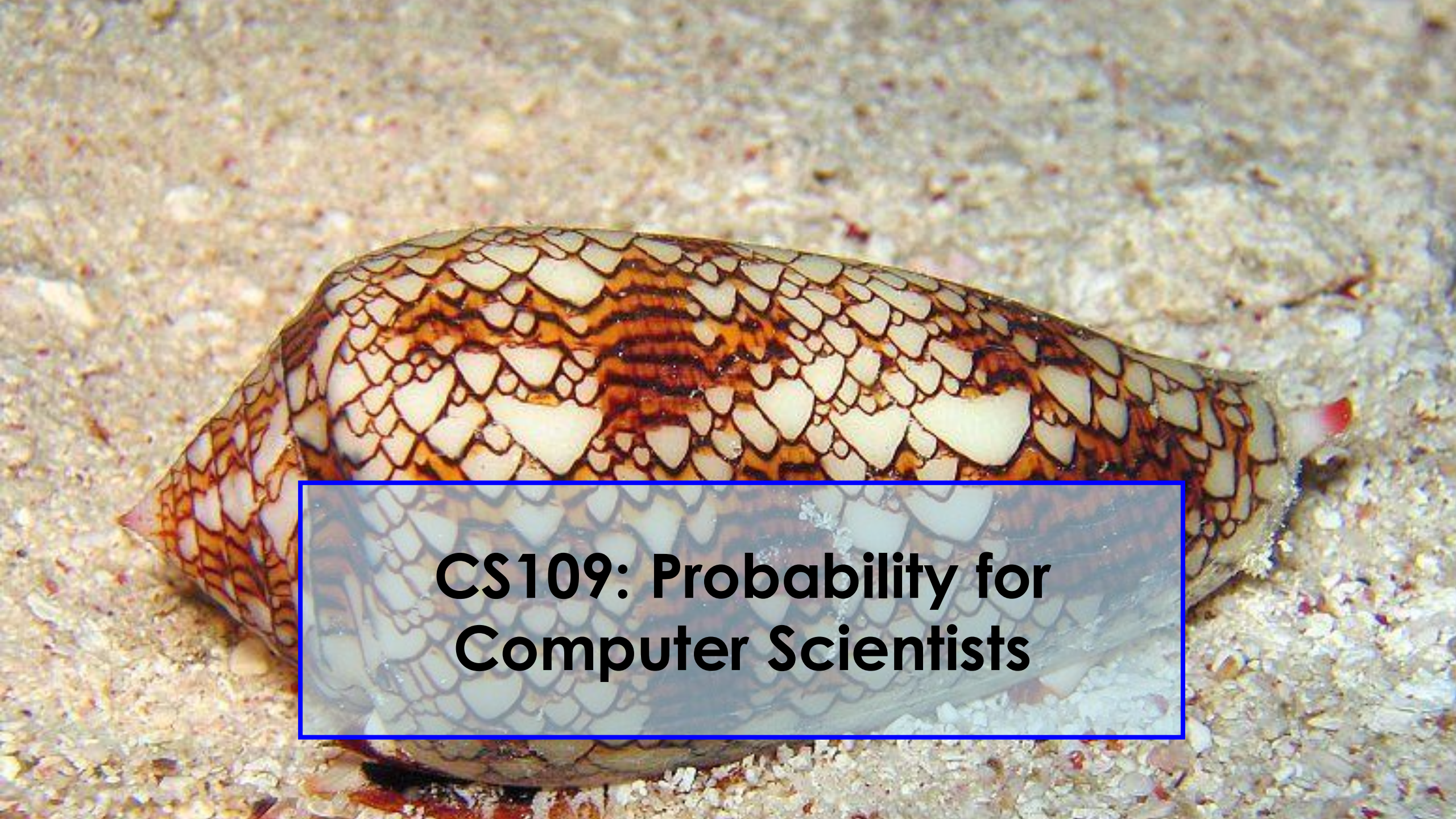


CS109: Probability for Computer Scientists

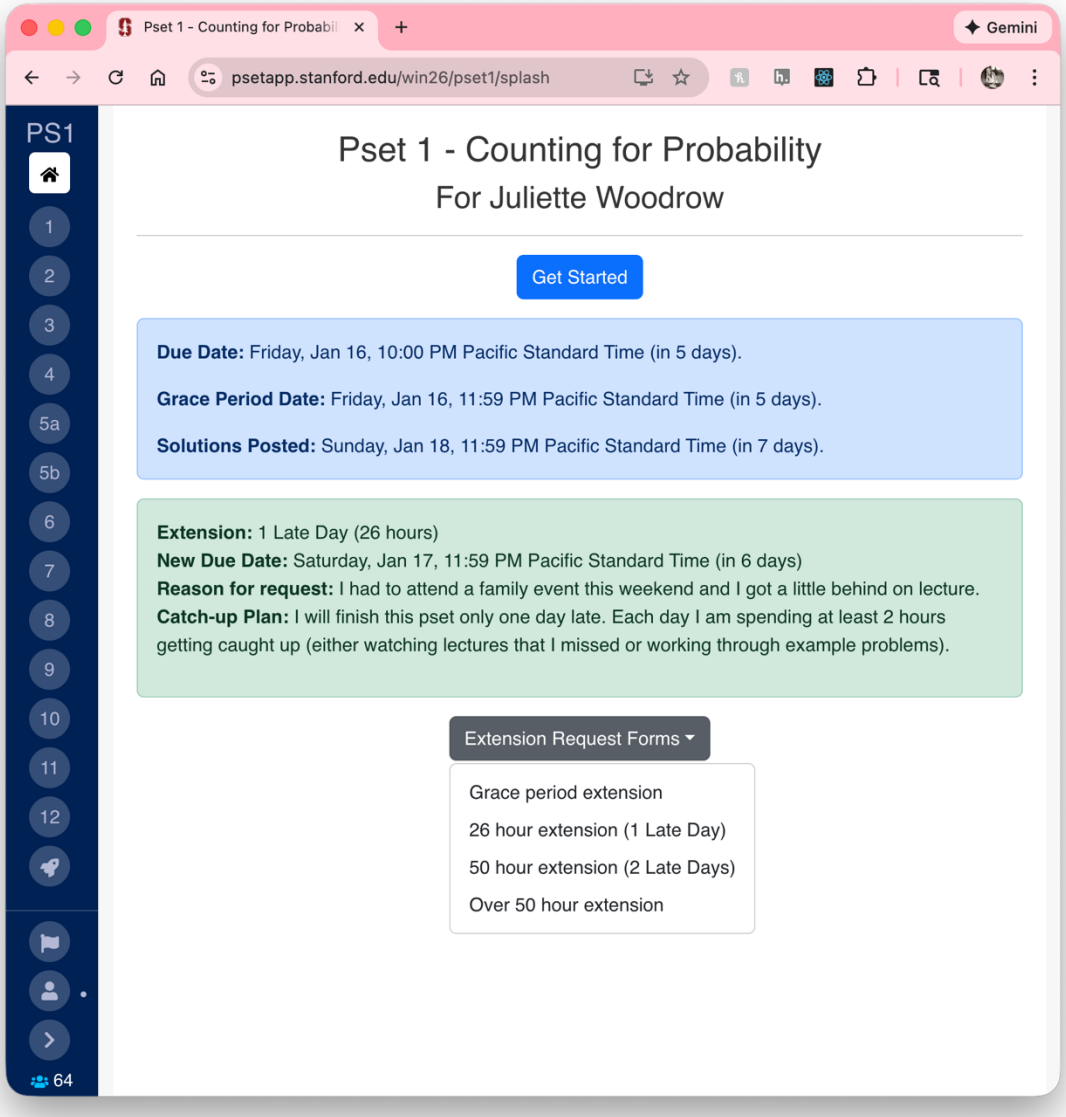
We are going to make history today

Announcements

- Introduce our awesome head TA – Isabel Michel
- Pset 1 is due on Friday ! This went out on Weds.
- Need to tell you about Late Days

- Sections start this week! You should have received an email from your TA with your section time and location.
  - For late sign ups: see post on Ed and sign up ASAP

Late Policy (5 Late Days!)



Three types of extensions:

