Prime Numbers

Review

//O(n)\*O(n) = O(n2)

for (int i = -92; i < n - 5; i+= 7) // O(n)

for (int j = i; j < n; j++) //O(n)

f(); // O(1)

n + (n-1) + (n-2) + …. (1) = n (n+1) / 2 = 1/2n2 + 1/2n = n2

//O(n) Ω(1) average case?

isPrime(n)

for i ← 2 to n-1

if n divisible by i

return false

end

end

return true

end

for n even, O(1)

for n multiple of 3 O(2) = O(1)

//O(sqrt(n))

isPrime(n)

for i ← 2 to √n

if n divisible by i

return false

end

end

return true

end

homework

2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

2 3 - 5 - 7 - 9 -- 11 -- 13 14 15 16 17 18 19 20

HW: find (the count) all primes between these two

10000 11000

2 → 11000

10000000000 10000000005

10000000000 10000002005

10000000000 20000000005

to determine a big range high up, use a smaller range:

10000000000 10000000005

2 sqrt(10000000005)

# Fermat

cn != an+bn , n > 3 “Fermat’s last theorem”

“little theorem”

for prime p, and a witness 2 <= a < p ap-1 mod p = 1

1928375928357

21928375928357

First, how long does arithmetic take?

a \* b = O(1)

a + b = O(1)

n=4 digits

|  |  |  |  |
| --- | --- | --- | --- |
| 4 | 3 | 9 | 8 |

+

|  |  |  |  |
| --- | --- | --- | --- |
| 9 | 1 | 3 | 5 |

2n operations = O(n)

n=4 digits

|  |  |  |  |
| --- | --- | --- | --- |
| 4 | 3 | 9 | 8 |

\*

|  |  |  |  |
| --- | --- | --- | --- |
| 9 | 1 | 3 | 5 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 2 | 1 | 9 | 9 | 0 |

O(n2) with convolutions O(n log n)

size of result?

n=4 digits

|  |  |  |  |
| --- | --- | --- | --- |
| 9 | 9 | 9 | 9 |

\*

|  |  |  |  |
| --- | --- | --- | --- |
| 9 | 9 | 9 | 9 |

output n = 8

999921298471924

10! = 3628800

n! mod 10, n = 1012, n = 1012

(a\*b) mod m = ((a mod m) (b mod m)) mod m

prod = 1;

for i = 1 to n

prod = prod \* i % 10;

if p is prime ⇒ ap-1 mod p == 1

2 <= a < p

if ap-1mod p == 1 ⇒ **p prime? probably**

Carmichael numbers

561 = 11\*17

a = 2

p = 11

210 mod 11 ==1

p=33

232 mod 33 ==1

bool fermat(p, k)

for i = 1 to k

a ← random(2, p-1);

if ap-1 mod p != 1

return false

end

return true (probably)

end

//O(n)

// xn mod m

bruteforcepowermod(x, n, m)

prod ← 1

for i ← 1 to n

prod ← prod \* x mod m

end

x17 = x\*x\*x\*x\* x\*x\*x\*x \* x\*x\*x\*x \* x\*x\*x\*x \* x

x17 = x1 x16

x16 = (x8)2

x8 = (x4)2

//O(log n)

// xn mod m

powermod(x, n, m)

prod ← 1

while n > 0

prod ← prod \* x mod m

x ← (x\*x) mod m

n ← n/2

end

return prod

end

1000000000000000000007

# Miller-Rabin

Problem with Fermat: Carmichael numbers. p = 561

x2 mod m = 1 (x mod m) (x mod m) =1



<https://en.wikipedia.org/wiki/Miller%E2%80%93Rabin_primality_test>

n = 221 prime or not?

d = n-1 = 220 = 128 + 64 + 16 + 4

bits = 11010100

2sd d = 55, s = 2

ap-1mod p ==1

MillerRabin(p, k)

for i ← 1 to k // k trials  
 a← random[2, *n* − 2]  
 *x* ← *ad* mod *n*  
 **if** *x* = 1 or *x* = *n* − 1 **then**  
 **continue** WitnessLoop  
 for j ← 1 to *s* − 1  
 *x* ← *x*2 mod *n*  
 **if** *x* = 1 **then**  
 **return** false *composite (Carmichael?)*  
 **if** *x* = *n* − 1 **then**  
 **continue** WitnessLoop  
 **end**

**return** false (*composite)*  
**return** *probably prime*

# Agrawal, Kayal, Saxena,

proved analytically that without trial division, deterministically prove ***isPrime(p) true or false***

n = pq, is it possible to find p and q (in other words, factor n)

Currently:

//O(sqrt(n)

factor(n)

for i ← 2 to sqrt(n)

if n mod i == 0

RSA encryption

replacement technologies: Discrete Logarithm (Suzanne Wetzl) Elliptical Integrals

first pick two random prime numbers n = 4192 bits / 64 → 60 words? O(log n)

n ← p q O(41922)

n is on the order of 8192 bits long

**false** ← MillerRAbin(n, k) not prime (we know that)

// Dov: TODO look up the real algorithm!

RSApickKey()

p ← random(24000, 24192) // what is prob(prime)?

last bit definitely 1

repeat

if p is prime

return p

// what if p mod 3 == 0

p ← p + 30 ;// maybe?

end

end

11 13 15 17 19

11 13 17 19

m’ ← encrypt(m, kpub)

m ← decrypt(m’, kprivate)

p← 61, q← 53

n ← p q ← 3233

1 < *e* < 3120

e← 17

d ← 2753

m’← 6517mod 3233 = 2790

m← 27902753mod 3233 = 65 // get original back

In practice, RSA is attackable (plaintext attack).

So:

1. Pick a random number using a cryptographically secure generator
2. encrypt that, and send it to a server.
3. The server then uses it to communicate with us
4. secondary method is AES-256

The codebreakers David Kahn (non-technical)

Bruce Schneier, Cryptography

German submarine reports weather

grid xx-yy weather zzzzz using knowledge of the text “plaintext attack”

# Prime Number Wheel

countPrimes(n1, n2)

count ← 0   
 for i ← n1 to n2 step 2

if isPrime(i)

count++

end

isPrime(n)

for i ← 3 to sqrt(n) step 2

if n mod i == 0

return false

end

end

2, 3, 5, 6, **7,** 8, 9, 10, **11**, 12, **13**, 14, 15, 16, **17**, 18, **19**, 20, 21, 22, 23,

24, **25**, 26, 27, 28, **29,** 30

n mod 6 = 0 NOT PRIME

n mod 6 = 1

n mod 6 = 2 MOD 2 NOT PRIME

n mod 6 = 3 MOD 3 NOT PRIME

n mod 6 = 4 MOD2 NOT PRIME

n mod 6 = 5

2, 3, 5 = 30

HW: For n < 232 print out prime or not prime using Miller-Rabin

100730001

your program prints out

prime or not prime

Next time:

bubble sort

selection sort

insertion sort

quicksort

heapsort

merge sort

shuffling

searching (linear, binary)