorithm gcd(a,b):
- //Precondition:(a > b > 0)
  - $r_0 = a$; $r_1 = b$
  - $i = 2$
  - repeat

    $r_i = r_{i-2} \bmod r_{i-1}$;

    $q_i = r_{i-2} \operatorname{div} r_{i-1}$

    until $(r_{i-1} \bmod r_i == 0)$
  - return r_i

// returns r_k

- Proof of $\mathbf{g} \mid \mathbf{r_k}$:
  - Observe that $\mathbf{g} \mid \mathbf{r_0}$ and $\mathbf{g} \mid \mathbf{r_1}$
  - Since $\mathbf{r_i = r_{i-2} - q_i * r_{i-1}}$ for all $\mathbf{i}$
    - if $\mathbf{g} \mid \mathbf{r_{i-2}}$ and $\mathbf{g} \mid \mathbf{r_{i-1}}$
    - then $\mathbf{g} \mid \mathbf{r_i}$ for all $\mathbf{i >= 2}$
  - Inductively,
    - $\mathbf{g} \mid \mathbf{r_k}$

EUCLID'S ALGORITHM – TIME COMPLEXITY

- Algorithm gcd(a,b):
- //Precondition:(a > b > 0)
  - $r_0 = a$; $r_1 = b$
  - $i = 2$
  - repeat

    $r_i = r_{i-2} \bmod r_{i-1}$;

    $q_i = r_{i-2} \text{ div } r_{i-1}$

    until ($r_{i-1} \bmod r_i == 0$)
  - return r_i

// returns r_k

- Time Complexity is **O(k*f(a,b))**
  - where **k** is the (worst case) <u>number of iterations</u>
  - and **f(a,b)** is the cost of basic operations **div** or **mod** which is
    - **O(1)** assuming <u>*uniform cost model*</u>
    - ?? otherwise

EUCLID'S ALGORITHM – TIME COMPLEXITY

- Algorithm gcd(a,b):
- //Precondition:(a > b > 0)
  - $r_0 = a$; $r_1 = b$
  - $i = 2$
  - repeat

    $r_i = r_{i-2} \bmod r_{i-1}$;

    $q_i = r_{i-2} \text{ div } r_{i-1}$

    until $(r_{i-1} \bmod r_i == 0)$
  - return r_i

// returns r_k

- When does the worst case happen?
  - Consider the case: **a ≈ b**
    - Then **(a mod b) << b**
  - Consider the case: **a >> b**,
    - **(a mod b) ≈ b << a**
  - *Either case will lead to quick convergence*:
    - i.e. **they will not result in worst case behavior**

EUCLID'S ALGORITHM – TIME COMPLEXITY

- Algorithm gcd(a,b):
- //Precondition:(a > b > 0)
  - r_0 = a; r_1 = b
  - i = 2
  - repeat

    $r_i = r_{i-2} \bmod r_{i-1}$;

    $q_i = r_{i-2} \text{ div } r_{i-1}$

    until ($r_{i-1} \bmod r_i == 0$)
  - return r_i

// returns r_k

- The worst case behavior will not be exhibited
  - if **a ≈ b or** if **a >> b**
- The worst case will happen when
  - neither of the above (conditions) is true for a and b
    - i.e. **!(a ≈ b) and !(a >> b)**
  - and that is also <u>the case for r_i and r_{i-1} in each iteration</u>
    - i.e. **!(r_{i-1} ≈ r_i) and !(r_{i-1} >> r_i)** for each i

EUCLID'S ALGORITHM – TIME COMPLEXITY

- Algorithm gcd(a,b):
- //Precondition:(a > b > 0)
  - $r_0 = a$; $r_1 = b$
  - $i = 2$
  - repeat

    $r_i = r_{i-2} \bmod r_{i-1}$;

    $q_i = r_{i-2} \text{ div } r_{i-1}$

    until ($r_{i-1} \bmod r_i == 0$)
  - return r_i

// returns r_k

- When does the worst case happen?
- Fibonacci numbers fit the bill:
  - Consider $a==F_m$ and $b==F_{m-1}$
  - Then the sequence is:
    - $r_0 = F_m$
    - $r_1 = F_{m-1}$
    - $r_2 = F_m - F_{m-1} = F_{m-2}$
    - $r_3 = F_{m-1} - F_{m-2} = F_{m-3}$
    - …
    -

In each iteration, the reduction in size is not significant:

the quotient q_i is 1 and the remainder r_i is comparable to r_{i-1}

EUCLID'S ALGORITHM – TIME COMPLEXITY

- When does the worst case happen?
- Fibonacci numbers fit the bill for the worst case behavior:
  - If **a = F_{m+1}** **b = F_m** then **gcd(a,b)** will take **m** steps
  - $F_m \approx ((1 + \sqrt{5})/2)^m$
    - i.e. $m = \Theta(\log(F_m))$

•Time Complexity of **gcd** is **$\Theta(\log(N))$,** where **N** is the larger of the two inputs, assuming ***the uniform cost model***

EUCLID'S ALGORITHM – TIME COMPLEXITY

- Time Complexity of gcd is $\Theta(\log(N))$, where N is the larger of the two inputs, assuming ***the uniform cost model***

- What if we use the logarithmic cost model?
    - Division operation would take time that is dependent on the size of the input values:
        - it could cost as much as **b*b** time for **b** bits.
        - in which case, the time complexity of gcd is $\Theta((\log(N))^3)$

Extended Euclid's Theorem

- Theorem:
  1. For all $a > b > 0$, there exist integers x and y such that

     $gcd(a,b) = a*x + b*y$
- Proof : Consider Euclid's algorithm
  - Loop Invariant:
    - $r_i = r_{i-2} - q_i * r_{i-1}$
  - So, if gcd(a,b) returns r_k
    - $r_k = r_{k-2} - q_k * r_{k-1}$
    - $= r_{k-2} - q_k * (r_{k-3} - q_{k-1} * r_{k-2})$
    - $= \ldots$
    - $= x*r_0 + y*r_1$

```
gcd(a,b)
r0 = a;  r1 = b
i = 2
repeat
    ri = ri-2 mod ri-1 ;
    qi = ri-2 div ri-1
  until (ri-1 mod ri == 0)
return ri
```

Aryabhatia's algorithm

- Theorem:
  1. For all a > b > 0 , there exist integers x and y such that
     gcd(a,b) = a*x + b*y
  2. Moreover, x and y can be computed in polynomial time.
- Proof of 2 (Algorithm):

```
gcdAB(a,b):  //Precondition:(a > b > 0)
    if (b=0) return (a, 1, 0);
    else {
        (d, x1, y1) = gcdAB(b, a mod b)
        // i.e. d = gcd(a, b) = gcd(b, a mod b) = b*x1 + (a mod b)*y1
        // i.e. d = gcd(a,b) = b*x1 + a*y1 – (a div b)*b*y1
        return (d, y1, x1 – (a div b)*y1);
        }
```