1. Grace period (2 hours)
2. Can take up to 2 late days (50 hours)
3. After the **hard** deadline (> 2 late days)

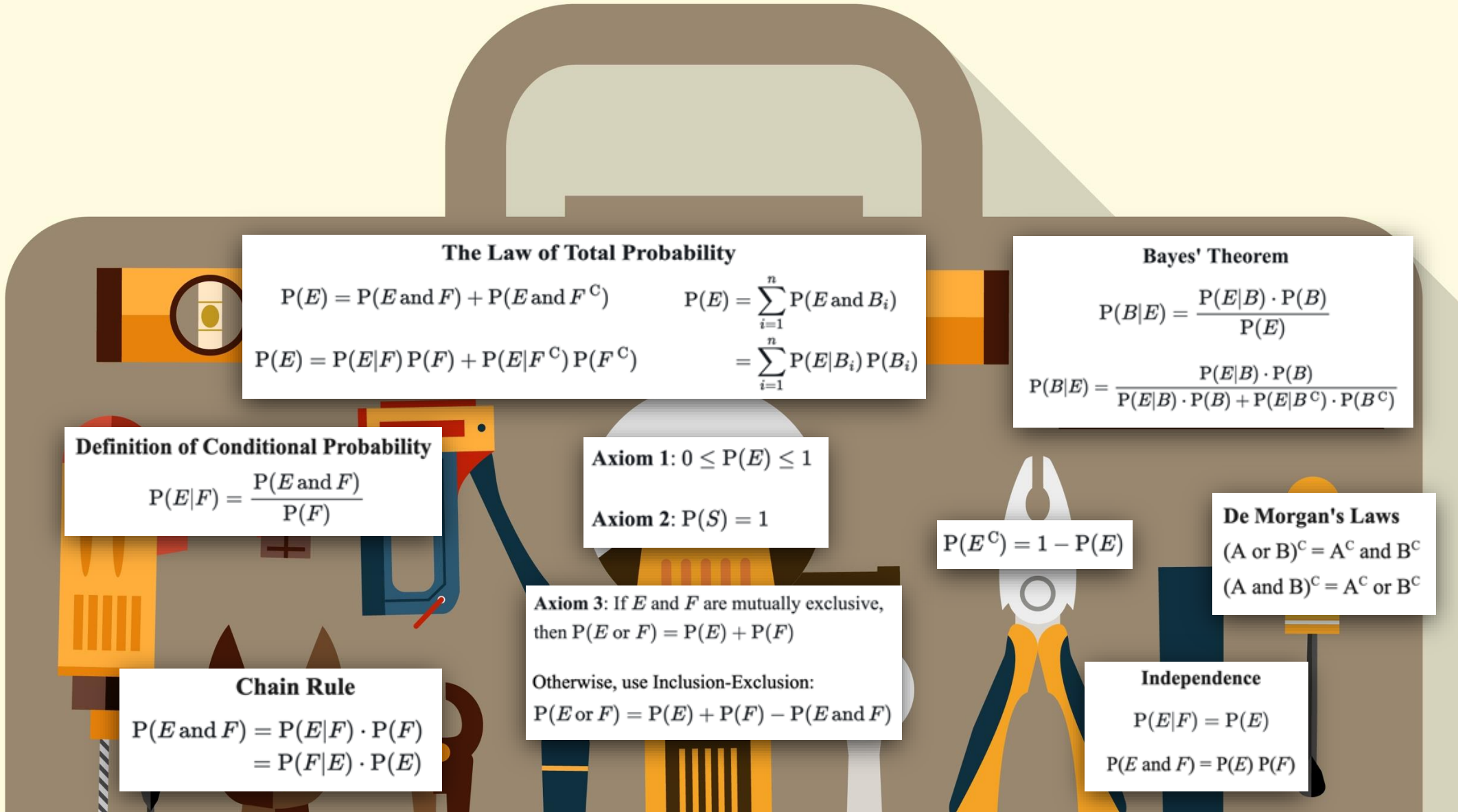
You give them to yourself. Need to talk to us if:

A. You need more than 5 late days / qtr
B. Extension past the hard deadline

But CS109 is a fast class. If you want a long extension I want you to be intentional about how you are going to catch up. Not all late days are created equal (especially before exams).

Review

The Core Probability Toolkit



The Law of Total Probability

$$P(E) = P(E \text{ and } F) + P(E \text{ and } F^C)$$

$$P(E) = P(E|F)\,P(F) + P(E|F^C)\,P(F^C)$$

$$P(E) = \sum_{i=1}^{n} P(E \text{ and } B_i)$$

$$= \sum_{i=1}^{n} P(E|B_i)\,P(B_i)$$

Bayes' Theorem

$$P(B|E) = \frac{P(E|B) \cdot P(B)}{P(E)}$$

$$P(B|E) = \frac{P(E|B) \cdot P(B)}{P(E|B) \cdot P(B) + P(E|B^C) \cdot P(B^C)}$$

Definition of Conditional Probability

$$P(E|F) = \frac{P(E \text{ and } F)}{P(F)}$$

Axiom 1: $0 \le P(E) \le 1$

Axiom 2: $P(S) = 1$

$$P(E^C) = 1 - P(E)$$

De Morgan's Laws

$(A \text{ or } B)^C = A^C \text{ and } B^C$

$(A \text{ and } B)^C = A^C \text{ or } B^C$

Axiom 3: If E and F are mutually exclusive, then $P(E \text{ or } F) = P(E) + P(F)$

Otherwise, use Inclusion-Exclusion:

$$P(E \text{ or } F) = P(E) + P(F) - P(E \text{ and } F)$$

Chain Rule

$$P(E \text{ and } F) = P(E|F) \cdot P(F)$$

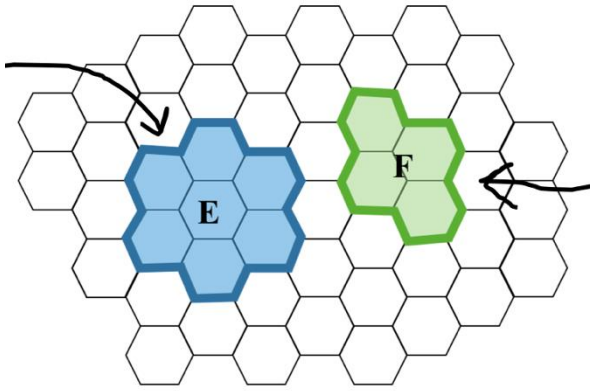
$$= P(F|E) \cdot P(E)$$

Independence

$$P(E|F) = P(E)$$

$$P(E \text{ and } F) = P(E)\,P(F)$$

Review: Last Lecture



Mutually Exclusive Events

make ***OR*** easy:

$$P(A \text{ or } B) = P(A) + P(B)$$

Independent Events

make ***AND*** easy:

$$P(A \text{ and } B) = P(A) \cdot P(B)$$

Sample Space

- **Sample space**, S, is set of all possible outcomes of an experiment
  - Coin flip: S = {Head, Tails}
  - Flipping two coins: S = {[H, H], [H, T], [T, H], [T, T]}
  - Roll of 6-sided die: S = {1, 2, 3, 4, 5, 6}
  - # emails in a day: S = $\{x \mid x \in \mathbf{Z}, x \geq 0\}$ {non-neg. ints}
  - YouTube hrs. in day: S = $\{x \mid x \in \mathbf{R}, 0 \leq x \leq 24\}$

Event Space

- **Event**, E, is some subset of S $\{E \subseteq S\}$
  - Coin flip is heads: E = {Head}
  - ≥ 1 head on 2 coin flips: E = {[H, H], [H, T], [T, H]}
  - Roll of die is 3 or less: E = {1, 2, 3}
  - # emails in a day ≤ 20: $E = \{x \mid x \in \mathbf{Z}, 0 \leq x \leq 20\}$
  - Wasted day {≥ 5 YT hrs.}: $E = \{x \mid x \in \mathbf{R}, 5 \leq x \leq 24\}$

Note: When Ross uses: $\subset$, he really means: $\subseteq$

Equally Likely Outcomes

Some sample spaces have **equally likely outcomes**.

- Coin flip: S = {Head, Tails}
- Flipping two coins: S = {[H, H], [H, T], [T, H], [T, T]}
- Roll of 6-sided die: S = {1, 2, 3, 4, 5, 6}

If we have equally likely outcomes, then P(Each outcome) $= \frac{1}{|S|}$

Therefore $$P(E) = \frac{\#\text{ outcomes in E}}{\#\text{ outcomes in S}} = \frac{|E|}{|S|}$$ {by Axiom 3}

End Review

Learning Goals and Outline

(1) Step rule of counting

(2) Permutations

(3) Combinations

(4) Probabilities with Equally Likely Outcomes

(5) Make history ☺

Counting!

Counting Rules

Counting operations on *n* **distinct** objects

Sort, order matters {perms}

$$n!$$

```
# give me each possible permutation
itertools.permutations([1,2,3,4])

# calculate 20!
math.factorial(20)
```

Choose *k* {combinations}

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}$$

```
# give me each possible combination
itertools.combinations([1,2,3,4,5], 3)

# calculate 10 choose 5
math.comb(10, 5)
```

Counting with Steps

Definition: Step Rule of Counting (aka Product Rule of Counting)

If an experiment has two parts, where the first part can result in one of m outcomes and the second part can result in one of n outcomes regardless of the outcome of the first part, then the total number of outcomes for the experiment is $m \cdot n$.

Counting with Steps

Definition: Step Rule of Counting (aka Product Rule of Counting)

If an experiment has two parts, where the first part can result in one of m outcomes and the second part can result in one of n outcomes regardless of the outcome of the first part, then the total number of outcomes for the experiment is $m \cdot n$.

Definition: Step Rule of Counting with Many Steps

If an experiment has k parts, where the first step has n_1 outcomes and the ith step has n_i outcomes (regardless of the result of any earlier steps), then the total number of outcomes of the experiment is:

$$\text{Number of Outcomes} = \prod_{i=1}^{k} n_i$$

How Many Unique Images?

Each pixel can be one of 17 million distinct colors

(a) 12 million pixels

(b) 300 pixels

(c) 12 pixels

$$(17 \text{ million})^n$$


10^{80}

How Many Unique Images?

Each pixel can be one of 17 million distinct colors

(a) 12 million pixels

$$\approx 10^{86696638}$$

(b) 300 pixels

$$\approx 10^{2167}$$

(c) 12 pixels

$$\approx 10^{86}$$

$$(17 \text{ million})^n$$

Orderings of Letters

How many letter orderings are possible for the following string?

