Homework 2   
Geraldo Braho

1. Write an algorithm to use a fair six-sided die to generate coin flips.

The dice has six sides and the coin two sides. I assigned for each sides H & T  
1=H  
2=H

3=H

4=T

5=T

6=T

2. The section “Getting Fairness from Biased Sources” explains how you

Can use a biased coin to get fair coin flips by flipping the coin twice. But

Sometimes doing two flips produces no result, so you need to repeat the

Process. Suppose the coin produces heads three-fourths of the time and

Tails one-fourth of the time. In that case, what is the probability that you’ll

get no result after two flips and have to try again?  
  
  
First flip for head ¾ second flip for head ¾ ==> ¾ \* ¾ = 9/16

First flip for tail ¼, second flip for tail ¼ 🡪 ¼\* ¼ =1/16

The probability that you will get nothing and have to try again after two flips is 9/19 + 1/16 = 10/16

3. Again consider the coin described in Exercise 2. This time, suppose you

Were wrong, and the coin is actually fair but you’re still using the algorithm

To get fair flips from a biased coin. In that case, what is the probability that

You’ll get no result after two flips and have to try again?  
If the coin is fair, then the first flip for head would be ½, second flip for head ½

First flip for tail ½, second flip for tail ½. So the probability that you will get not result after two flips is ¼ + ¼ = 1/2, or 50%

4. Write an algorithm to use a biased six-sided die to generate fair values

Between 1 and 6. How efficient is this algorithm?  
  
Roll the dice 6 times   
 if the rolls include all 6 possible values, return first one.

Otherwise, reroll the dice again.

The efficiency of this algorithm depends on the how much biased it is. As the bias increases, the efficiency of the algorithm drops down

5. Write an algorithm to pick M random values from an array containing N

Items (where M ≤ N). What is its run time? How does this apply to the

Example described in the text where you want to give books to five people

Selected from 100 entries? What if you got 10,000 entries?  
  
For i from 0 to m-1:

j = random integer with 0 ≤ j ≤ i

swap the values of the array[i] and array[j]

next

The run time will be less than O(n) because we are picking m elements, and since it is stated that m<n, it means run time will be less than O(n) as well.

6. Write an algorithm to deal five cards to players for a poker program. Does

It matter whether you deal one card to each player in turn until every

Player has five cards or whether you deal five cards all at once to each player in turn?  
  
Randomize the deck of 52 cards, (like randomizing the list of 52 numbers) and then deal them to players, either by 1 or by 5 cards. Because it won’t change anything as long as the deck is randomized.  
  
  
  
8)What happens to Euclid’s Algorithm if A<B initially?   
Places of A and B are exchanged, and the algorithm continues in the same way, as where the A is great than B.

**function** gcd(a, b)

**while** a ≠ b

**if** a < b

b := b − a;

**else**

b := a − b;

**return** b;  
  
  
9. The least common multiple (LCM) of integers A and B is the smallest integer

that A and B both divide into evenly. How can you use the GCD to

calculate the LCM?  
  
Find the GCD of the two given numbers. Then divide the A\*B by the GCD

(A, B). The result will be the LCM (A, B).

12. \*Write a program that calculates the GCD for a series of pairs of pseudorandom

numbers and graphs the number of steps required by the GCD

algorithm versus the average of the two numbers. Does the result look

logarithmic?  
  
import random

import matplotlib.pyplot as plt

average = 0.0

def GCD(a, b):

average=float((a+b)/2)

count=0

while (b != 0):

remainder = a % b

a = b

b = remainder

count+=1

return count

gcd\_values = GCD(random.randint(1,10000),random.randint(1,10000))

print "It takes " +str(gcd\_values) + " steps to find the GCD"

gcdSteps = plt.bar(1,gcd\_values, width=0.3,color='c',label="gcd steps")

averageSteps = plt.bar(2,1,width=0.3,color='m',label="average")

plt.show()

Yes it will look logarithmic because every trial the GCD has time complexity log N will be divided by O(1) ( to find the average of two numbers)   
  
7. Write a program that simulates rolling two six-sided dice and draws a bar

chart or graph showing the number of times each roll occurs. Compare the

number of times each value occurs with the number you would expect for

two fair dice in that many trials. How many trials do you need to perform?

before the results fi t the expected distribution well?

import matplotlib.pyplot as plt

import random

count1=0

count2=0

count3=0

count4=0

count5=0

count6=0

count1\_2=0

count2\_2=0

count3\_2=0

count4\_2=0

count5\_2=0

count6\_2=0

for i in range(10000):

dice1=random.randint(1,6)

if dice1==1:

count1+=1

if dice1==2:

count2+=1

if dice1==3:

count3+=1

if dice1==4:

count4+=1

if dice1==5:

count5+=1

if dice1==6:

count6+=1

for i in range(10000):

dice2=random.randint(1,6)

if dice2==1:

count1\_2+=1

if dice2==2:

count2\_2+=1

if dice2==3:

count3\_2+=1

if dice2==4:

count4\_2+=1

if dice2==5:

count5\_2+=1

if dice2==6:

count6\_2+=1

print "You entered from first dice "+ str(count1)+ " ones, " + str(count2) + " twos, "+str(count3) + " threes, " +str(count4)+ " fours, " +str(count5)+ " fives, and "+str(count6) +" sixes."

count1 = plt.bar( 1,count1, width=0.3,color='b',label='dice 1 ')

count2= plt.bar( 2,count2, width=0.3,color='b',label='dice 1 ')

count3 = plt.bar( 3,count3, width=0.3, color='b',label='dice 1 ')

count4 = plt.bar( 4,count4, width=0.3,color='b',label='dice 1 ')

count5 = plt.bar( 5,count5, width=0.3, color='b',label='dice 1 ')

count6 = plt.bar( 6,count6, width=0.3, color='b',label='dice 1 ')

plt.show()

print "You entered from second dice "+ str(count1\_2)+ " ones, " + str(count2\_2) + " twos, "+str(count3\_2) + " threes, " +str(count4\_2)+ " fours, " +str(count5\_2)+ " fives, and "+str(count6\_2) +" sixes."

count1\_2 = plt.bar( 1,count1\_2, width=0.3,color='c',label='dice 2')

count2\_2= plt.bar( 2,count2\_2, width=0.3,color='c',label='dice 2')

count3\_2 = plt.bar( 3,count3\_2, width=0.3, color='c',label='dice 2')

count4\_2 = plt.bar( 4,count4\_2, width=0.3,color='c',label='dice 2')

count5\_2 = plt.bar( 5,count5\_2, width=0.3, color='c',label='dice 2')

count6\_2 = plt.bar( 6,count6\_2, width=0.3, color='c',label='dice 2')

plt.show()  
  
  
  
  
  
10.The fast exponentiation algorithm described in this chapter is at a very

high level. Write a low-level algorithm that you could actually implement.  
 1: Divide the exponent to powers of 2

2: Calculate the mod of the powers of 2 until the given exponent is reached

3: Combine the calculated mod values using modular multiplication rules