Homework2

September 22, 2021

[2]:
```
########## Problem 1
def slowPower(g,A,N):
    x = 1
    for i in range(0,A):
        x = x*g % N
    return x
print("2^5 mod 10 =",slowPower(2,5,10))
```

```
2^5 mod 10 = 2
```

[1]:
```
########## Problem 2
def getBinary(A):
    binaryList = []
    while A>0:
        if A%2 == 0:
            binaryList.append(0)
        else:
            binaryList.append(1)
        A = A//2 #This does the same operation as math.floor(A/2), but without␣
 ↪ever converting to a float, and therefore avoiding the lack of precision for␣
 ↪values of A that are larger than 64 bits.
    return binaryList
print("7 in binary is",getBinary(7))
print("The list is [A_0,A_1,...,A_r] is reversed from traditional binary, as␣
 ↪A_i corresponds to the coefficient of 2^i, so read backwards if you want to␣
 ↪write in binary")
```

```
7 in binary is [1, 1, 1]
The list is [A_0,A_1,…,A_r] is reversed from traditional binary, as A_i
corresponds to the coefficient of 2^i, so read backwards if you want to write in
binary
```

[3]:
```
########## Problem 3(a)
def fastPower(g,A,N):
    binaryList = getBinary(A)
    powersOfG = [g % N] #initiate a list of the powers of g with g^1
    for i in range(0,len(binaryList)):
        newPower = powersOfG[-1]**2 % N #square the last element of the list␣
↪and add it
        powersOfG.append(newPower)
    output = 1
    for i in range(0,len(binaryList)):
        if binaryList[i]==1:
            output = output * powersOfG[i] % N #multiply all the ones␣
↪corresponding to nonzero binary coefficients
    return output
print("2^5 mod 10 =",fastPower(2,5,10))
```

```
2^5 mod 10 = 2
```

[4]:
```
########## Problem 3(b)
def fastPowerSmall(g,A,N):
    a = g
    b = 1
    while A>0:
        if A % 2 == 1:
            b = b * a % N
        A = A//2
        a = a*a % N
    return b
print("2^5 mod 10 =",fastPowerSmall(2,5,10))
```

```
2^5 mod 10 = 2
```

[5]:
```
########## Problem 4
##### part(a)
print("17^183 mod 256:")
print("slow:",slowPower(17,183,256))
print("fast:",fastPower(17,183,256))
print("fastSmall:",fastPowerSmall(17,183,256))

##### part(b)
print("11^507 mod 1273:")
print("slow:",slowPower(11,507,1237))
print("fast:",fastPower(11,507,1237))
print("fastSmall:",fastPowerSmall(11,507,1237))

##### part(c)
print("2^123456789 mod 987654321:")
#print("slow:",slowPower(2,123456789,987654321)) TOO SLOW!
print("fast:",fastPower(2,123456789,987654321))
print("fastSmall:",fastPowerSmall(2,123456789,987654321))

##### part(d)
```

```python
print("5^1000000000000 mod 10000:")
#print("slow:",slowPower(5,1000000000000,10000)) TOO SLOW!
print("fast:",fastPower(5,1000000000000,10000))
print("fastSmall:",fastPowerSmall(5,1000000000000,10000))
```

```
17^183 mod 256:
slow: 113
fast: 113
fastSmall: 113
11^507 mod 1273:
slow: 322
fast: 322
fastSmall: 322
2^123456789 mod 987654321:
fast: 804307517
fastSmall: 804307517
5^1000000000000 mod 10000:
fast: 625
fastSmall: 625
```

[0]:

```

```