BAYES

Orderings of Letters

BAYES	BAYSE	BAEYS	BAESY	BASYE	BASEY	BYAES	BYASE
BYEAS	BYESA	BYSAE	BYSEA	BEAYS	BEASY	BEYAS	BEYSA
BESAY	BESYA	BSAYE	BSAEY	BSYAE	BSYEA	BSEAY	BSEYA
ABYES	ABYSE	ABEYS	ABESY	ABSYE	ABSEY	AYBES	AYBSE
AYEBS	AYESB	AYSBE	AYSEB	AEBYS	AEBSY	AEYBS	AEYSB
AESBY	AESYB	ASBYE	ASBEY	ASYBE	ASYEB	ASEBY	ASEYB
YBAES	YBASE	YBEAS	YBESA	YBSAE	YBSEA	YABES	YABSE
YAEBS	YAESB	YASBE	YASEB	YEBAS	YEBSA	YEABS	YEASB
YESBA	YESAB	YSBAE	YSBEA	YSABE	YSAEB	YSEBA	YSEAB
EBAYS	EBASY	EBYAS	EBYSA	EBSAY	EBSYA	EABYS	EABSY
EAYBS	EAYSB	EASBY	EASYB	EYBAS	EYBSA	EYABS	EYASB
EYSBA	EYSAB	ESBAY	ESBYA	ESABY	ESAYB	ESYBA	ESYAB
SBAYE	SBAEY	SBYAE	SBYEA	SBEAY	SBEYA	SABYE	SABEY
SAYBE	SAYEB	SAEBY	SAEYB	SYBAE	SYBEA	SYABE	SYAEB
SYEBA	SYEAB	SEBAY	SEBYA	SEABY	SEAYB	SEYBA	SEYAB

Orderings of Letters

Step 1:
Chose first letter

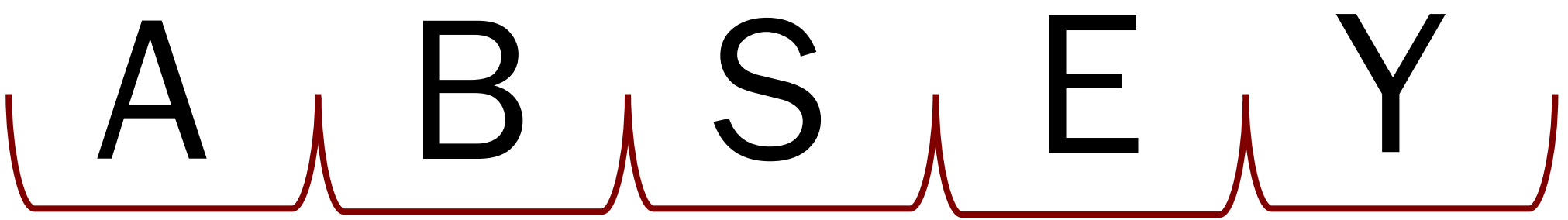
Step 2:
Chose 2nd letter

Step 3:
Chose 3rd letter

Step 4:
Chose 4th letter

Step 5:
Chose 5th letter

Orderings of Letters



Step 1:
Chose first letter
(5 options)

Step 2:
Chose 2nd letter
(4 options)

Step 3:
Chose 3rd letter
(3 options)

Step 4:
Chose 4th letter
(2 options)

Step 5:
Chose 5th letter
(1 option)

To the Code!

```
import itertools

def main():
  letters = ['B', 'A', 'Y', 'E' ,'S']
  perms = set(itertools.permutations(letters))

  for p in perms:
    print(p)
```

Permutations

A permutation is an ordered arrangement of objects.

The number of unique orderings (permutations) of n distinct objects is

$$n! = n \times (n-1) \times (n-2) \times \cdots \times 2 \times 1.$$

Unique 6-digit passcodes with six smudges

How many unique 6-digit passcodes are possible if a phone password uses each of **six** distinct numbers?

Unique 6-digit passcodes with six smudges

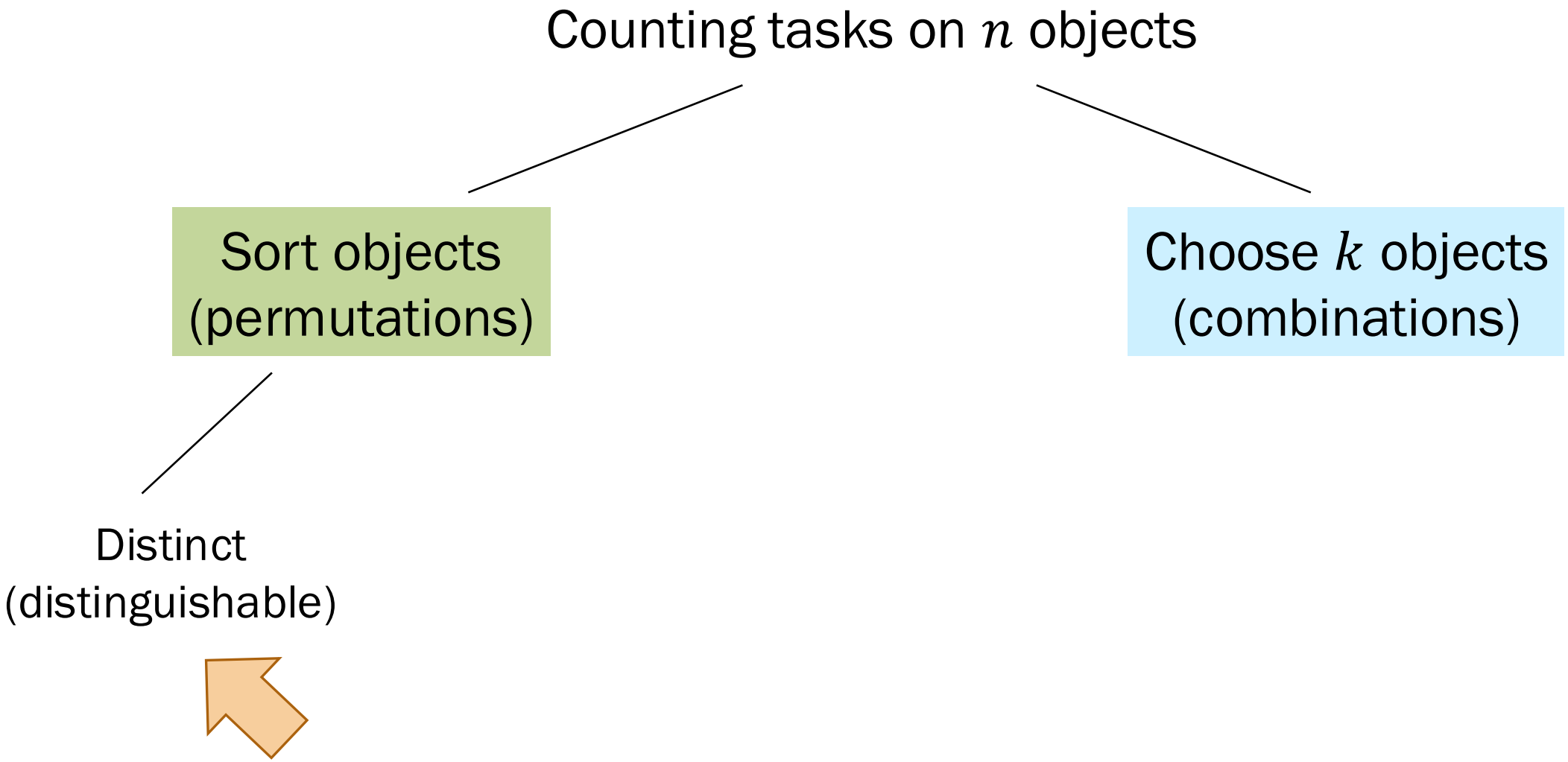
How many unique 6-digit passcodes are possible if a phone password uses each of **six** distinct numbers?

$$\text{Total} = 6! = 720 \text{ passcodes}$$

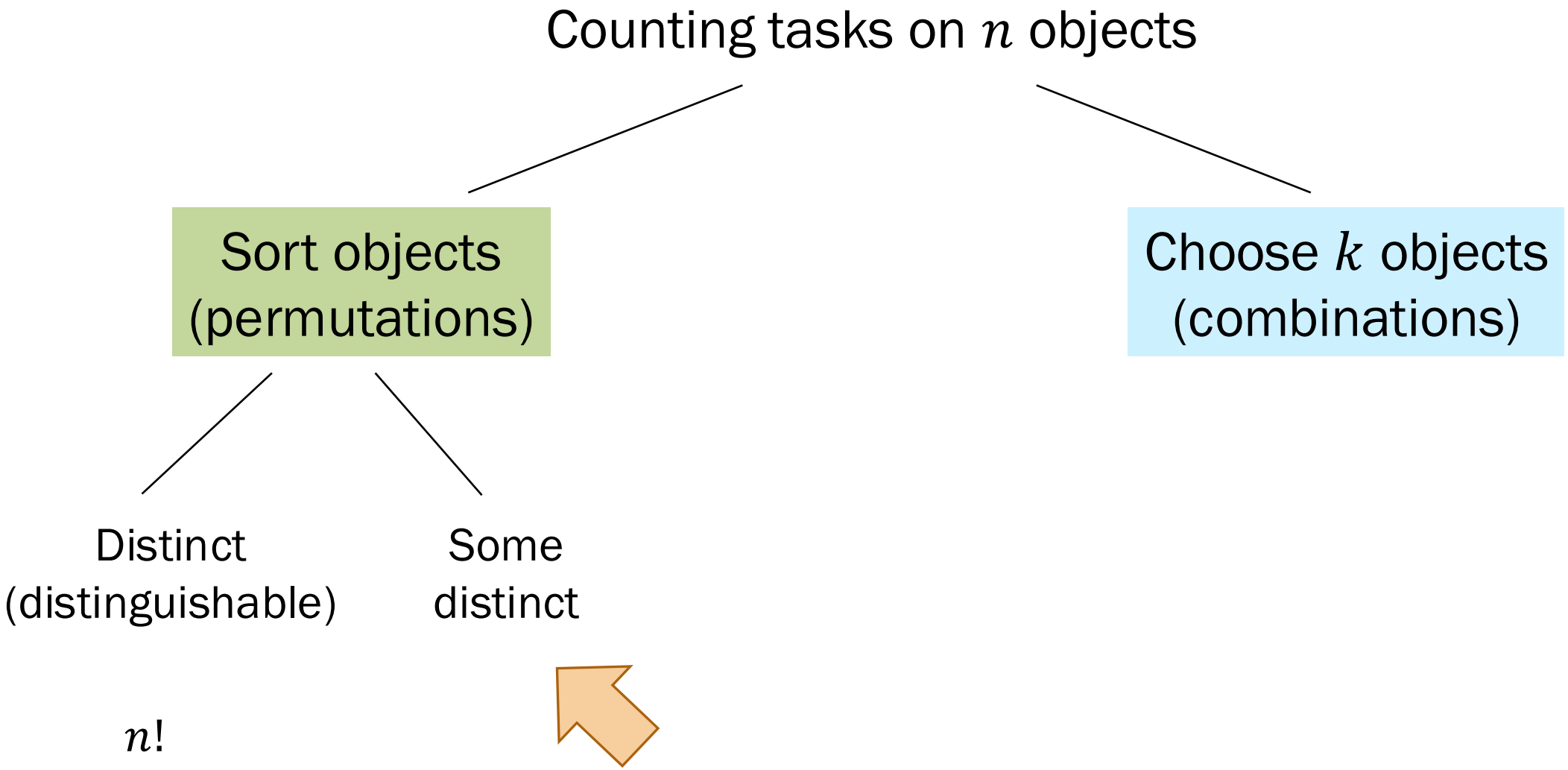
What if you had no smudges, but you knew it was still 6 digits long?

