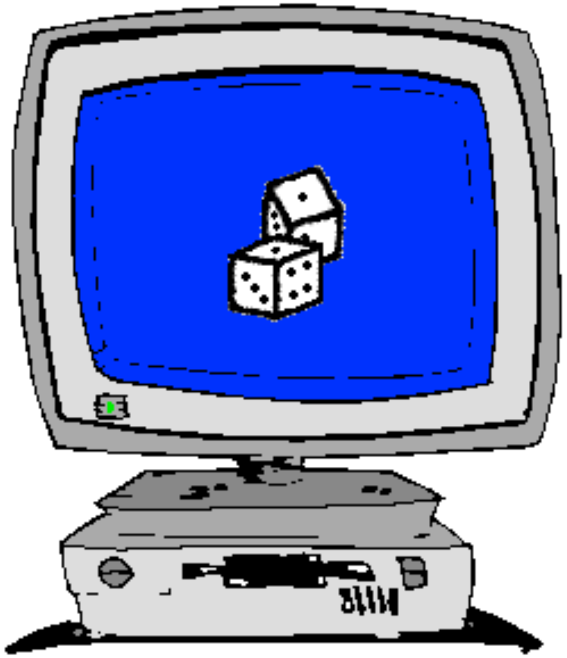
15-112
Fundamentals of Programming

Week 14, Lecture 1:
Monte Carlo Methods



April 19, 2016

Origins of Probability

France, 1654

Let’s bet:

I will roll a dice four times.
I win if I get a 1.

“Chevalier de Méré”

Antoine Gombaud

Origins of Probability

France, 1654

Hmm.

No one wants to take this bet anymore.

"Chevalier de Méré"

Antoine Gombaud

Origins of Probability

France, 1654

New bet:

I will roll two dice, 24 times.

I win if I get double-1's.

"Chevalier de Méré"

Antoine Gombaud

Origins of Probability

France, 1654

Hmm.

I keep losing money!

“Chevalier de Méré”

Antoine Gombaud

Origins of Probability

France, 1654

“Chevalier de Méré”

Antoine Gombaud

Alice and Bob are flipping a coin.
Alice gets a point for heads.
Bob gets a point for tails.
First one to 4 points wins 100 francs.

Alice is ahead 3-2 when gendarmes arrive to break up the game.

How should they divide the stakes?

Origins of Probability

Pascal

Fermat

Probability Theory is born!

Monte Carlo Method

Estimating a quantity of interest (e.g. a probability) by simulating random experiments/trials.

General approach:

Run trials

In each trial, simulate event (e.g. coin toss, dice roll, etc)

Count # successful trials

$$\text{Estimate for probability} = \frac{\#\ \text{successful trials}}{\#\ \text{trials}}$$

Law of Large Numbers:

As trials —> infinity, estimate —> true probability

Odds of Méré winning

```
def mereOdds():
    trials = 100*1000
    successes = 0
    for trial in range(trials):
        if(mereWins()):
            successes += 1
    return successes/trials


def mereWins():
    for i in range(4):
        dieValue = random.randint(1,6)
        if(dieValue == 1): return True
    return False
```

Example 1: Estimating dice roll odds

You have a certain number of dice.

They each have a certain number of sides.

Estimate the odds of getting various sums.

```
def diceOdds(numDice, sides, total)
```

Example 1: Estimating dice roll odds

```
def diceOdds(numDice, sides, total):
    trials = 10*1000
    successes = 0
    for trial in range(trials):
        if trialSucceeds(numDice, sides, total):
            successes += 1
    return successes / trials
```

```
def trialSucceeds(numDice, sides, total):
    dieTotal = 0
    for roll in range(numDice):
        die = random.randint(1, sides)
        dieTotal += die
    return (dieTotal == total)
```

Example 2: Longest run of heads or tails

You flip a coin 200 times.

H T T H T … H

Pr [longest consecutive run of heads or tails ≥ 8] = ?

Example 3: Birthday problem

- Let n = # people in a room.
- Assume people have random birthdays (discard the year).
- What is the minimum n such that:

$$\Pr[\text{ any 2 people share a birthday }] > 1/2$$

What is the probability if n = 366?

What is the probability if n = 1?

Example 3: Birthday problem

```
def birthdayOdds(n):
    trials = 10*1000
    successes = 0
    for trial in range(trials):
        if trialSucceeds(n):
            successes += 1
    return successes / trials
```

```
def trialSucceeds(n):
    seenBirthdays = set()
    for person in range(n):
        birthday = random.randint(1, 365)
        if (birthday in seenBirthdays):
            return True
        seenBirthdays.add(birthday)
    return False
```

Example 3: Birthday problem

More generally:

- Pick T numbers randomly from 1 to N.

