N and n Relations

Largest num with n bits can be rep. By

Addition and Multiplication Costs:

In Unit Cost: loop with addition O(N log(N)) or O(n)

Multiplication: O( ) or O()

Adding or Multiplying Number Sizes

Add two n-bit nums = n+1 bit num max

Mult two n-bit nums = 2n bit num max

Big – Oh, Big – Theta, and Big - Omega

f = 2n, g = 🡪 f = Ω(g) f =, g = 3n 🡪 f = Θ(g)

f = 3, g = 🡪 f = O(g) f = , g = 🡪 f = Θ(g)

Everything Modular

x ≡ z mod N implies N | (x − z)

13 mod 7 = 6 100 mod 9 = 1 100 mod 20 = 0 -21 mod 10 = 9

-3 mod 12 = 9 0 mod 6 = 6, 12, …

(a/b)(modN) exists if and only if GCD(b, N) = 1

20/15 (mod 50) 🡪 undefined 109/19 (mod 115) 🡪 defined

≡ 1 (mod p).

mod 53 = 1

(25 + 7 · 51) mod 53 =

(25mod 53)·( 25mod 53) + (7 mod 53)·( mod 53) · ( mod 53) = 8

Extended - Euclid

extended-euclid(90, 66)

d = 90x + 66y

90 = 66(1) + 24 6 = 66(-1) + (90(1) + 66(-1))(3) = 6 = 90(3) + 66(-4)

66 = 24(2) + 18 6 = 24(1) + (66 + 24(-2))(-1) = 6 = 66(-1) + 24(3)

24 = 18(1) + 6 6 = 24(1) + 18(-1)

18 = 6(3) + 0

so….d = 6, x = 3, y = -4

Some Helpful Equations

1 + 2 + … + n =

r ≠ 0 🡪

r > 1 🡪 f(i) = 🡪 f(i) = Θ()

0 < r < 1 🡪 f(i) = 🡪 f(i) = Θ(1)

Modular Tricks with Exponents

a0 = 5

for i = 1 to 4

ai = ai−1 ∗ ai−1 mod 18

a0 = 5, a1 = 7, a2 = 13, a3 = 7, a4 = 13

13 · 13 ≡ 7 mod 18

7 · 7 ≡ 13 mod 18

13 · 5 ≡ 11 mod 18

Modular Inverse

6/5 mod 49 🡪 GCD(5, 49) = 1 🡪 extended-euclid(49, 5)

d = 49x + 5y

49 = 5(9) + 4 1 = 5 + (49 + 5(-9))(-1) 1 = 49(-1) + 5(10)

5 = 4(1) + 1 1 = 5 + 4(-1)

4 = 1(4) + 0

so…d = 1, x = -1, y = 10 so…10 is the modular inverse of 5 mod 49

6/5 mod 49 = 6 · 10 mod 49 = 11

Random Primes

prob of rand prime = 1.44/n so trying to find x primes in n numbers expect tries

RSA

RSA. p, q are prime. N = p\*q. GCD(e, (p-1)\*(q-1)) = 1. d = inverse of e mod (p-1)\*(q-1).

N, e are public. Anyone can encrypt a message x with . Person with d decrypts with

P = 5, q = 11, N = 55. (p-1)\*(q-1) = 4\*10 = 40. e = 3 (it’s usually 3).

Mult inverse of 3 mod 40 = d = 27. Can confirm with 27\*3 = 1 mod 40

Send x = 9. . Decrypt:

Define T(n)

T(n) = # rec calls \* T(size of new input) + O(local work)

Master Theorem

if: a > 0, b > 1, d ≥ 0

O() if d < O( if d = O() if d <

Master Theorem Examples

T(n) = T(n/2) + O(1) a = 1, b = 2, d = 0. so O() = O(log(n))

T(n) = 2\*T(n/2) + O(n) a = 2, b = 2, d = 1. so O() = O(nlog(n))

T(n) = 3\*T(n/2) + O(n) a = 3, b = 2, d = 1. so O(= O()

Solving Exact Recurrences

T(n) = 2T(n/3) +1;  T(1) = 1

number of nodes at Level i is

the value in the nodes at Level i:

bottom nodes have the value 1 (since the base case is n = 1)

Solving for i, we get that the bottom level is Level

In levels above bottom, local contribution at each node is 1 (from the +1 term in the recurrence)

local contribution at the nodes in the bottom level is also 1, because T(1) = 1.

total contribution at level i is × 1 =

Summing the total contribution at each level is

So T(n) = = = = =

2 × (3log3 2 ) log3 n − 1 = 2 × (3log3 n ) log3 2 = 2n log3 2 − 1

T(n) = 2T(n/3);  T(1) = 2

Same as above, but now no local work in nodes above the bottom level. And the work at the bottom level is 2

So T(N) = = = =

Solving for Run-Times without Master Theorem

T(n) = T(n − 2) + 5n; T(1) = T(2) = 1

Number of nodes at level i = n - 2(i)

Solving for i, we get that the bottom level is level: 1 = n - 2(i) 🡪 i = n/2

= 5n + T(n-2)

= 5n + (5n -2) + T(n-2-2) = 5n + (5n - 2) + T(2(2))

= 5n -2(0) + 5n - 2(1) + 5n - 2(2) + T(2(3)

= 5n -2(0) + 5n - 2(1) + 5n - 2(2) + … + 5n - 2(n/2)

So you’re adding 5n n/2-times, so the runtime is Θ(n2)

T(n) = T(n/4) +

Number of nodes at level i = n/4i

Solving for i, we get that the bottom level is level: 1 = n/4i = log4n

= n1/3 + T(n/4)

=

so the total runtime is Θ(n1/3)

Fast Integer Multiplication

x\*y. x = 87 (= 10101112 in binary) and y = 67 (= 10000112).

xL = (0101)2 xR = (0111)2 xL = 5 and xR = 7

yL = (0100)2 yR = (0011)2 yL = 4 and yR = 3

P1 = multiply(xL, yL) = multiply(5, 4) = multiply(0101, 0100)

P2 = multiply(xR, yR) = multiply(7, 3) = multiply(0111, 0011)

P3 = multiply(xL + xR, yL + yR) = multiply(12, 7) = multiply(1100, 0111)

P1 = 5 × 4 = 20

P2 = 7 × 3 = 21

P3 = 12 × 7 = 84.

x\*y = P1 × + (P3 − P1 − P2) × + P2

20 · 256 + 43 · 16 + 21 = 5120 + 688 + 21 = 5829 or 87 · 67 = 5829,