$$\text{Total} = 10^6 = 1 \text{ million passcodes}$$

Summary of Combinatorics



Summary of Combinatorics



Unique Bit Strings

1,0,1,0,0

Sort n distinct objects

Sort n distinct objects

Sort n distinct objects

Steps:

1. Choose 1st can — 5 options
2. Choose 2nd can — 4 options

…

5. Choose 5th can — 1 option

$$\text{Total} = 5 \times 4 \times 3 \times 2 \times 1$$
$$= 120$$

How many ways can we sort coke cans!

Coke1

Coke0

Coke1

Coke0

Coke0

Sort n distinct objects

# of permutations =

Sort semi-distinct objects

Order n distinct objects	$n!$

All distinct

Ayesha Tim Irina Joey Waddie

Some indistinct

Coke Tim Coke Joey Waddie

Sort semi-distinct objects

How do we find the number of permutations considering some objects are indistinct?

By the product rule of counting (aka step rule), permutations of distinct objects is a two-step process:

permutations of distinct objects = permutations considering some objects are indistinct × Permutations of just the indistinct objects

Sort semi-distinct objects

How do we find the number of permutations considering some objects are indistinct?

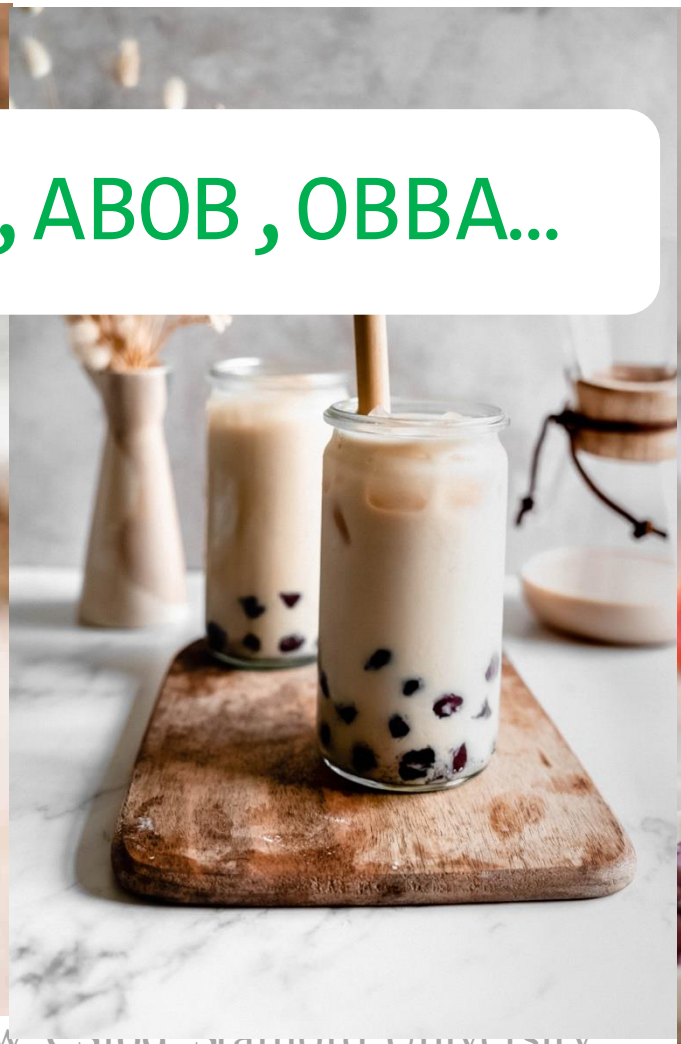
By the product rule of counting (aka step rule), permutations of distinct objects is a two-step process:

$$\frac{\text{permutations of distinct objects}}{\text{Permutations of just the indistinct objects}} = \text{permutations considering some objects are indistinct}$$

BOBA

How many *different* orderings of letters are possible for the string **BOBA**?

BOBA, ABOB, OBBA...



Strings

Is this just a regular permutation? 4! = 24 unique permutations!

boba
boab
bboa
bbao
baob
babo
obba
obab
obba
obab
oabb
oabb

bboa
bbao
boba
boab
babo
baob
abob
abbo
aobb
aobb
abbo
abob

Strings

Is this just a regular permutation? 4! = 24 unique permutations!

boba
boab
bboa
bbao
baob
babo
obba
obab
obba
obab
oabb
oabb

bboa
bbao
boba
boab
babo
baob
abob
abbo
aobb
aobb
abbo
abob

Strings

Is this just a regular permutation? 4! = 24 unique permutations!

boba
boab
bboa
bbao
baob
babo
obba
obab
obba
obab
oabb
oabb

bboa
bbao
boba
boab
babo
baob
abob
abbo
aobb
aobb
abbo
abob

Strings

Is this just a regular permutation? 4! = 24 unique permutations!

boba
boab
bboa
bbao
baob
babo
obba
obab
obba
obab
oabb
oabb

bboa
bbao
boba
boab
babo
baob
abob
abbo
aobb
aobb
abbo
abob

To the Code!

baob
bbao
obba
oabb
boab
babo
abbo
aobb
boba
abob
bboa
obab

```
import itertools

def main():
  letters = ['b','o','b','a']
  perms = set(itertools.permutations(letters))
  for perm in perms:
    pretty_perm = "".join(perm)
    print(pretty_perm)
```

```
import math

def main():
  n = math.factorial(4)
  d = math.factorial(2)
  print(n / d)
```

To the Code!

baob
bbao
obba
oabb
boab
babo
abbo
aobb
boba
abob
bboa
obab

```
import itertools

def main():
  letters = ['b','o','b','a']
  perms = set(itertools.permutations(letters))
  for perm in perms:
    pretty_perm = "".join(perm)
    print(pretty_perm)
```

```
import math

def main():
  n = math.factorial(4)
  d = math.factorial(2)
  print(n / d)
```

To the Code!

baob
bbao
obba
oabb
boab
babo
abbo
aobb
boba
abob
bboa
obab

```
import itertools

def main():
  letters = ['b','o','b','a']
  perms = set(itertools.permutations(letters))
  for perm in perms:
    pretty_perm = "".join(perm)
    print(pretty_perm)
```

```
import math

def main():
  n = math.factorial(4)
  d = math.factorial(2)
  print(n / d)
```

General approach to counting permutations

When there are n objects such that

n_1 are the same (indistinguishable or indistinct), and

n_2 are the same, and

…

n_r are the same,

The number of unique orderings (permutations) is

$$\frac{n!}{n_1!\, n_2! \cdots n_r!}.$$

For each group of indistinct objects,
Divide by the overcounted permutations.

Sort semi-distinct objects

Order n semi-distinct objects $\frac{n!}{n_1! n_2! \cdots n_r!}$

How many permutations?

Strings

Order n semi-distinct objects $\frac{n!}{n_1! n_2! \cdots n_r!}$

How many letter orderings are possible for the following strings?

1. BOBA

2. MISSISSIPPI

This is Jerry's dog, Doris. She puts her little Doris paw up to her chin when she's thinking.