- For what T do you have

Pr [two numbers are the same] > 1/2 ?

Answer:

T ~ N**0.5

Aside: Birthday Attack

Birthday problem often described as Birthday Paradox.

In crypto, it is often called Birthday Attack.

Aside: Birthday Attack

Cryptographic Hash Functions

A hash function that maps any string to a k-bit string/hash.

$$s \longrightarrow h(s) \qquad \text{k-bits}$$

> given h(s), should be "hard" to recover s.

> should be hard to find collisions.
i.e., two strings $s_1 \neq s_2$ with $h(s_1) = h(s_2)$.

Many applications: authentication schemes, e-cash, data integrity schemes, digital signatures, …

Aside: Birthday Attack

Cryptographic Hash Functions

1991: Rivest publishes MD5. ($k = 128$)

1993: NSA publishes SHA-0. ($k = 160$)

1995: "Umm. Never mind. Please use SHA-1 instead."

SHA-1 was/is widely used.

2001: NSA introduces SHA-2

($variants with $k = 224$, $k = 256$, $k = 384$, $k = 512$)

2012: Non-NSA introduces SHA-3

Aside: Birthday Attack

Cryptographic Hash Functions

A strategy to find a collision in, say, SHA-1:

- start hashing strings and hope two hash to the same 160 bits.

If SHA-1 is really safe, each hash h(s) should be like

random.randint(1, 2**160)

This is like the birthday problem with $N = 2^{160}$.

tries before good chance of collision:

$$\sqrt{N} = 2^{80} = 1208925819614629174706176$$

Aside: Birthday Attack

Cryptographic Hash Functions

Everybody knows this.
2^{80} is considered safely "too large".

A crypto hash function is considered "broken"
if you can beat the Birthday Attack.

Xiaoyun Wang

2005: SHA-1 collisions in 2^{69}

Later with co-authurs: 2^{63}

SHA-1: broken.

Example 4: Estimating the density of primes

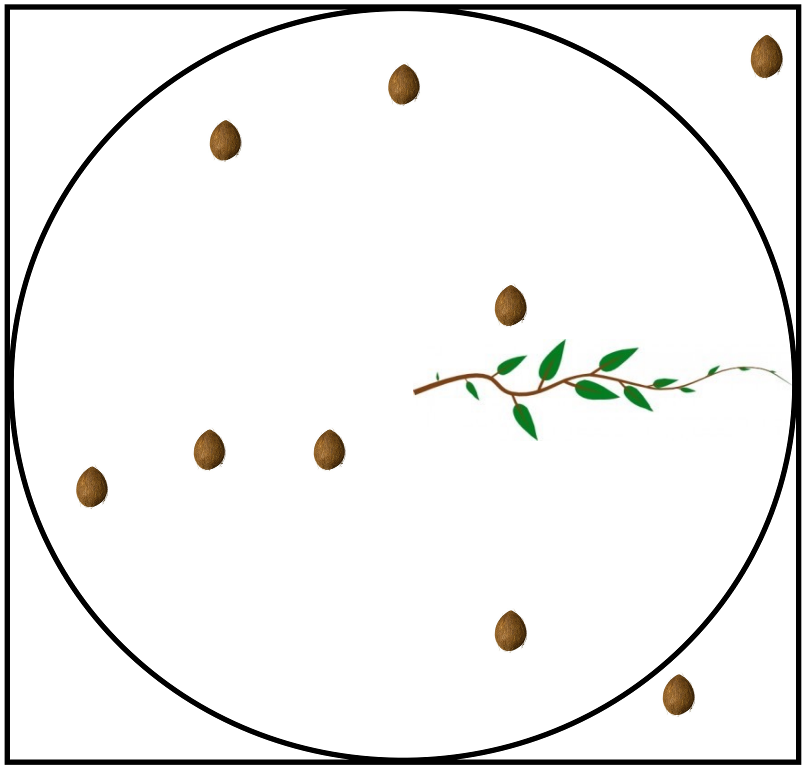
If we were to pick a random number from 1 to n, what is the likelihood that it would be a prime number?

```
def oddsOfPrime(n):
    trials = 500*1000
    successes = 0
    for trial in range(trials):
        if (trialSucceeds(n)):
            successes += 1
    return successes / trials

def trialSucceeds(n):
    randomNumber = random.randint(1 , n)
    if (isPrime(randomNumber)): return True
    else: return False
```

Example 5: Estimating Pi

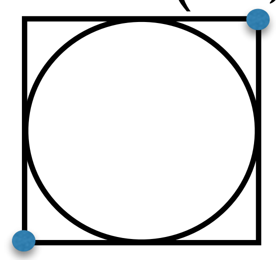
Example 5: Estimating Pi



Pr [random coconut lands in circle] =

$$\frac{\text{area of circle}}{\text{area of square}} = \frac{\pi r^2}{4r^2} = \frac{\pi}{4}$$

