cmput 204 seminar 2 fall 2015

1. For each, show output for `euc(385,623)`.

```
def euc(a,b):
  print a,
  while b>0:
    a, b = b, a % b
    print a,
  print ''
  return a
```

```
def euc(a,b):
  print a,
  if b>0:
    euc(b, a%b)
```

2.
```
def eee(a,b):
  if (b==0): return a
  return eee(b, a-b)
```

   (i) Give the first six nodes in the recursion tree for `eee(1,2)`.

   (ii) Does `eee(1,2)` terminate? Justify with a statement that can be checked by induction.

3. Show the computation of 419^{560} mod (561) using the algorithm from class.

4. (i) Using the formula for the class webpage, give the number of bits in the binary representation of decimal 123456789.

   (ii) How can you check your answer to (i) ?

   (iii) As precisely as you can, give the number of bits in the binary representation of 10^{100}.

   (iv) As precisely as you can, give the number of bits in the binary representation of 10^t.

5. Consider `isComposite(n,t,True)` from class. (i) Explain output `8911 192 1527 yields root 267`.

   (ii) Explain output `8911 2270 8302 fails Fermat`.

   (iii) Assume `isComposite(n,5, )` returns True. What is the probability that n is composite? What assumption does this depend on?

   (iv) Assume `isComposite(n,2, )` returns False. What is the probability that n is composite? What assumption does this depend on?

solutions

1. `385 623 385 238 147 91 56 35 21 14 7`

2. (i)

```
( 1,  2 )
( 2, -1 )
(-1,  3 )
( 3, -4 )
(-4,  7 )
( 7,-11 )
```

(ii) No. Define b_n as the value b from the nth call, where $b_1 = 2$ is from the first call. This is a Fibonacci-style sequence: for all $n \geq 3$, $b_n = b_{n-2} - b_{n-1}$. It is sufficient to prove, for all odd $n \geq 3$, $b_n > b_{n-2}$. So b_n gets arbitrarily large, and never reaches 0.

3. `560 280 140 70 35 17 8 4 2 1`

```
  all operations are mod 561
a^1 = 419
a^2 = (a^1)^2 = 419*419 = 529
a^4 = (a^2)^2 = 529*529 = 463
a^8 = (a^4)^2 = 463*463 =  67
a^17 = a*(a^8)^2 = 419*67*67 = 419
a^35 = a*(a^17)^2 = 419*419*419 = 56
a^70 = (a^35)^2 = 56*56 = 331
a^140 = (a^70)^2 = 331*331 = 166
a^280 = (a^140)^2 = 166*166 = 67
a^560 = (a^280)^2 = 67*67 = 1
```

4. (i) $1+\lfloor \lg 123456789 \rfloor = 1+\lfloor \lg 26.8\ldots \rfloor = 27$

(ii) Convert to binary and count the bits:

`0b111010110111100110100010101`

(iii) $1 + \lfloor \lg 10^{100} \rfloor = 1 + \lfloor 100 \lg 10 \rfloor = 1 + \lfloor 100(3.321\ldots) \rfloor = 1 + 332 = 333$

(iv) $1 + \lfloor \lg 10^t \rfloor = 1 + \lfloor t \lg 10 \rfloor$

5. To answer this question, you need to read the code in the algorithm.

(i) First number 8911 is n. Next number 192 not followed by `yields` or `fails`, so this must be a non-witness (i.e. passed both Fermat and Euclid tests). Next number 1527 followed by `yields root 267`, so 1527 must be a witness, in computing 1527^{n-1} algorithm discovered that $267^2 = 1$ (mod n), failing the Euclid test.

To check: confirm $192^{n-1} = 1$ (mod n) and $267^2 = 1$ (mod n), and 267 is one of the intermediate powers considered in computing 1527^{n-1} (mod n).

(ii) 8911 is n. 2270 is non-witness. 8302 is witness that fails the Fermat test.

To check: confirm $2270^{n-1} = 1$ (mod n) and $8302^{n-1} \neq 1$ (mod n).

(iii) n is composite with probability 1, because a witness was found. This relies on the correctness of Euclid's theorem for nontrivial square roots of 1, and Fermat's theorem about a^{n-1} (mod n).

(iv) n is prime with probability at least $1 - 1/(4^t)$ where here $t = 2$, so probability at least $1 - 1/16 = 15/16$. The assumes that the psuedorandom number generator selected the two trial values of a uniformly randomly from `[2, ..., n-2]`.