Strings

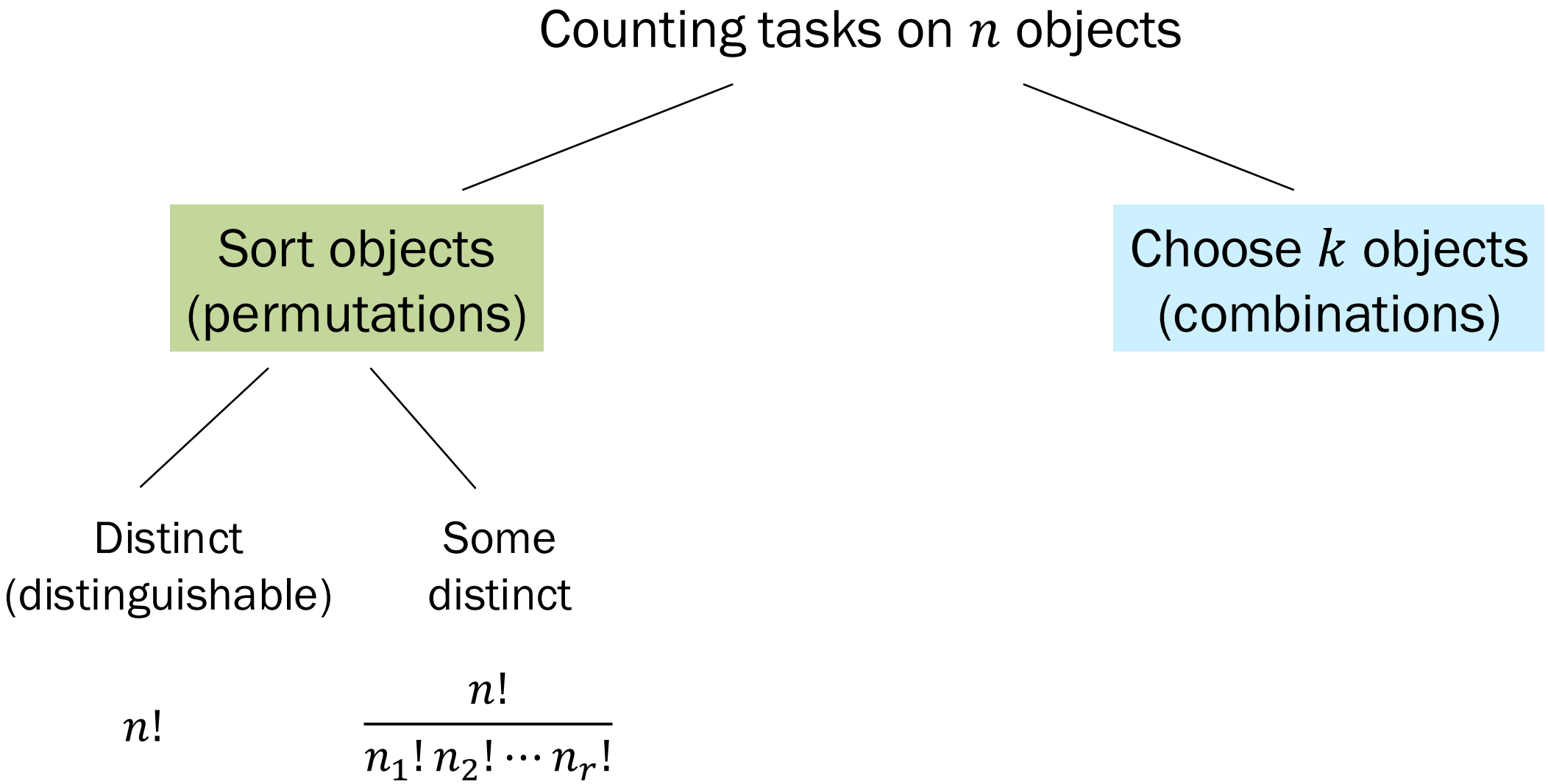
Order n semi-distinct objects $\frac{n!}{n_1!\,n_2!\cdots n_r!}$

How many letter orderings are possible for the following strings?

1. BOBA $= \frac{4!}{2!} = 12$

2. MISSISSIPPI $= \frac{11!}{1!4!4!2!} = 34{,}650$

Summary of Combinatorics



Combinations

Summary of Combinatorics

Counting tasks on n objects

Sort objects (permutations)

Distinct (distinguishable)

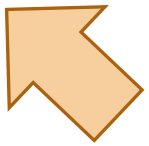
$n!$

Some distinct

$$\frac{n!}{n_1!\, n_2! \cdots n_r!}$$

Choose k objects (combinations)

Distinct



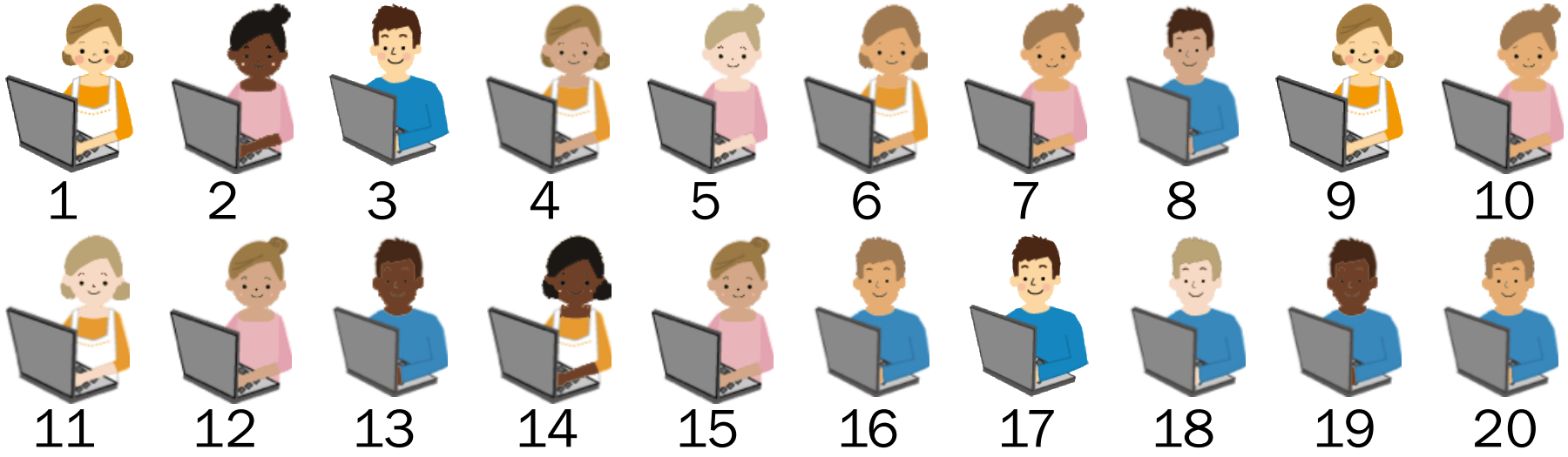
Combinations with cake

There are $n = 20$ people.
How many ways can we choose $k = 5$ people to get cake?

Consider the following generative process...

Combinations with cake

There are $n = 20$ people.
How many ways can we choose $k = 5$ people to get cake?

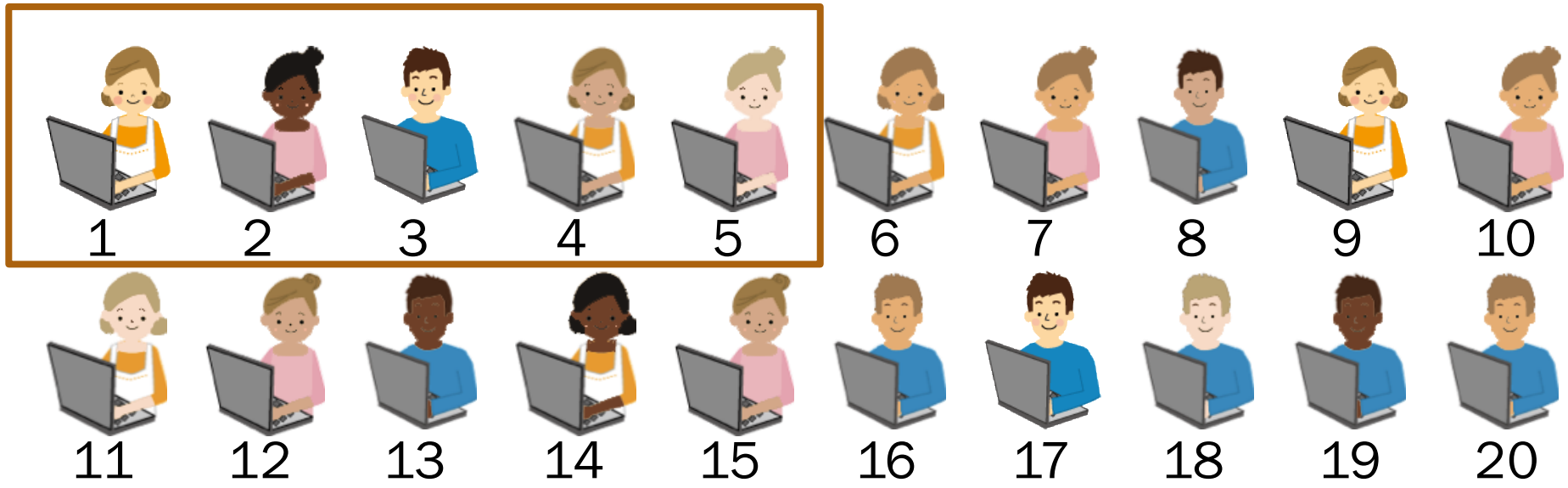


1. n people get in line

   $n!$ ways

Combinations with cake

There are $n = 20$ people.
How many ways can we choose $k = 5$ people to get cake?