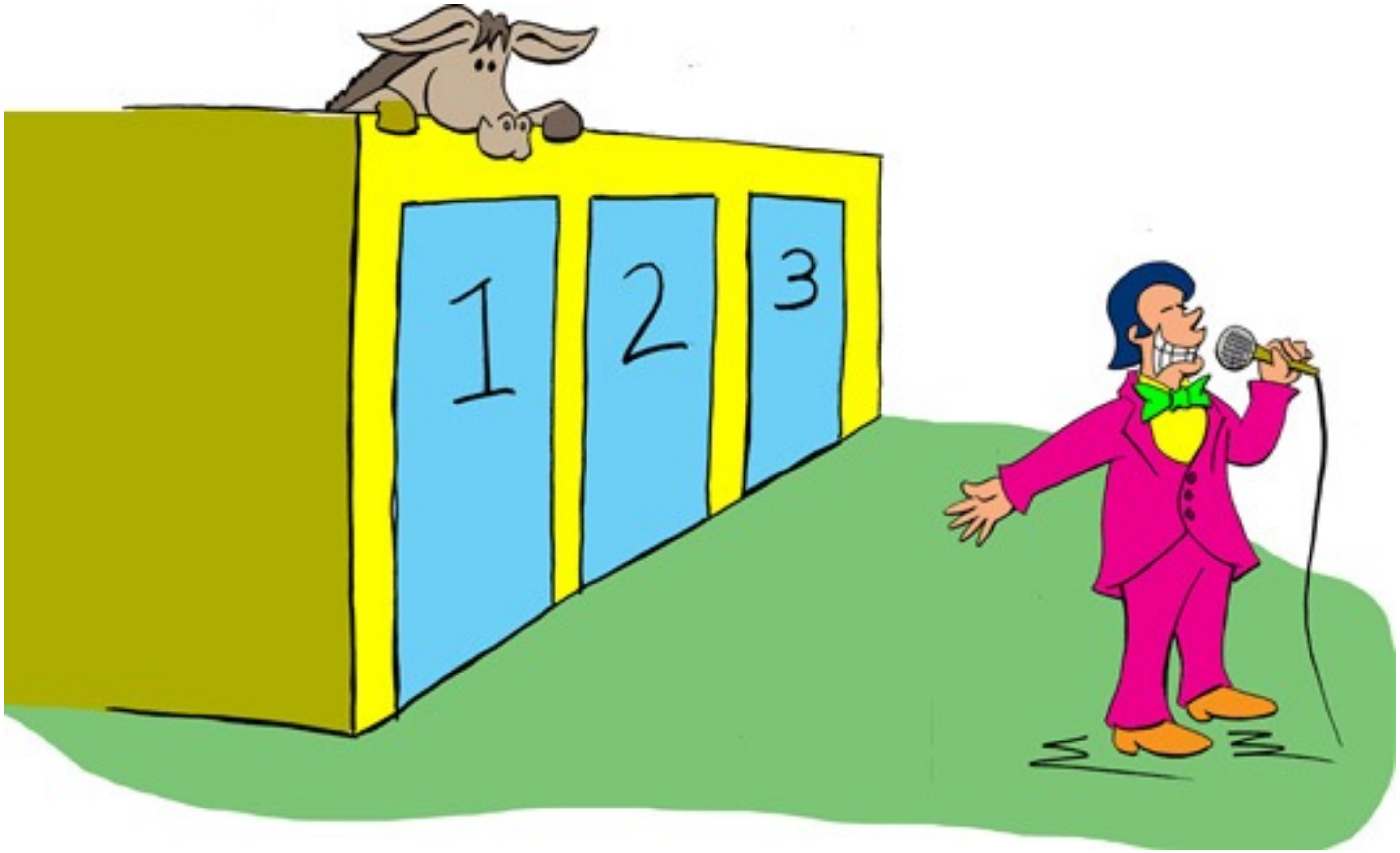
Example 5: Estimating Pi




```
def findPi(throws):                  # throws = # trials
    throwsInCircle = 0           # throwsInCircle = # successes
    for throw in range(throws):
        x = random.uniform(-1, +1)
        y = random.uniform(-1, +1)
        if (inUnitCircle(x,y)):
            throwsInCircle += 1
    return 4*(throwsInCircle/throws)
```

```
def inUnitCircle(x,y):
    return (x**2 + y**2 <= 1)
```

Example 6: Monty Hall problem



Your turn

1. What is the best strategy for the Monty Hall problem?

2. What is the probability that 2 numbers are relatively prime? (i.e. their gcd is 1)

Can assume numbers are in [1, 2^30]