1. n people get in line

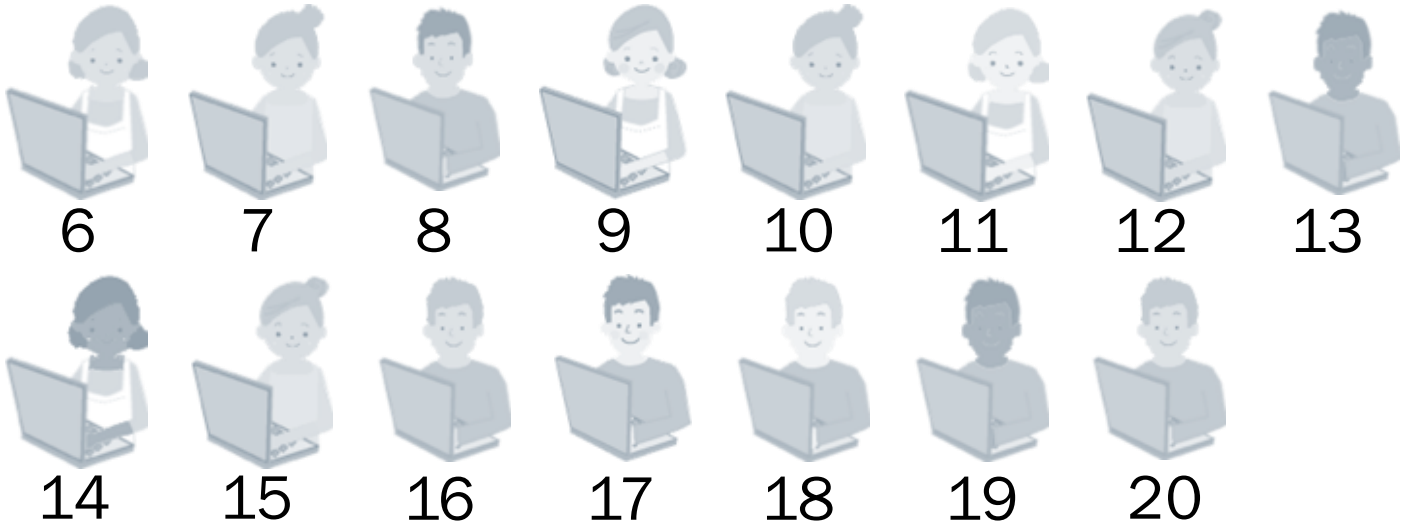
   $n!$ ways

2. Put first k in cake room

   1 way

Combinations with cake

There are $n = 20$ people.
How many ways can we choose $k = 5$ people to get cake?



1. n people get in line

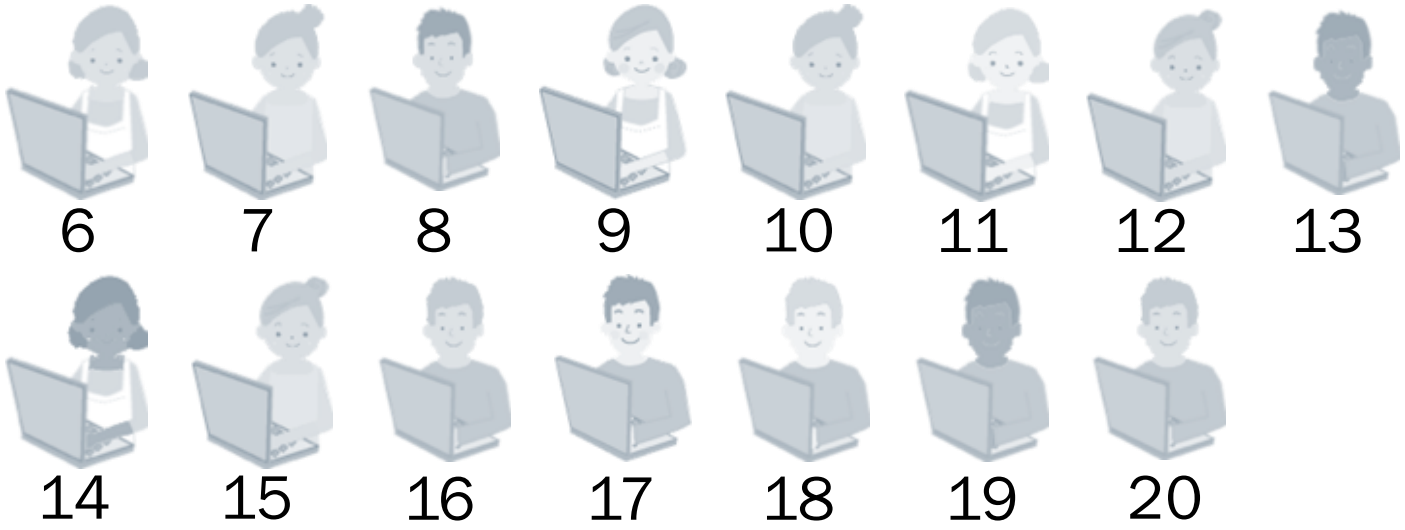
   $n!$ ways

2. Put first k in cake room

   1 way

Combinations with cake

There are $n = 20$ people.
How many ways can we choose $k = 5$ people to get cake?



1. n people get in line

   $n!$ ways

2. Put first k in cake room

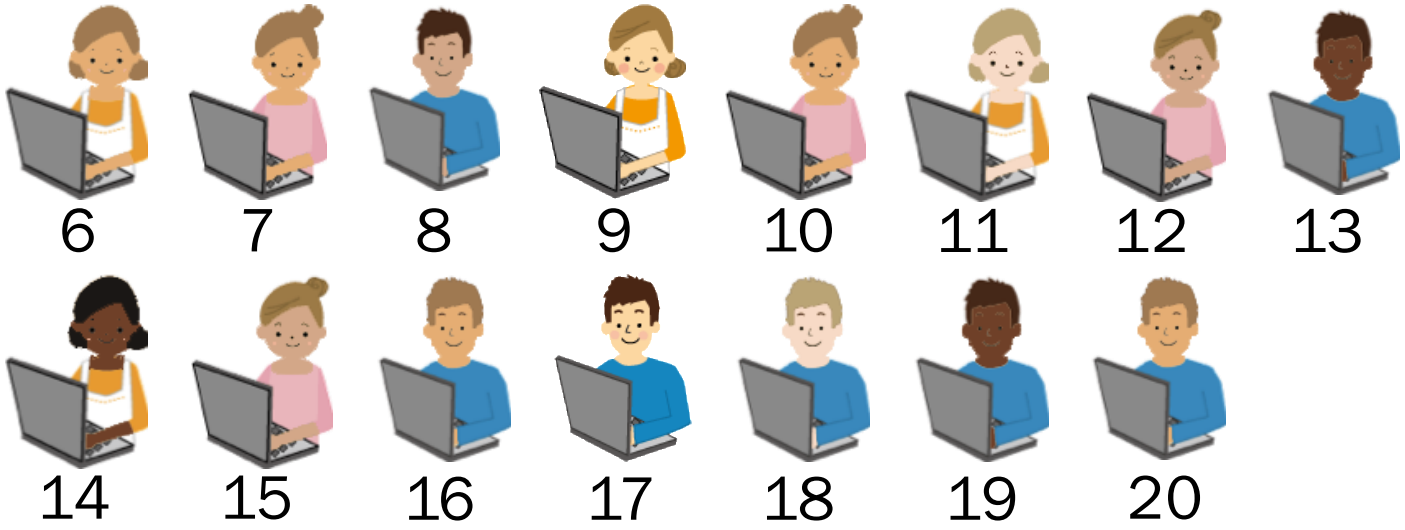
   1 way

3. Allow cake group to mingle

   $k!$ different permutations lead to the same mingle

Combinations with cake

There are $n = 20$ people.
How many ways can we choose $k = 5$ people to get cake?



1. n people get in line

   $n!$ ways

2. Put first k in cake room

   1 way

3. Allow cake group to mingle

   $k!$ different permutations lead to the same mingle

4. Allow non-cake group to mingle

Combinations with cake

There are $n = 20$ people.
How many ways can we choose $k = 5$ people to get cake?

1. n people get in line

   $n!$ ways

2. Put first k in cake room

   1 way

3. Allow cake group to mingle

   $k!$ different permutations lead to the same mingle

4. Allow non-cake group to mingle

   $(n-k)!$ different permutations lead to the same mingle

Combinations

A combination is an unordered selection of k objects from a set of n distinct objects.

The number of ways of making this selection is

$$\frac{n!}{k!\,(n-k)!} = n! \times 1 \times \frac{1}{k!} \times \frac{1}{(n-k)!}$$

1. Order n distinct objects
2. Take first k as chosen
3. Overcounted: any ordering of chosen group is same choice
4. Overcounted: any ordering of unchosen group is same choice

Combinations

A combination is an unordered selection of k objects from a set of n distinct objects.

The number of ways of making this selection is

$$\frac{n!}{k!\,(n-k)!} = n! \times 1 \times \frac{1}{k!} \times \frac{1}{(n-k)!} = \binom{n}{k}$$

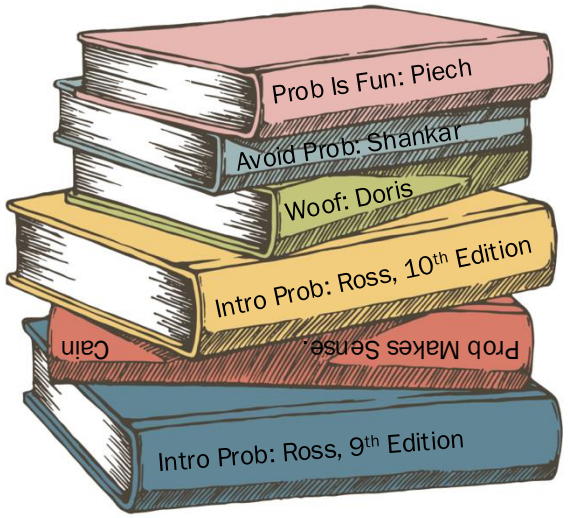
Binomial coefficient

Fun Fact: $\binom{n}{k} = \binom{n}{n-k}$

Probability textbooks

Choose k of n distinct objects $\binom{n}{k}$

How many ways are there to choose 3 books from a set of 6 distinct books?



$$\binom{6}{3} = \frac{6!}{3!\,3!} = 20 \text{ ways}$$

To the code!

How many unique hands of 5 cards are there in a 52 card deck?

```
def main():
  cards = make_deck()
  all_hands = itertools.combinations(cards, 5)
  for hand in all_hands:
    print(hand)
```

```
def main():
  total = math.comb(52, 5)
  print(total)
```

Probabilities With Equally Likely Outcomes

Straight Poker Hand

- Consider 5 card poker hands.
  - "straight" is 5 consecutive rank cards of any suit
  - What is P(straight)?

Straight Poker Hand

- Consider 5 card poker hands.
  - "straight" is 5 consecutive rank cards of any suit
  - What is P(straight)?

What is an example of one outcome?

Is each outcome equally likely?

$$|S| = \binom{52}{5}$$

$$|E| = 10 \cdot \binom{4}{1}^5$$

$$P(\text{straight}) = \frac{|E|}{|S|} = \frac{10 \cdot \binom{4}{1}^5}{\binom{52}{5}} \approx 0.00394$$

Bunnies and Foxes

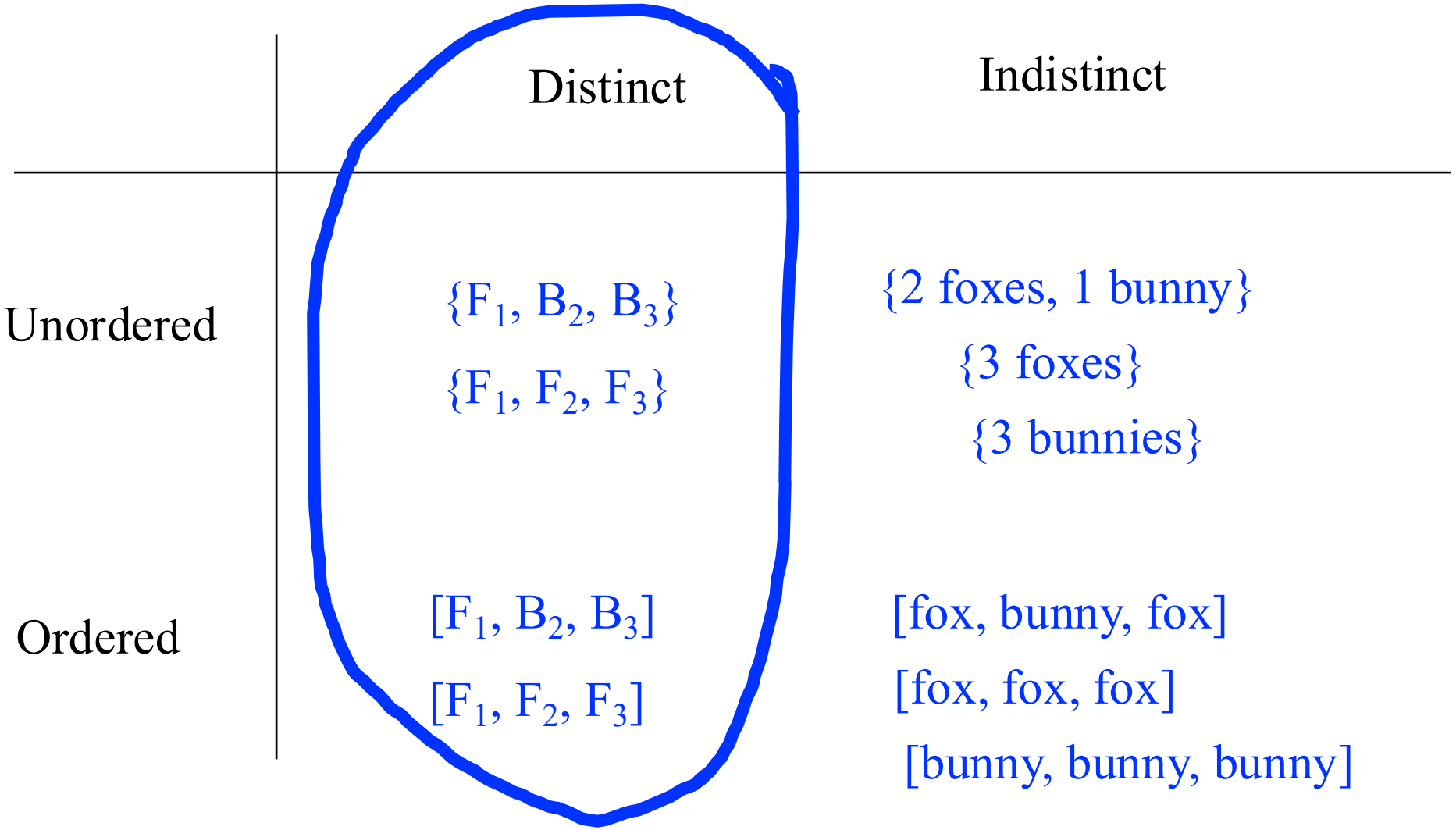
- 4 bunnies and 3 foxes in a toy box. 3 drawn.
  - What is P(1 bunny and 2 fox drawn)?

Equally likely sample space? Thought experiment

3 foxes

4 bunnies

The Choice of Sample Space is Yours!



Which choice will lead to equally likely outcomes?

Bunnies and Foxes

- 4 bunnies and 3 foxes in a Bag. 3 drawn.
  - What is P(1 bunny and 2 foxes drawn)?
- **Ordered and Distinct**:
  - Pick 3 ordered items: |S| = 7 * 6 * 5 = 210
  - Pick bunny as either 1st, 2nd, or 3rd item:
    |E| = (4 * 3 * 2) + (3 * 4 * 2) + (3 * 2 * 4) = 72
  - P(1 bunny, 2 foxes) = 72/210 = 12/35
- **Unordered and Distinct**:
  - $|S| = \binom{7}{3} = 35$
  - $|E| = \binom{4}{1}\binom{3}{2} = 12$
  - P(1 bunny, 2 foxes) = 12/35

Make indistinct items **distinct** to get equally likely sample space outcomes

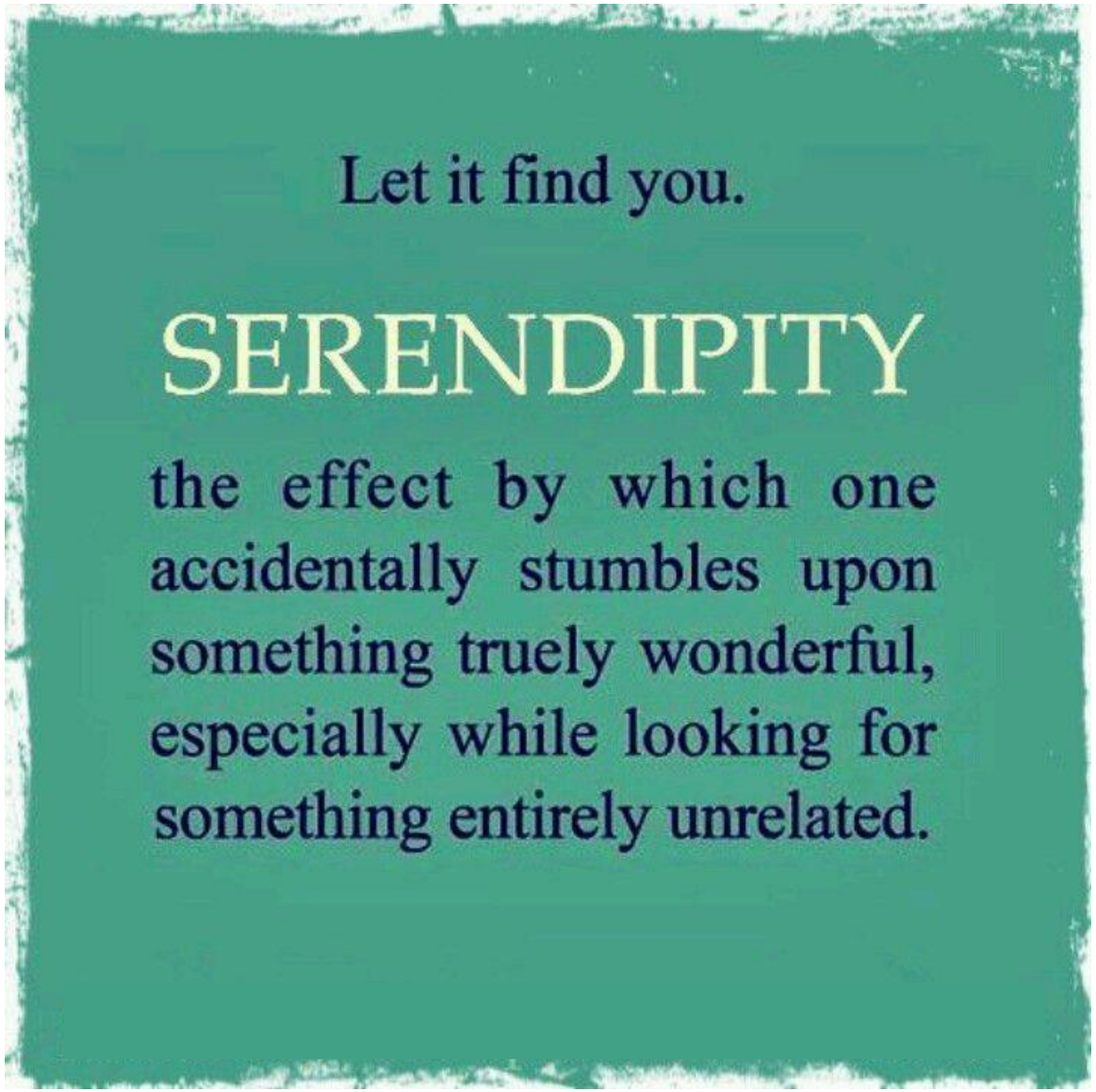
*You will need to use this “trick” with high probability

When approaching an "**equally likely probability**" problem, start by defining **sample spaces** and **event spaces**.

Chip Defect Detection

- *n* chips manufactured, 1 of which is defective.
- *k* chips randomly selected from *n* for testing.
  - What is P(defective chip is in *k* selected chips)?
- |S| = $\binom{n}{k}$
- |E| = $\binom{1}{1}\binom{n-1}{k-1}$
- P(defective chip is in *k* selected chips)

$$= \frac{\binom{1}{1}\binom{n-1}{k-1}}{\binom{n}{k}} = \frac{\frac{(n-1)!}{(k-1)!(n-k)!}}{\frac{n!}{k!(n-k)!}} = \frac{k}{n}$$

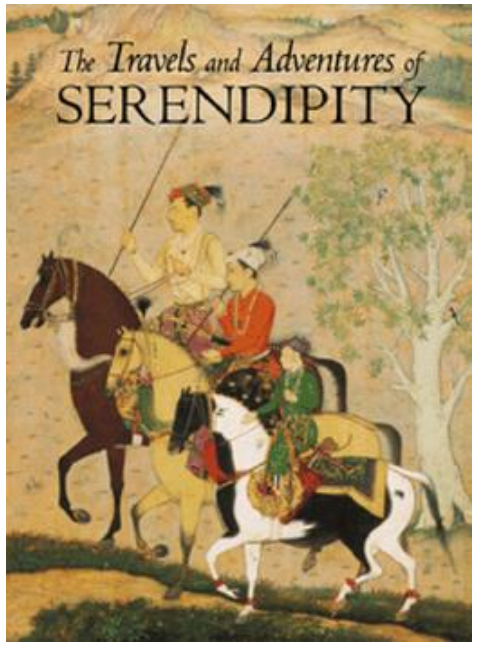

Let it find you.
SERENDIPITY
the effect by which one accidentally stumbles upon something truely wonderful, especially while looking for something entirely unrelated.

WHEN YOU MEET YOUR BEST FRIEND

Somewhere you didn't expect to.

Serendipity

- Say the population of Stanford is 17,000 people
  - You are friends with 100
  - Walk into a room, see 450 random people.
  - What is the probability that you see someone you know?
  - Assume you are equally likely to see each person at Stanford



Many times it is easier to calculate $P(E^C)$.

Trailing the dovetail shuffle to it’s lair – Persi Diaconis

Trailing the Dovetail Shuffle to Its Lair

The Annals of Applied Probability
1992, Vol. 2, No. 2, 294–313

TRAILING THE DOVETAIL SHUFFLE TO ITS LAIR

BY DAVE BAYER[1] AND PERSI DIACONIS[2]

Columbia University and Harvard University

We analyze the most commonly used method for shuffling cards. The main result is a simple expression for the chance of any arrangement after any number of shuffles. This is used to give sharp bounds on the approach to randomness: $\frac{3}{2}\log_2 n + \theta$ shuffles are necessary and sufficient to mix up n cards.

Key ingredients are the analysis of a card trick and the determination of the idempotents of a natural commutative subalgebra in the symmetric group algebra.

1. Introduction. The dovetail, or riffle shuffle is the most commonly used method of shuffling cards. Roughly, a deck of cards is cut about in half and then the two halves are riffled together. Figure 1 gives an example of a riffle shuffle for a deck of 13 cards.

A mathematically precise model of shuffling was introduced by Gilbert and Shannon [see Gilbert (1955)] and independently by Reeds (1981). A deck of n cards is cut into two portions according to a binomial distribution; thus, the chance that k cards are cut off is $\binom{n}{k}/2^n$ for $0 \le k \le n$. The two packets are then riffled together in such a way that cards drop from the left or right heaps with probability proportional to the number of cards in each heap. Thus, if there are A and B cards remaining in the left and right heaps, then the chance that the next card will drop from the left heap is $A/(A+B)$. Such shuffles are easily described backwards: Each card has an equal and independent chance of being pulled back into the left or right heap. An inverse riffle shuffle is illustrated in Figure 2.

Experiments reported in Diaconis (1988) show that the Gilbert–Shannon–Reeds (GSR) model is a good description of the way real people shuffle real cards. It is natural to ask how many times a deck must be shuffled to mix it up. In Section 3 we prove:

THEOREM 1. *If n cards are shuffled m times, then the chance that the deck is in arrangement π is $\binom{2^m+n-r}{n}/2^{mn}$, where r is the number of rising sequences in π.*

Rising sequences are defined and illustrated in Section 2 through the analysis of a card trick. Section 3 develops several equivalent interpretations of

Received January 1990; revised May 1991.
[1]Partially supported by the Alfred P. Sloan Foundation, by ONR contract N00014-87-K0214 and by NSF Grant DMS-90-06116.
[2]Partially supported by NSF Grant DMS-89-05874.
AMS 1980 *subject classifications.* 20B30, 60B15, 60C05, 60F99.
Key words and phrases. Card shuffling, symmetric group algebra, total variation distance.

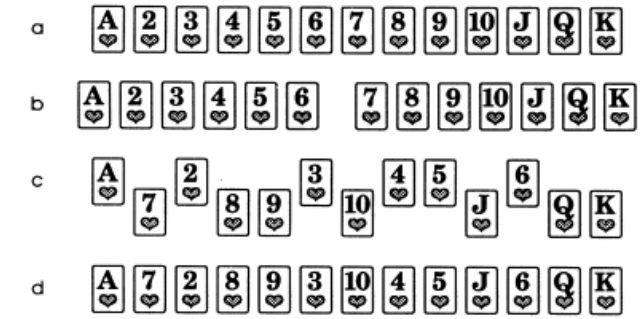


FIG. 1. *A riffle shuffle.* (a) *We begin with an ordered deck.* (b) *The deck is divided into two packets of similar size.* (c) *The two packets are riffled together.* (d) *The two packets can still be identified in the shuffled deck as two distinct "rising sequences" of face values.*

the GSR distribution for riffle shuffles, including a geometric description as the motion of n points dropped at random into the unit interval under the baker's transformation $x \to 2x \pmod 1$. This leads to a proof of Theorem 1.

Section 3 also relates shuffling to some developments in algebra. A permutation π has a descent at i if $\pi(i) > \pi(i+1)$. A permutation π has r rising sequences if and only if π^{-1} has $r-1$ descents. Let

$$A_k = \sum_{\pi \text{ has } k \text{ descents}} \pi$$

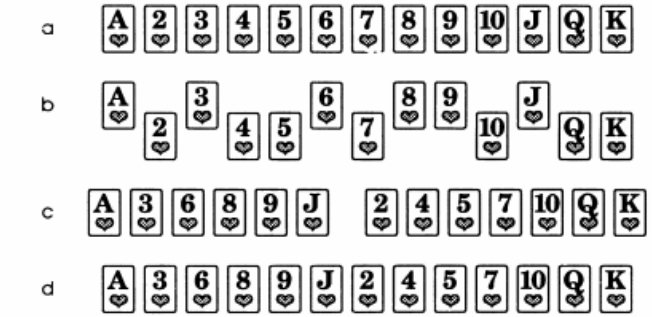


FIG. 2. *An inverse riffle shuffle.* (a) *We begin with a sorted deck.* (b) *Each card is moved one way or the other uniformly at random, to "pull apart" a riffle shuffle and retrieve two packets.* (c) *The two packets are placed in sequence.* (d) *The two packets can still be identified in the shuffled deck; they are separated by a "descent" in the face values. This shuffle is inverse to the shuffle diagrammed in Figure 1.*

Trailing the Dovetail Shuffle to Its Lair (Simple)

- What is the probability that a **single** shuffle is different from yours?
  - |S| = 52!
  - |E| = 52! – 1
  - P(different) = (52!-1)/52!

P(different) > 0.9999999999…

P(same) < 0.0000000…

About 68 “9”s

Trailing the Dovetail Shuffle to Its Lair (Medium)

- What is the probability that 2 shuffles are different from yours?
  - $|S| = (52!)^2$
  - $|E| = (52! - 1)^2$
  - P(no deck matching yours) = $(52!-1)^2/(52!)^2$

Trailing the Dovetail Shuffle to Its Lair (Full)

- What is the probability that in the *n* shuffles seen since the start of time, yours is unique?
  - $|S| = (52!)^n$
  - $|E| = (52! - 1)^n$
  - P(no deck matching yours) = $(52!-1)^n/(52!)^n$

- For n = 10^{20},
  - P(deck matching yours) < 0.000000001

* Assume 7 billion people have been shuffling cards once a second since 52 deck cards were invented (see Topkapı deck)

That’s all for today ☺

Trailing the Dovetail Shuffle to Its Lair (Full)

```
from decimal import *
import math

n = math.pow(10, 20)
card_perms = math.factorial(52)
denominator = card_perms
numerator = card_perms - 1

# decimal library because these are tiny numbers
getcontext().prec = 100 # increase precision
log_numer = Decimal(numerator).ln()
log_denom = Decimal(denominator).ln()
log_pr = log_numer - log_denom

# approximately -1.24E-68
print(log_pr)
```

$$\log\left(\frac{52! - 1}{52!}\right)$$

Let’s Gamble

Need to explain the two hands together…

Telling in Cards

Your opponent gets excited if they are winning…

Probability of a tell given they **have a winning** hand: **0.5**

Probability of a tell given they don't have a winning hand: **0.1**

The Tell Game

```
You are playing a game. Your opponent has two unseen cards. If either is an ace you lose...
The following cards are seen by you (ie are not in opponent hand):
3H
2S
3D
5S
9D
KD
AC
Opponent does *not* have an excited tell...
Do you choose to play? yes
```

Probability of a tell given they **have a winning** hand: **0.5**

Probability of a tell given they don't have a winning hand: **0.1**